



Infor LN Public Interfaces & Process Extensions Reference Guide (On-premises)

Release 10.8

Copyright © 2025 Infor

Important Notices

The material contained in this publication (including any supplementary information) constitutes and contains confidential and proprietary information of Infor.

By gaining access to the attached, you acknowledge and agree that the material (including any modification, translation or adaptation of the material) and all copyright, trade secrets and all other right, title and interest therein, are the sole property of Infor and that you shall not gain right, title or interest in the material (including any modification, translation or adaptation of the material) by virtue of your review thereof other than the non-exclusive right to use the material solely in connection with and the furtherance of your license and use of software made available to your company from Infor pursuant to a separate agreement, the terms of which separate agreement shall govern your use of this material and all supplemental related materials ("Purpose").

In addition, by accessing the enclosed material, you acknowledge and agree that you are required to maintain such material in strict confidence and that your use of such material is limited to the Purpose described above. Although Infor has taken due care to ensure that the material included in this publication is accurate and complete, Infor cannot warrant that the information contained in this publication is complete, does not contain typographical or other errors, or will meet your specific requirements. As such, Infor does not assume and hereby disclaims all liability, consequential or otherwise, for any loss or damage to any person or entity which is caused by or relates to errors or omissions in this publication (including any supplementary information), whether such errors or omissions result from negligence, accident or any other cause.

Without limitation, U.S. export control laws and other applicable export and import laws govern your use of this material and you will neither export or re-export, directly or indirectly, this material nor any related materials or supplemental information in violation of such laws, or use such materials for any purpose prohibited by such laws.

Trademark Acknowledgements

The word and design marks set forth herein are trademarks and/or registered trademarks of Infor and/or related affiliates and subsidiaries. All rights reserved. All other company, product, trade or service names referenced may be registered trademarks or trademarks of their respective owners.

Publication Information

Release: Infor LN 10.8

Publication date: December 4, 2025

Contents

About this guide	45
Intended audience	45
Related documents	45
Contacting Infor	45
Chapter 1 Introduction	46
Public Interfaces	46
Public Interfaces explained	46
How to request a new Public Interface	46
Public Interface example	47
Available Public Interfaces	48
Public Interface to call BDE methods	48
Proxy DLL	49
Coding example	49
Process Extensions	53
Process Extensions explained	53
How to request a new Process Extension	54
Available Process Extensions	54
Chapter 2 Public Interfaces for Extensibility	55
Public Interfaces for Exception	55
Exception.Delete	55
Exception.GetMessage	55
Exception.NumberOfMessages	56
Public Interfaces for ProcessingOptionSet	56
ProcessingOptionSet.Create	57
ProcessingOptionSet.Delete	58
Chapter 3 Public Interfaces for Common	59
Public Interfaces for Address	59
Address.Create	59

Address.StartDetail	61
Address.StartPrintByBusinessPartner	62
Public Interfaces for Warehouse	63
Warehouse.StartOverview	63
Public Interfaces for Common	64
Common.CalculateDiscountAmounts	65
Common.CalculateDueDates	66
Common.ConvertAmount	67
Common.ConvertAmountToWords	68
Common.ConvertISODurationToSeconds	69
Common.ConvertLeadTimeToISODuration	70
Common.ConvertPrice	71
Common.ConvertQuantity	71
Common.ConvertTime	72
Common.GenerateFirstFreeNumber	73
Common.GetAdditionalEntityInfo	74
Common.GetConversionFactor	74
Common.GetCurrencyRates	75
Common.GetFinancialCompanyOfEntity	76
Common.GetFormattedAddress	77
Common.GetISOCCodeOfLNLanguage	78
Common.GetParameters	79
Common.RoundAmount	80
Common.RoundQuantity	80
Common.UpdateApprovedConversionFactor	81
Common.ValidateNumberGroupSeries	82
Public Interfaces for Contact	82
Contact.StartOverview	83
Public Interfaces for BusinessPartner	84
BusinessPartner.CheckOverdueInvoices	84
BusinessPartner.GetBuyFromData	85
BusinessPartner.GetCreditLimitAndBalances	86
BusinessPartner.GetGeneralData	87
BusinessPartner.GetInvoiceFromData	88
BusinessPartner.GetInvoiceToData	89
BusinessPartner.GetPayByData	90
BusinessPartner.GetPayToData	91
BusinessPartner.GetShipFromData	93

BusinessPartner.GetShipToData	94
BusinessPartner.GetSoldToData	95
BusinessPartner.StartEasyEntry	97
BusinessPartner.StartMultiMain	98
BusinessPartner.StartOverview	99
Public Interfaces for BuyFromBusinessPartner	100
BuyFromBusinessPartner.StartOverview.....	100
Public Interfaces for InvoiceToBusinessPartner	101
InvoiceToBusinessPartner.StartDetail	102
Public Interfaces for HoldReason	102
HoldReasons.StartOverview	103
Public Interfaces for BillingCycle	103
BillingCycle.GetInvoicedate.....	104
Public Interfaces for Series.....	105
Series.StartOverview	105
Public Interfaces for Task	106
Task.StartMultiMain	106
Chapter 4 Public Interfaces for Calendar	107
Public Interfaces for Calendar	107
Calendar.CalculateDaysFromHours	107
Calendar.GetPeriodCapacity.....	108
Calendar.GetWorkingDays.....	109
Calendar.PlanLeadTimeBackward.....	109
Calendar.PlanLeadTimeForward	111
Chapter 5 Public Interfaces for IntercompanyTrade.....	113
Public Interfaces for IntercompanyTradeOrder	113
IntercompanyTradeOrder.PresentForWarehouseOrderLine	113
Chapter 6 Public Interfaces for Item	115
Public Interfaces for Item	115
Item.ConvertItemToPlanItem	116
Item.ConvertPlanItemToItem	116
Item.Copy	117
Item.CopySiteData	118
Item.CreateSerialNumber.....	119
Item.GetCostingData	120

Item.GetData	121
Item.GetLotSerialRegistration	123
Item.GetOrderingData	123
Item.GetPlanningTimeFenceDate	124
Item.GetProductionData	125
Item.GetPurchaseData	127
Item.GetRevision	129
Item.GetSalesData	129
Item.GetServiceData	131
Item.GetSupplier	132
Item.GetWarehousingData	133
Item.InsertSerialNumber	135
Item.ProjectPegAllowed	135
Item.StartCopy	136
Item.StartDetail	137
Item.StartMultiMain	138
Item.StartOverview	139
Item.StartWhereUsedComponent	140
Public Interfaces for ItemBySite	141
ItemBySite.StartDetail	141
ItemBySite.StartOverview	142
Public Interfaces for ItemPurchase	143
ItemPurchase.StartDetail	143
ItemPurchase.StartMultiMain	144
ItemPurchase.StartOverview	144
Public Interfaces for ItemPurchaseBySite	145
ItemPurchaseBySite.StartDetail	145
Public Interfaces for ItemPurchaseBusinessPartner	146
ItemPurchaseBusinessPartner.CalculateLeadTimes	147
ItemPurchaseBusinessPartner.GetSafetyTime	148
ItemPurchaseBusinessPartner.StartDetail	149
ItemPurchaseBusinessPartner.StartOverview	150
Public Interfaces for ItemsByManufacturerPartNumber	151
ItemsByManufacturerPartNumber.StartOverview	151
Public Interfaces for ItemSalesBusinessPartner	152
ItemSalesBusinessPartner.StartDetail	153
ItemSalesBusinessPartner.StartOverview	153

Public Interfaces for ItemSalesByOffice	155
ItemSalesByOffice.StartOverview	155
Public Interfaces for ItemPlanning.....	156
ItemPlanning.StartDetail	156
ItemPlanning.StartOverview	157
Public Interfaces for ItemSerial.....	158
ItemSerial.CreateOrUpdate.....	158
ItemSerial.SetStatus	159
Public Interfaces for LotAndSerialSet.....	160
LotAndSerialSet.StartOverview	160
Public Interfaces for PlanningBillOfCriticalMaterial	161
PlanningBillOfCriticalMaterial.Generate.....	161
Public Interfaces for PlanningBillOfCriticalCapacities	162
PlanningBillOfCriticalCapacities.Generate.....	162
Chapter 7 Public Interfaces for Engineering Data Management	164
Public Interfaces for EngineeringItemRevision.....	164
EngineeringItemRevision.ApproveByProduction	164
EngineeringItemRevision.CopyToItem.....	165
Chapter 8 Public Interfaces for CRM	167
Public Interfaces for Opportunity	167
Opportunity.StartMultiMain	167
Public Interfaces for Opportunities	168
Opportunities.StartOverview	168
Chapter 9 Public Interfaces for Sales	170
Public Interfaces for Sales.....	170
Sales.CalculatePlannedDeliveryDate	170
Sales.GenerateRetrobilledPriceChangeAdvices	172
Sales.GetContractSettings	174
Sales.GetGeneralSettings.....	175
Sales.GetOrderSettings.....	177
Sales.GetQuoteSettings	178
Sales.GetScheduleSettings.....	180
Sales.StartProcessProFormalInvoices	181
Public Interfaces for SalesOffice	183
SalesOffices.StartOverview.....	183

Public Interfaces for SalesQuote	184
SalesQuote.Approve	185
SalesQuote.Process	186
SalesQuote.StartMultiMain	187
SalesQuotes.StartOverview	187
Public Interfaces for SalesQuoteLine	189
SalesQuoteLine.RecalculatePriceAndDiscounts	189
SalesQuoteLine.StartDetail	190
Public Interfaces for SalesContract	191
SalesContract.GetTotalAmounts	191
SalesContract.StartMultiMain	192
SalesContract.StartWorkbench	193
SalesContracts.StartOverview	194
Public Interfaces for SalesContractLine	195
SalesContractLines.StartOverview	195
Public Interfaces for SalesSchedule	196
SalesSchedule.GetForecastQuantity	197
SalesSchedule.StartMultiMain	198
Public Interfaces for SalesScheduleLine	199
SalesScheduleLine.Reprice	199
Public Interfaces for SalesOrder	199
SalesOrder.Approve	200
SalesOrder.Block	201
SalesOrder.CalculateAmounts	202
SalesOrder.CopyBOM	204
SalesOrder.CreditCheck	205
SalesOrder.DetermineAdditionalCostLines	206
SalesOrder.DetermineDefaultPurchaseOrderData	207
SalesOrder.FinalizeBlocking	208
SalesOrder.GenerateStructure	209
SalesOrder.InitiateChangeRequest	212
SalesOrder.RecalculatePriceAndDiscounts	213
SalesOrder.ReleaseBlocking	214
SalesOrder.Rush	215
SalesOrder.StartIntakeWorkbench	215
SalesOrder.StartMultiMain	216
SalesOrder.StartPrintAcknowledgement	217

SalesOrders.StartOverview	218
Public Interfaces for SalesOrderLine	219
SalesOrderLine.Block	220
SalesOrderLine.Cancel	221
SalesOrderLine.ConfirmBackorderLine	223
SalesOrderLine.ConfirmSalesDelivery	224
SalesOrderLine.CopyFromOriginalInvoice	225
SalesOrderLine.CopyFromOriginalOrder	226
SalesOrderLine.CopyFromOriginalScheduleShipment	227
SalesOrderLine.CopyFromOriginalShipment	228
SalesOrderLine.CreateSalesDelivery	230
SalesOrderLine.ExecuteTradeComplianceCheck	231
SalesOrderLine.GenerateProductionOrder	231
SalesOrderLine.GetStatusDescription	232
SalesOrderLine.LinkContract	233
SalesOrderLine.LinkOrUnlinkInstallationGroup	235
SalesOrderLine.RecalculatePriceAndDiscounts	236
SalesOrderLine.ReleaseBlocking	237
SalesOrderLine.ReleaseManualActivity	238
SalesOrderLine.ReleaseToWarehousing	239
SalesOrderLine.Split	239
SalesOrderLine.StartAutomaticProcessing	241
SalesOrderLine.StartDetail	242
SalesOrderLines.ReleaseManualActivities.StartOverview	242
Public Interfaces for SalesOrderLineComponent	244
SalesOrderLineComponent.Block	244
SalesOrderLineComponent.ReleaseBlocking	245
Public Interfaces for SalesOrderActualDeliveryLine	245
SalesOrderActualDeliveryLine.ReadWarehousingStatusDescription	246
Public Interfaces for SalesOrderInvoiceLine	247
SalesOrderInvoiceLine.ChangePriceAndDiscounts	247
SalesOrderInvoiceLine.ReadInvoicingStatusDescription	249
SalesOrderInvoiceLine.ReleaseToInvoicing	250
Public Interfaces for SalesOrderChangeRequest	250
SalesOrderChangeRequest.Approve	251
SalesOrderChangeRequest.Process	251
Public Interfaces for SalesOrderInstallment	252

SalesOrderInstallments.Close	252
SalesOrderInstallments.StartMultiMain	253
Chapter 10 Public Interfaces for Commissions and Rebates	254
Public Interfaces for CommissionsAndRebates	254
CommissionsAndRebates.CalculateCumulativeSales	254
CommissionsAndRebates.DetermineSalesOrderData	256
CommissionsAndRebates.EvaluateAgreementGrades	258
CommissionsAndRebates.InsertBasedOnAgreementGrades	262
CommissionsAndRebates.InsertCumulativeSales	264
CommissionsAndRebates.InsertCumulativeSalesV2	266
Public Interfaces for Rebate	268
Rebates.StartOverview	269
Public Interfaces for Commission	270
Commissions.StartOverview	270
Chapter 11 Public Interfaces for Product Catalogs	273
Public Interfaces for Catalog	273
Catalog.StartOverview	273
Chapter 12 Public Interfaces for Purchase	275
Public Interfaces for Purchase	275
Purchase.GenerateRetrobilledPriceChangeAdvices	275
Purchase.GenerateRetrobilledPriceChangeAdvicesForContracts	278
Purchase.GetContractSettings	279
Purchase.GetGeneralSettings	280
Purchase.GetOrderSettings	282
Purchase.GetRequisitionSettings	283
Purchase.GetRFQSettings	285
Purchase.GetScheduleSettings	286
Public Interfaces for RequestForQuoteLine	287
RequestForQuoteLines.StartOverview	288
Public Interfaces for RequestForQuoteResponse	289
RequestForQuoteResponse.Accept	289
RequestForQuoteResponse.Reject	290
Public Interfaces for PurchaseRequisition	291
PurchaseRequisition.ApproveApprovalRecord	291
PurchaseRequisition.RejectApprovalRecord	292

PurchaseRequisition.Submit	293
Public Interfaces for PurchaseRequisitionLine.....	293
PurchaseRequisitionLine.ConvertToPurchaseOrder	293
PurchaseRequisitionLine.ConvertToPurchaseRFQ.....	295
PurchaseRequisitionLine.StartDetail	296
Public Interfaces for PurchaseContract.....	297
PurchaseContract.Activate	297
PurchaseContract.Deactivate.....	298
PurchaseContract.GetTotalAmounts	300
PurchaseContract.StartWorkbench	300
PurchaseContract.Terminate	301
Public Interfaces for PurchaseContractLine	302
PurchaseContractLine.Activate	302
PurchaseContractLine.Deactivate.....	303
PurchaseContractLine.GetNetAmountOnPriceDate	304
PurchaseContractLine.Terminate.....	305
Public Interfaces for PurchaseContractLineLogisticData	306
PurchaseContractLineLogisticData.Activate	307
PurchaseContractLineLogisticData.Deactivate.....	307
Public Interfaces for PurchaseContractPrice.....	308
PurchaseContractPrice.Activate.....	308
PurchaseContractPrice.Deactivate	309
Public Interfaces for PurchaseSchedule	310
PurchaseSchedule.GenerateReleaseForPushAndPullForecast	311
PurchaseSchedule.Regenerate	312
PurchaseSchedule.StartMultiMain	312
PurchaseSchedule.Terminate	313
Public Interfaces for PurchaseOrder	314
PurchaseOrder.InitiateChangeRequest	315
PurchaseOrder.PrintPurchaseOrder	315
PurchaseOrder.RecalculateLandedCosts.....	317
PurchaseOrder.StartAutomaticProcessing	319
PurchaseOrder.StartIntakeWorkbench	320
PurchaseOrder.StartMultiMain	321
PurchaseOrder.StartPrintReminders	321
PurchaseOrders.StartOverview.....	323
Public Interfaces for PurchaseOrderLine	324

PurchaseOrderLine.Approve	324
PurchaseOrderLine.CalculatePlannedReceiptDate	325
PurchaseOrderLine.Cancel	326
PurchaseOrderLine.ConfirmPotentialBackorder	329
PurchaseOrderLine.GetCurrentActivity	330
PurchaseOrderLine.ReleaseToWarehousing	331
PurchaseOrderLine.Split	332
PurchaseOrderLines.StartOverview	333
Public Interfaces for PurchaseOrderReceipt	334
PurchaseOrderReceipt.Confirm	334
PurchaseOrderReceipt.ConfirmV2	335
PurchaseOrderReceipt.StartDetail	337
Public Interfaces for PurchasePayableReceipt	338
PurchasePayableReceipt.ChangePriceAndDiscountAfterReceipt	338
Public Interfaces for PurchaseOrderMaterialSupplyLine	340
PurchaseOrderMaterialSupplyLines.StartOverview	341
Public Interfaces for PurchaseOrderChangeRequest	342
PurchaseOrderChangeRequest.Approve	342
PurchaseOrderChangeRequest.Process	343
Chapter 13 Public Interfaces for Pricing	344
Public Interfaces for Pricing	344
Pricing.CalculateAmounts	344
Pricing.GetGeneralSettings	348
Pricing.GetPurchaseSettings	349
Pricing.GetSalesSettings	350
Pricing.GetServiceSettings	352
Pricing.RetrievePriceBookDataForItem	353
Pricing.SimulatePurchasePrice	357
Pricing.SimulateSalesPrice	360
Pricing.SimulateServicePrice	363
Pricing.SimulateTransferPrice	369
Public Interfaces for PriceBook	370
PriceBook.StartMultiMain	371
Public Interfaces for PriceBookLine	371
PriceBookLines.StartOverview	372
Public Interfaces for SalesPriceMatrices	374

SalesPriceMatrices.StartOverview	374
Chapter 14 Public Interfaces for Landed Costs	376
Public Interfaces for LandedCosts.....	376
LandedCosts.GetSettings	376
Chapter 15 Public Interfaces for Material Pricing	378
Public Interfaces for MaterialPricing.....	378
MaterialPricing.GetPurchaseSettings	378
MaterialPricing.GetSalesSettings.....	379
Chapter 16 Public Interfaces for Planning	382
Public Interfaces for PlanItem.....	382
PlanItem.StartATPHandling	382
Public Interfaces for ItemMasterPlan	383
ItemMasterPlan.Generate	383
ItemMasterPlan.StartPlan	384
ItemMasterPlan.Update.....	385
Public Interfaces for ItemOrderPlan	386
ItemOrderPlan.Get	386
ItemOrderPlan.StartPlan	388
ItemOrderPlan.StartPlanForProject	389
ItemOrderPlan.StartPlanForProjectPeg	390
Public Interfaces for PlanItemExceptionMessageTotal.....	391
PlanItemExceptionMessageTotals.StartOverview	391
Public Interfaces for PlannedOrder	392
PlannedOrder.Replan.....	393
PlannedOrders.StartGenerate.....	393
PlannedOrders.StartOverview.....	394
PlannedOrders.TransferOrderSet	395
Public Interfaces for PlannedProductionOrder	396
PlannedProductionOrder.Transfer	397
Public Interfaces for PlannedPeggingRelations	397
PlannedPeggingRelations.FindDemandForOrder	397
PlannedPeggingRelations.FindSupplyForOrder	399
Public Interfaces for ProductAvailability	401
ProductAvailability.GetWhenAvailableSchedule.....	402

Public Interfaces for OrderPegging	403
OrderPegging.StartSelect	404
Chapter 17 Public Interfaces for Standard Costs.....	406
Public Interfaces for SimulatedPurchasePrices	406
SimulatedPurchasePrices.StartOverview	406
Public Interfaces for CalculatedItemCosts	407
CalculatedItemCosts.StartMultiMain	407
CalculatedItemCosts.StartOverview	408
Public Interfaces for StandardCosts.....	409
StandardCosts.Actualize	410
StandardCosts.Calculate.....	411
StandardCosts.CalculateAndGetStandardCost	412
StandardCosts.CalculateForNewItem	414
StandardCosts.CalculateForSimulation	414
StandardCosts.CalculateSingleItem	415
StandardCosts.PrintMultilevelCostCalculation.....	417
Chapter 18 Public Interfaces for Manufacturing Master.....	419
Public Interfaces for ProductionBillOfMaterial	419
ProductionBillOfMaterial.ApproveRevision	419
ProductionBillOfMaterial.CopyToJobShop.....	420
ProductionBillOfMaterial.ExpireRevision.....	421
ProductionBillOfMaterial.GenerateProductSubcontractingModel	422
ProductionBillOfMaterial.StartMultiMain.....	423
ProductionBillOfMaterial.ValidateRevision	423
Public Interfaces for ProductionWarehouseOrders	424
ProductionWarehouseOrders.StartDetail	424
Public Interfaces for ProductionItemBySite	425
ProductionItemBySite.Start360	426
Public Interfaces for AsBuilt.....	426
AsBuilt.StartMultiMain	427
AsBuilt.StartOverview	428
Chapter 19 Public Interfaces for Job Shop.....	430
Public Interfaces for BillOfMaterialLines.....	430
BillOfMaterialLines.StartOverview	430
Public Interfaces for ItemRoutingOperations.....	431

ItemRoutingOperations.StartOverview.....	431
Public Interfaces for JobShop.....	432
JobShop.GetMasterDataSettings.....	433
JobShop.GetOrderSettings	434
Public Interfaces for JobShopBillOfMaterial	435
JobShopBillOfMaterial.ApproveRevision	436
JobShopBillOfMaterial.Copy	437
JobShopBillOfMaterial.CreateNewRevision.....	438
JobShopBillOfMaterial.ExpireRevision.....	439
JobShopBillOfMaterial.Explode.....	439
JobShopBillOfMaterial.RemoveUseUp	441
JobShopBillOfMaterial.SetUseForCosting	442
JobShopBillOfMaterial.SetUseForPlanning	443
JobShopBillOfMaterial.StartMultiMain.....	444
JobShopBillOfMaterial.StartOverview	445
JobShopBillOfMaterial.ValidateRevision	446
Public Interfaces for JobShopRouting	447
JobShopRouting.ApproveRevision	447
JobShopRouting.ClearUseForPlanning	448
JobShopRouting.Copy.....	449
JobShopRouting.CopyOperations.....	450
JobShopRouting.CreateNewRevision	451
JobShopRouting.ExpireRevision.....	452
JobShopRouting.SetUseForCosting	453
JobShopRouting.SetUseForPlanning	453
JobShopRouting.StartMultiMain.....	454
Public Interfaces for ItemRouting	455
ItemRoutings.StartOverview.....	455
Public Interfaces for ProductVariant.....	456
ProductVariant.CreateWithOptions	456
ProductVariant.GenerateProductStructureWithoutProject.....	460
ProductVariant.GenerateProductStructureWithoutProjectV2	461
ProductVariant.GenerateStructure	461
ProductVariant.GenerateStructureV2	462
ProductVariant.RegenerateJobShopStructure.....	463
ProductVariant.RegenerateJobShopStructureV2	464
ProductVariant.StartConfigurator	465

ProductVariant.StartMultiMain.....	466
ProductVariant.StartValidateProductVariants	467
Public Interfaces for ConfigurableItemConstraintIDs	468
ConfigurableItemConstraintIDs.StartOverview	468
Public Interfaces for ConfigurableItemGenericBillOfMaterial	469
ConfigurableItemGenericBillOfMaterial.StartOverview	469
Public Interfaces for ConfigurableItemGenericPriceLists.....	470
ConfigurableItemGenericPriceLists.StartOverview	470
Public Interfaces for ConfigurableItemGenericRouting.....	471
ConfigurableItemGenericRouting.StartOverview	471
Public Interfaces for ConfigurableItemProductFeatures	472
ConfigurableItemProductFeatures.StartOverview	473
Public Interfaces for ConfigurableItemSettingsForDataGeneration	474
ConfigurableItemSettingsForDataGeneration.StartOverview	474
Public Interfaces for StandardItemConfigurationsOptions	475
StandardItemConfigurationsOptions.StartOverview	475
Public Interfaces for ProductVariantOptions	476
ProductVariantOptions.StartOverview	477
Public Interfaces for ProductionOrder	478
ProductionOrder.Cancel	479
ProductionOrder.CheckShortages	479
ProductionOrder.CreateAsBuilt	480
ProductionOrder.FinishOperationAddition	481
ProductionOrder.InitiateInventoryIssue.....	481
ProductionOrder.InitiateInventoryIssueV2	482
ProductionOrder.InitiateMaterialInventoryIssue	484
ProductionOrder.MoveRejectedMaterialToQuarantine.....	484
ProductionOrder.NewVersion.....	485
ProductionOrder.Release	486
ProductionOrder.ReportProduct.....	487
ProductionOrder.ReportProductV2	488
ProductionOrder.ReportProductWithSerials	490
ProductionOrder.ReportProductWithSerialsV2.....	492
ProductionOrder.ResetCompletedToActive	494
ProductionOrder.ScrapRejectedMaterial	494
ProductionOrder.StartClose	495
ProductionOrder.StartDetail	496

ProductionOrder.StartInventoryOverview	497
ProductionOrder.StartMultiMain	498
ProductionOrder.StartOperationAddition	499
ProductionOrder.StartPrintDocuments.....	500
ProductionOrder.StartReportOperationsCompleted	501
ProductionOrder.StartReportOrdersCompleted	502
ProductionOrder.StartResetStatus.....	503
ProductionOrder.UpdateMaterialToIssueQuantity	504
ProductionOrders.StartOverview.....	505
Public Interfaces for ProductionOrderGroup	506
ProductionOrderGroup.StartMultiMain	506
Public Interfaces for ProductionOrderOperation	507
ProductionOrderOperation.BackflushMaterialAndHours	507
ProductionOrderOperation.Block	508
ProductionOrderOperation.PlanOperation	509
ProductionOrderOperation.ReportProduct.....	510
ProductionOrderOperation.ReportProductV2	513
ProductionOrderOperation.ResetStatus	515
ProductionOrderOperation.SetRemainderStartDate.....	516
ProductionOrderOperation.SetStatusCompleted	516
ProductionOrderOperation.StartReportCompleted	518
Public Interfaces for ProductionOrderReject	518
ProductionOrderReject.MoveToQuarantine.....	519
ProductionOrderReject.Scrap	520
Public Interfaces for WorkCenter.....	521
WorkCenters.StartOverview	521
Public Interfaces for JobShopTask.....	522
JobShopTasks.StartOverview	522
Public Interfaces for UtilizationByWeek.....	523
UtilizationByWeek.StartOverview	524
Public Interfaces for SubcontractedOperation.....	525
SubcontractedOperations.StartDetail.....	525
SubcontractedOperations.StartOverview	526
Chapter 20 Public Interfaces for Subcontracting	528
Public Interfaces for SubcontractingModel.....	528
SubcontractingModel.ApproveRevision	528

SubcontractingModel.SetUseForCosting	529
SubcontractingModel.SetUseForPlanning	530
SubcontractingModel.ValidateRevision	531
Chapter 21 Public Interfaces for Repetitive Manufacturing	533
Public Interfaces for Repetitive	533
Repetitive.GetSettings	533
Chapter 22 Public Interfaces for Assembly	535
Public Interfaces for Assembly	535
Assembly.CalculatePartRequirements	535
Assembly.GetSettings	536
Assembly.UpdateSegmentSchedules	537
Public Interfaces for AssemblyOrder	538
AssemblyOrder.Create	538
AssemblyOrder.CreateV2	539
Public Interfaces for AssemblyLineStationOrder	541
AssemblyLineStationOrder.AllocateParts	541
AssemblyLineStationOrder.ReportFinished	542
AssemblyLineStationOrder.ReportFinishedV2	542
Public Interfaces for AssemblyPart	543
AssemblyPart.StartProcessReturn	544
Public Interfaces for AssemblyProductVariant	545
AssemblyProductVariant.CalculateStandardCost	545
Chapter 23 Public Interfaces for Manufacturing Tools	547
Public Interfaces for Tool	547
Tool.Request	547
Tool.Return	548
Tool.ReturnToOrder	549
Tools.GetSettings	550
Chapter 24 Public Interfaces for Manufacturing Project	552
Public Interfaces for ProjectPCS	552
ProjectPCS.CalculateCost	552
ProjectPCS.CopyCustomizedProductStructureToCustomizedStructure	553
ProjectPCS.CopyCustomizedProductStructureToStandardStructure	555
ProjectPCS.CopyProject	557

ProjectPCS.CopyStandardProductStructureToCustomizedStructure	558
ProjectPCS.Start360	560
Chapter 25 Public Interfaces for Warehousing	562
Public Interfaces for AdjustmentOrder	562
AdjustmentOrder.Process	562
AdjustmentOrder.StartDetail	563
AdjustmentOrder.StartMultiMain	563
AdjustmentOrder.StartOverview.....	564
Public Interfaces for AdjustmentOrderLine.....	565
AdjustmentOrderLine.Process	566
AdjustmentOrderLine.Submit	566
Public Interfaces for CycleCountingOrder	567
CycleCountingOrder.Generate.....	567
CycleCountingOrder.Process.....	570
CycleCountingOrder.StartDetail	570
CycleCountingOrder.StartMultiMain.....	571
CycleCountingOrder.StartOverview	572
Public Interfaces for ItemWarehouseFixedLocation.....	573
ItemWarehouseFixedLocation.StartOverview.....	573
Public Interfaces for ItemWarehouseInventoryTransaction	575
ItemWarehouseInventoryTransaction.StartDetail	575
ItemWarehouseInventoryTransaction.StartOverview	576
Public Interfaces for OrderInventoryTransaction.....	578
OrderInventoryTransaction.StartOverview	578
Public Interfaces for ItemIssueByPeriod	580
ItemIssueByPeriod.StartDetail	580
ItemIssueByPeriod.StartOverview	581
Public Interfaces for PlannedInventoryTransaction.....	582
PlannedInventoryTransaction.StartDetail.....	582
PlannedInventoryTransaction.StartOverview	583
Public Interfaces for Inventory	585
Inventory.CalculateTimePhasedAvailability	586
Inventory.ChangeMAUC.....	587
Inventory.ChangeMAUCCostDetails	589
Inventory.DetermineCompanyOwnedValueDependingOnValuationBasis	591
Inventory.DetermineQuantity.....	594

Inventory.DetermineQuantityAndValue	596
Inventory.DetermineQuantityAndValueV2	598
Inventory.GetAnonymousInventoryQuantities	601
Inventory.GetTimePhasedAvailability	602
Inventory.GetTimePhasedAvailabilityV2	602
Inventory.Start360	603
Public Interfaces for SpecificationInventory	605
SpecificationInventory.GetUnallocatedAvailableQuantity	605
Public Interfaces for InventoryCommitment	605
InventoryCommitment.AllowedForOrderLine	606
InventoryCommitment.CalculateCommitmentDate	607
InventoryCommitment.CommitInventoryToOrderLine	607
InventoryCommitment.ConsumeFromBuffer	609
InventoryCommitment.CreateAllocationBuffer	610
InventoryCommitment.Generate	611
InventoryCommitment.GetAvailableQuantityToCommit	613
Public Interfaces for SiteItemInventory	614
SiteItemInventory.StartDetail	615
SiteItemInventory.StartOverview	615
Public Interfaces for WarehouseItemInventory	616
WarehouseItemInventory.StartDetail	617
WarehouseItemInventory.StartMultiMain	618
WarehouseItemInventory.StartOverview	618
Public Interfaces for WarehouseReceipt	620
WarehouseReceipt.PrintReceipts	620
WarehouseReceipt.StartDetail	622
WarehouseReceipt.StartMultiMain	623
WarehouseReceipt.StartOverview	624
WarehouseReceipts.StartWorkbench	625
Public Interfaces for Warehousing	625
Warehousing.ConvertKindOfOrderToOrderOrigin	626
Warehousing.ConvertOrderOriginToKindOfOrder	626
Warehousing.GetSettings	627
Warehousing.StartAutomaticInboundProcessing	629
Warehousing.StartAutomaticOutboundProcessing	630
Public Interfaces for StockPoint	630
StockPoint.CheckBlocking	631

StockPoint.Move.....	632
StockPoint.PackHandlingUnits.....	634
Public Interfaces for StockPointInventory.....	635
StockPointInventory.StartDetail.....	635
StockPointInventory.StartOverview.....	636
Public Interfaces for Warehouse.....	638
Warehouse.StartDetail.....	638
Warehouse.StartOverview.....	639
Public Interfaces for WarehouseLocation.....	640
WarehouseLocation.StartDetail.....	640
WarehouseLocation.StartOverview.....	641
WarehouseLocation.StartRecalculateOccupation.....	643
Public Interfaces for CompanyItemInventory.....	644
CompanyItemInventory.StartDetail.....	644
CompanyItemInventory.StartOverview.....	645
Public Interfaces for ShipmentNotice.....	646
ShipmentNotice.ReceiveDirectDelivery.....	646
ShipmentNotice.StartDetail.....	647
ShipmentNotice.StartMultiMain.....	648
ShipmentNotice.StartOverview.....	648
Public Interfaces for ShipmentNoticeLine.....	650
ShipmentNoticeLine.Receive.....	650
Public Interfaces for WarehouseOrder.....	651
WarehouseOrder.CalculateDeliveryDate.....	652
WarehouseOrder.CreateReturn.....	653
WarehouseOrder.PrintLabelForActivity.....	655
WarehouseOrder.StartDetail.....	657
WarehouseOrder.StartMultiMain.....	658
WarehouseOrder.StartOverview.....	659
Public Interfaces for PurchaseOrderAdvice.....	661
PurchaseOrderAdvice.Transfer.....	661
Public Interfaces for ProductionOrderAdvice.....	663
ProductionOrderAdvice.StartDetail.....	663
ProductionOrderAdvice.StartOverview.....	664
Public Interfaces for PackageDefinition.....	665
PackageDefinition.DetermineMultipleOfQuantity.....	665

PackageDefinition.Validate	667
Public Interfaces for HandlingUnit	668
HandlingUnit.ApproveRemaining	669
HandlingUnit.CalculateWeightAndDimensions	669
HandlingUnit.Close	670
HandlingUnit.ConfirmReceipt	671
HandlingUnit.Create	671
HandlingUnit.GenerateEmptyHandlingUnits	672
HandlingUnit.GenerateLots	673
HandlingUnit.GenerateSerials	674
HandlingUnit.Link	676
HandlingUnit.MoveToLoad	676
HandlingUnit.MoveToShipment	677
HandlingUnit.PrintLabelByPackageLabelBOD	678
HandlingUnit.PrintLabels	679
HandlingUnit.Receive	682
HandlingUnit.RejectRemaining	683
HandlingUnit.RepackInInventory	683
HandlingUnit.SetToNotShipped	684
HandlingUnit.SetToShipped	685
HandlingUnit.StartAutomaticInboundProcessing	686
HandlingUnit.StartDetail	687
HandlingUnit.StartOverview	688
HandlingUnit.StartPrintLabels	689
HandlingUnit.Transfer	690
HandlingUnit.Unlink	691
HandlingUnit.Unpack	692
Public Interfaces for HandlingUnitStructure	694
HandlingUnitStructure.GetBottomLevel	694
HandlingUnitStructure.GetComponentLevel	695
HandlingUnitStructure.GetInventoryLevel	696
HandlingUnitStructure.GetOrderLineLevel	697
HandlingUnitStructure.GetWholeTree	697
Public Interfaces for InboundOrderLineActivity	698
InboundOrderLineActivity.ModifyAutomaticProcessing	698
Public Interfaces for InboundAdvice	700
InboundAdvice.GetAvailableLocation	700
InboundAdvice.StartAutomaticInboundProcessing	703

Public Interfaces for InboundInspection	704
InboundInspection.StartAutomaticInboundProcessing	705
Public Interfaces for InboundOrderLine.....	706
InboundOrderLine.PrintGoodsReceivedNote	707
InboundOrderLine.Receive	708
InboundOrderLine.StartAutomaticInboundProcessing	710
InboundOrderLine.StartPrintGoodsReceivedNote	712
Public Interfaces for InboundRun	713
InboundRun.GenerateStorageList.....	713
InboundRun.StartAutomaticInboundProcessing	714
Public Interfaces for ProjectCostPegTransfer	715
ProjectCostPegTransfer.CheckAllowedBetweenPegs	716
ProjectCostPegTransfer.CheckInventoryAvailable	717
ProjectCostPegTransfer.Generate	717
ProjectCostPegTransfer.StartDetail	719
ProjectCostPegTransfer.StartMultiMain	720
ProjectCostPegTransfer.StartOverview	721
Public Interfaces for ProjectCostPegTransferLine	722
ProjectCostPegTransferLine.Process	722
ProjectCostPegTransferLine.StartDetail	723
ProjectCostPegTransferLine.StartOverview	723
Public Interfaces for AllocationChangeOrder	726
AllocationChangeOrder.Generate	726
AllocationChangeOrder.Process	729
Public Interfaces for Lot.....	730
Lot.Generate	730
Lot.StartDetail	731
Lot.StartGlobalLotBlocking	732
Lot.StartOverview	733
Lot.StartPrintLotBlocking	735
Public Interfaces for LotInventoryByWarehouseAndItem	736
LotInventoryByWarehouseAndItem.StartDetail	736
LotInventoryByWarehouseAndItem.StartMultiMain	737
LotInventoryByWarehouseAndItem.StartOverview	738
Public Interfaces for SerialsByWarehouse	739
SerialsByWarehouse.StartDetail	739
SerialsByWarehouse.StartOverview	740

Public Interfaces for ItemLotSerialTransaction.....	743
ItemLotSerialTransaction.GetData	743
Public Interfaces for SerializedItem	744
SerializedItem.Convert	744
Public Interfaces for LotItem	746
Public Interfaces for Load	746
Load.Freeze	747
Load.Reopen	747
Load.StartAutomaticOutboundProcessing	748
Load.StartDetail	749
Load.StartMultiMain	750
Load.StartOverview	751
Public Interfaces for DeliveryNote	752
DeliveryNote.Print	753
Public Interfaces for Shipment.....	755
Shipment.Confirm	756
Shipment.Freeze	757
Shipment.MoveToLoad	757
Shipment.NotifyAutomaticOutboundProcess	758
Shipment.PrintLabelsByPackageLabelBOD	759
Shipment.PrintPackingList	759
Shipment.PrintPackingSlip	761
Shipment.PrintPickAndLoadSheet	762
Shipment.ProcessInvoice	764
Shipment.Reopen	766
Shipment.StartAutomaticOutboundProcessing	767
Shipment.StartDetail	768
Shipment.StartMultiMain	769
Shipment.StartOverview	769
Public Interfaces for ShipmentLine.....	771
ShipmentLine.Confirm	772
ShipmentLine.ConfirmIsAllowed	772
ShipmentLine.MoveToLoad	773
ShipmentLine.MoveToShipment	774
ShipmentLine.PrintLabelsByPackageLabelBOD	774
ShipmentLine.Split.....	775
ShipmentLine.StartDetail.....	777

ShipmentLine.StartOverview	778
Public Interfaces for ShipmentLineStockPointDetail	780
ShipmentLineStockPointDetails.StartOverview	780
Public Interfaces for ProjectedShipment	781
ProjectedShipment.Generate	782
ProjectedShipment.GenerateOutboundAdvice	784
ProjectedShipment.ReleaseOutboundAdvice	785
ProjectedShipment.StartGenerate	786
Public Interfaces for OutboundAdvice	787
OutboundAdvice.StartAutomaticOutboundProcessing	788
OutboundAdvice.StartDetail	789
OutboundAdvice.StartGenerate	790
OutboundAdvice.StartOverview	792
Public Interfaces for OutboundRun	794
OutboundRun.HandlePickingList	794
OutboundRun.StartAutomaticOutboundProcessing	797
Public Interfaces for QuarantineInventory	798
QuarantineInventory.Move	799
QuarantineInventory.StartAutomaticInboundProcessing	799
QuarantineInventory.StartDetail	801
QuarantineInventory.StartDisposition	802
QuarantineInventory.StartOverview	804
Public Interfaces for QuarantineInventoryDisposition	805
QuarantineInventoryDisposition.StartDetail	805
QuarantineInventoryDisposition.StartOverview	806
Public Interfaces for QuarantineInventoryDispositionLine	808
QuarantineInventoryDispositionLine.GenerateInboundAdvice	809
QuarantineInventoryDispositionLine.Move	809
QuarantineInventoryDispositionLine.Process	810
Public Interfaces for QuarantineInventoryHandlingUnit	811
QuarantineInventoryHandlingUnit.Move	811
QuarantineInventoryHandlingUnit.Process	812
Public Interfaces for ProjectPeggedInventory	813
ProjectPeggedInventory.StartDetail	813
ProjectPeggedInventory.StartOverview	814
Public Interfaces for OutboundOrderLineActivity	816

OutboundOrderLineActivity.Create	817
OutboundOrderLineActivity.ModifyAutomaticProcessing	817
Public Interfaces for OutboundOrderLine	819
OutboundOrderLine.StartAutomaticOutboundProcessing	819
Public Interfaces for ReceiptLine	820
ReceiptLine.Confirm	821
ReceiptLine.CorrectFinalReceipt	822
ReceiptLine.GenerateHandlingUnits	822
ReceiptLine.GenerateHandlingUnitsV2	823
ReceiptLine.GenerateLots	824
ReceiptLine.GenerateSerials	825
ReceiptLine.GetLanguage	826
ReceiptLine.Reverse	827
ReceiptLine.Split	828
ReceiptLine.SplitForLotsBatches	829
ReceiptLine.SplitForSerials	830
ReceiptLine.StartAutomaticInboundProcessing	832
Public Interfaces for ReceiptLineStockPointDetail	833
ReceiptLineStockPointDetails.StartOverview	834
Public Interfaces for StorageList	835
StorageList.SplitAdvice	835
StorageList.StartAutomaticInboundProcessing	836
Public Interfaces for PickingList	837
PickingList.SplitAdvice	837
Public Interfaces for WarehouseInspection	838
WarehouseInspection.CopyImage	838
WarehouseInspection.LinkHandlingUnit	839
WarehouseInspection.Print	840
WarehouseInspection.Process	842
Public Interfaces for WarehouseInspectionLine	843
WarehouseInspectionLine.Inspect	843
WarehouseInspectionLine.StartDetail	844
WarehouseInspectionLine.StartOverview	845
Public Interfaces for InventoryInspection	847
InventoryInspection.Generate	847
Public Interfaces for HandlingUnitStockPointDetail	849
HandlingUnitStockPointDetail.Inspect	849

HandlingUnitStockPointDetail.SplitForSerial	851
Public Interfaces for NegativeInventory.....	851
NegativeInventory.Consume	852
Public Interfaces for Weight.....	852
Weight.Convert.....	853
Public Interfaces for UnexpectedReceipt	853
UnexpectedReceipt.Create	853
Public Interfaces for WarehouseOrderLine	855
WarehouseOrderLine.Activate	856
WarehouseOrderLine.ActivityIsApplicable	856
WarehouseOrderLine.ActivityIsMandatory.....	858
WarehouseOrderLine.UpdatePlannedDates.....	859
Public Interfaces for Kanban	859
Kanban.GenerateOrders	860
Kanban.StartGenerate.....	861
Public Interfaces for TimePhasedOrderPoint.....	861
TimePhasedOrderPoint.GenerateOrders.....	862
Public Interfaces for ProductionOrder	863
ProductionOrder.GenerateHandlingUnitForLineSideLabeling	863
ProductionOrder.GenerateHandlingUnitForLineSideLabelingV2	864
Public Interfaces for CrossDockOrderLine	866
CrossDockOrderLines.Print.....	866
Public Interfaces for Run	868
Run.DetermineNumber.....	868
Chapter 26 Public Interfaces for Freight	870
Public Interfaces for FreightOrderCluster.....	870
FreightOrderCluster.ConfirmDelivery	870
Public Interfaces for FreightPlanning	871
FreightPlanning.ActualizePlan	871
FreightPlanning.GenerateForSpecificOrderSelection.....	872
FreightPlanning.MoveShipmentToLoad.....	874
Public Interfaces for Load	875
Load.ConfirmDelivery	875
Public Interfaces for Shipment.....	876
Shipment.ConfirmDelivery	876

Chapter 27 Public Interfaces for Service	878
Public Interfaces for Service	878
Service.DetermineStartOrFinishTime	878
Service.GetClaimSettings	880
Service.GetGeneralSettings	881
Service.GetMaintenanceSalesOrderSettings	883
Service.GetMobileServiceSettings	884
Service.GetPlanningSettings	886
Service.GetQuoteSettings	887
Service.GetResourceCalendarCode	888
Service.GetServiceOrderSettings	890
Service.GetWorkOrderSettings	891
Public Interfaces for SerializedItem	893
SerializedItem.ChangeStatus	893
SerializedItem.GetActiveContracts	894
SerializedItem.GetSupplierWarrantyEndDate	895
SerializedItem.GetWarrantyEndDate	896
SerializedItem.StartMultiMain	897
SerializedItem.StartOverview	898
SerializedItem.Supersede	898
Public Interfaces for PhysicalBreakdown	901
PhysicalBreakdown.Change	901
PhysicalBreakdown.CreateFromProjectBreakdownStructure	904
PhysicalBreakdown.CreateStructure	905
PhysicalBreakdown.ItemExistsInBreakdownOfParent	909
Public Interfaces for ServiceInspections	910
ServiceInspections.StartOverview	910
Public Interfaces for Call	912
Call.TransferToCustomerClaim	912
Call.TransferToPlannedActivity	913
Call.TransferToServiceOrder	914
Call.TransferToServiceOrderV2	916
Call.TransferToServiceQuote	918
Call.TransferToWorkOrder	920
Public Interfaces for ServiceContract	922
ServiceContract.Activate	922
ServiceContract.ActivateV2	924

ServiceContract.Create	926
ServiceContract.GenerateConfigurationLines	929
ServiceContract.GetTotalSalesAmount	931
ServiceContract.StartMultiMain	931
ServiceContract.UnpackObjectReference	932
Public Interfaces for ServiceContractChange	933
ServiceContractChange.Create	933
Public Interfaces for ServiceContractConfigurationLine.....	935
ServiceContractConfigurationLine.Create.....	935
ServiceContractConfigurationLine.GetContractCoverageDetails	938
Public Interfaces for ServiceContractInstallment	940
ServiceContractInstallment.Cancel	940
ServiceContractInstallment.SwitchStatus	941
ServiceContractInstallment.Transfer	942
Public Interfaces for ServiceOrder.....	943
ServiceOrder.Cancel	943
ServiceOrder.CheckAllMaterialLinesHaveBeenReceivedAndIssued	945
ServiceOrder.Close	947
ServiceOrder.Complete	948
ServiceOrder.Cost	949
ServiceOrder.GenerateActivitiesFromMasterRouting.....	952
ServiceOrder.GetTotalSalesAndCostAmounts	954
ServiceOrder.IssueInventory	956
ServiceOrder.Plan	957
ServiceOrder.ReadShippingAddressForCostLine	962
ServiceOrder.Release	963
ServiceOrder.StartMultiMain	968
Public Interfaces for ServiceOrderActivity	969
ServiceOrderActivity.Cancel.....	969
ServiceOrderActivity.Complete	972
ServiceOrderActivity.Cost	974
ServiceOrderActivity.Plan.....	977
ServiceOrderActivity.ProcessReturnDeliveries	980
ServiceOrderActivity.Release.....	981
ServiceOrderActivity.StartMultiMain.....	985
Public Interfaces for ServiceOrderMaterialCosts	986
ServiceOrderMaterialCosts.ProcessReturnDeliveries	986

Public Interfaces for ServiceOrderCostLine	987
ServiceOrderCostLine.Cost.....	987
Public Interfaces for ServiceQuote	991
ServiceQuote.Process.....	991
Public Interfaces for MaintenanceSalesOrder.....	993
MaintenanceSalesOrder.Cost	993
MaintenanceSalesOrder.GenerateSerializedItem	994
Public Interfaces for MaintenanceSalesOrderLine	995
MaintenanceSalesOrderLine.Cost	996
MaintenanceSalesOrderLine.ItemReceipt	997
MaintenanceSalesOrderLine.Release	998
Public Interfaces for WorkOrder	999
WorkOrder.Cancel.....	1000
WorkOrder.Close.....	1001
WorkOrder.Complete.....	1003
WorkOrder.DetermineStartOrFinishTime.....	1005
WorkOrder.GenerateActivitiesFromMasterRouting	1007
WorkOrder.GetActualRepairCosts	1008
WorkOrder.Release.....	1009
WorkOrder.StartDetail	1011
WorkOrder.StartMultiMain.....	1012
WorkOrder.StartOverview	1013
Public Interfaces for WorkOrderActivity.....	1015
WorkOrderActivity.ProcessReturnDeliveries	1015
WorkOrderActivity.SignOff	1016
Public Interfaces for WorkOrderMaterialCosts.....	1018
WorkOrderMaterialCosts.ProcessReturnDeliveries.....	1018
Public Interfaces for PlannedActivity	1019
PlannedActivity.CreateMaintenancePlan	1019
PlannedActivity.SwitchStatus	1024
PlannedActivity.TransferToFieldService	1027
Public Interfaces for CustomerClaim	1031
CustomerClaim.Close.....	1031
CustomerClaim.StartMultiMain.....	1032
CustomerClaim.Submit.....	1033
Public Interfaces for CustomerClaimLine	1034
CustomerClaimLine.Approve	1034

CustomerClaimLine.ApproveForInvoiceLinking	1036
CustomerClaimLine.CalculateClaimAmount	1036
CustomerClaimLine.Reject	1038
Public Interfaces for SupplierClaim	1040
SupplierClaim.SetRMAReceived	1040
Public Interfaces for SupplierClaimLine	1041
SupplierClaimLine.Approve	1041
SupplierClaimLine.Reject	1042
SupplierClaimLine.Settle	1043
Chapter 28 Public Interfaces for Invoicing.....	1044
Public Interfaces for Invoice.....	1044
Invoice.ComposePrintPostInvoices.....	1044
Invoice.CreditInvoiceLine	1046
Invoice.GetInterimRevenueLedgerAccountAndDimensions	1047
Invoice.StartMultiMain	1048
Public Interfaces for ProformaInvoice	1049
ProformaInvoice.CancelInvoice.....	1049
ProformaInvoice.StartWorkbench	1050
Public Interfaces for Installment	1051
Installment.ApproveCorrections	1051
Installment.CreateCorrections	1052
Installment.TransferCorrections	1053
Public Interfaces for InvoicingBatch	1053
InvoicingBatch.CreateInvoice	1054
Public Interfaces for BillableLine	1055
BillableLine.CreateInvoice	1055
BillableLine.UpdateStatus	1057
Public Interfaces for ManualSalesInvoice	1058
ManualSalesInvoice.ConfirmInvoices	1058
Chapter 29 Public Interfaces for Taxation.....	1060
Public Interfaces for Tax	1060
Tax.Calculate	1060
Tax.GetBusinessPartnerTaxRegistrationID	1065
Tax.GetDefaultsGoods	1065
Tax.GetRate	1069

Tax.ProcessTestScenario	1069
Chapter 30 Public Interfaces for BOD & BDE	1071
Public Interfaces for BOD	1071
BOD.ActionsAfterProcessingIncomingRequest	1072
BOD.ActionsAfterProcessingIncomingRequestWithAutomaticProcessing	1073
BOD.ActionsBeforeProcessingIncomingRequest	1074
BOD.ComposeLocationId	1074
BOD.ConvertFromERPDateToISODate	1075
BOD.ConvertFromERPItem	1076
BOD.ConvertFromISODateToERP	1077
BOD.ConvertFromISOUnitToERP	1078
BOD.ConvertToERPItem	1079
BOD.CreateCodeDefinitionReferenceNode	1080
BOD.CreateIDNode	1080
BOD.CreateLastModificationPersonNode	1081
BOD.CreateNoteNode	1082
BOD.ExecuteMethod	1083
BOD.ExecuteOnAcknowledgeForChameleon	1084
BOD.ExecuteOnLoadForChameleon	1084
BOD.ExecuteOnProcessForChameleon	1085
BOD.ExecuteOnSyncForChameleon	1086
BOD.ExecuteOnUpdateForChameleon	1087
BOD.ExecutePublishForChameleon	1087
BOD.ExecuteRequest	1088
BOD.ExecuteShowForChameleon	1089
BOD.GetIdAccountingEntity	1090
BOD.GetIdLocation	1090
BOD.GetIdLogicalID	1091
BOD.GetPublishingAllowed	1092
BOD.GetTextNumberFromNoteNode	1093
BOD.GetValueFromNodeElement	1094
BOD.HandleStagingAfterProcessingIncomingRequest	1094
BOD.HandleStagingBeforeProcessingIncomingRequest	1095
BOD.InterruptPublishing	1096
BOD.Publish	1097
BOD.PublishLNMessage	1097
BOD.PublishWithBatch	1099
BOD.ResumePublishing	1101

Public Interfaces for BDE.....	1101
BDE.ExecuteMethod	1102
Chapter 31 Public Interfaces for Factory Track.....	1103
Public Interfaces for FactoryTrackBDE	1103
FactoryTrackBDE.CreateElement.....	1103
FactoryTrackBDE.CreateNode.....	1104
Public Interfaces for FactoryTrackQuery	1105
FactoryTrackQuery.CreateElement	1106
FactoryTrackQuery.CreateRow.....	1107
FactoryTrackQuery.GetElement.....	1108
Chapter 32 Public Interfaces for Project	1110
Public Interfaces for Project.....	1110
Project.Activate.....	1110
Project.Start360.....	1111
Project.StartCopy	1112
Public Interfaces for ProjectContract.....	1114
ProjectContract.StartMultiMain.....	1114
Public Interfaces for ProjectBid	1115
ProjectBid.StartMultiMain	1115
Public Interfaces for ProjectEstimate.....	1116
ProjectEstimate.GenerateStructuralElements	1116
Public Interfaces for ProjectContractDeliverable.....	1118
ProjectContractDeliverable.AddCPQConfiguration.....	1118
Public Interfaces for ProjectActivityBudget.....	1120
ProjectActivityBudget.AddCPQConfiguration	1120
Public Interfaces for ProjectActivity	1121
ProjectActivity.SetInsertMode	1122
Chapter 33 Public Interfaces for Authorization and Security	1124
Public Interfaces for DataAuthorization	1124
DataAuthorization.DocumentActionAllowed.....	1124
DataAuthorization.DocumentFilter	1128
Chapter 34 Public Interfaces for Quality Management.....	1133
Public Interfaces for OrderInspection	1133
OrderInspection.Close.....	1133

OrderInspection.Complete	1134
OrderInspection.Process	1135
Public Interfaces for InspectionOrder	1136
InspectionOrder.Print.....	1136
Public Interfaces for CorrectiveActionPlan	1137
CorrectiveActionPlan.SetStatus	1138
CorrectiveActionPlan.StartMultiMain.....	1138
Public Interfaces for CorrectiveActionPlanTask	1139
CorrectiveActionPlanTask.SetStatus	1139
Public Interfaces for FRACASHeader	1140
FRACASHeader.StartMultiMain	1140
Public Interfaces for TestingCombination.....	1141
TestingCombinations.StartOverview	1141
Public Interfaces for StandardTestProcedure	1143
StandardTestProcedure.StartMultiMain	1143
Public Interfaces for NonConformanceReport.....	1144
NonConformanceReport.StartDetail	1144
Chapter 35 Public Interfaces for Cash Management	1146
Public Interfaces for PaymentAdvice.....	1146
PaymentAdvice.StartProcessPayments.....	1146
Chapter 36 Public Interfaces for Accounts Receivable	1148
Public Interfaces for AccountsReceivable	1148
AccountsReceivable.DetermineDueDate	1148
AccountsReceivable.StartOverview	1149
Chapter 37 Public Interfaces for General Ledger	1151
Public Interfaces for GeneralLedger.....	1151
GeneralLedger.CheckBatchLineIsBalanced	1151
GeneralLedger.CheckDocumentIsBalanced	1152
Public Interfaces for Account.....	1153
Account.CrossValidateDimensions	1153
Chapter 38 Public Interfaces for Fixed Assets	1154
Public Interfaces for Asset.....	1154
Asset.GetDefaultLedgerAccountAndDimensions	1154
Asset.StartAdjustAssets	1155

Asset.StartTransferAssets.....	1157
Public Interfaces for AssetBook.....	1159
AssetBook.CalculateRemainingDepreciationLife.....	1160
Chapter 39 Public Interfaces for Product Lifecycle Management.....	1161
Public Interfaces for PLMObject.....	1161
PLMObject.CheckIn.....	1161
PLMObject.CheckOut.....	1162
PLMObject.UndoCheckIn.....	1163
PLMObject.UndoCheckOut.....	1164
Chapter 40 Public Interfaces for Object Configuration Management.....	1165
Public Interfaces for Workflow.....	1165
Workflow.DisableSubmit.....	1165
Workflow.EnableSubmit.....	1167
Workflow.Recall.....	1167
Workflow.RecallByTypedObjectID.....	1168
Workflow.Submit.....	1169
Chapter 41 Public Interfaces for Time Management.....	1170
Public Interfaces for ProductionOrderHours.....	1170
ProductionOrderHours.CloseActiveLine.....	1170
ProductionOrderHours.InsertActiveLine.....	1171
ProductionOrderHours.StartOverview.....	1172
Public Interfaces for GeneralHours.....	1173
GeneralHours.StartOverview.....	1173
Chapter 42 Public Interfaces for ZWF AG Shipment and Customs.....	1176
Public Interfaces for SanctionList.....	1176
SanctionList.Check.....	1176
Public Interfaces for ExportControl.....	1177
ExportControl.Check.....	1177
Chapter 43 Process Extensions.....	1179
Process Extensions for AccountsReceivable.....	1179
AccountsReceivable.GetEffectiveDateForAging.....	1179
tfext.acr0001.get.effective.date.for.aging.analysis.....	1179
Process Extensions for AssemblyOrder.....	1180
AssemblyOrder.SkipDelete.....	1180

AssemblyOrder.SkipGenerate.....	1181
Process Extensions for AssemblyPart	1182
AssemblyPart.SkipBackflush.....	1182
Process Extensions for BillableLine	1183
BillableLine.CheckConfirmedAllowed	1183
ciext.sli0002.check.billable.line.status.confirmed.allowed	1183
BillableLine.SkipComposing	1184
Process Extensions for BillOfLading	1185
BillOfLading.SkipShipmentLine	1185
Process Extensions for BlockedSalesOrderOrSalesOrderLine	1186
BlockedSalesOrderOrSalesOrderLine.SkipFirmRelease	1186
BlockedSalesOrderOrSalesOrderLine.SkipSoftRelease	1187
Process Extensions for BOD	1187
BOD.PublishCustomBOD.....	1188
ttext.bod0001.get.custom.bod.properties	1189
ttext.bod0001.get.custom.bod.publish.function	1190
ttext.bod0001.get.custom.bod.names	1192
ttext.bod0001.get.initial.load.query.....	1193
Process Extensions for CommissionsRebates.....	1194
CommissionsRebates.DetermineTaxCountry	1194
tdext.cms0001.commission.rebates.determine.tax.country	1195
CommissionsRebates.SkipCalculate	1196
Process Extensions for Common	1197
Common.DeterminePurchaseType	1198
ttext.mcs0001.get.customer.defined.purchase.type.....	1198
Process Extensions for ComposeShippingStructure	1199
ComposeShippingStructure.CommandHandling	1199
whext.dll0007.compose.shipping.structure.command.is.allowed	1200
whext.dll0007.compose.shipping.structure.on.execute.command	1201
Process Extensions for ConfigurableItem	1202
ConfigurableItem.SkipCompileConstraints	1202
Process Extensions for CycleCountingData.....	1202
CycleCountingData.SkipDelete	1203
Process Extensions for DeliveryNote	1203
DeliveryNote.SkipPrint.....	1204
Process Extensions for EffectivityUnit.....	1204

EffectivityUnit.CheckInterchangeable	1204
tcext.uef0001.eff.units.are.interchgng	1204
Process Extensions for ExceptionMessagesByItem	1206
ExceptionMessagesByItem.SkipPrint	1206
Process Extensions for FactoryTrackQuery	1207
FactoryTrackQuery.QueryExtend	1207
brext.qry0001.custom.query	1207
brext.qry0001.standard.query.extension	1210
Process Extensions for FreightLoadBuilding	1218
FreightLoadBuilding.CustomBuildingConstraints	1218
fmext.dli0002.custom.load.building.constraints	1218
Process Extensions for FreightShipmentBuilding	1219
FreightShipmentBuilding.CustomBuildingConstraints	1220
fmext.dli0001.custom.shipment.building.constraints	1220
Process Extensions for GenerateFieldChangeOrderLine	1221
GenerateFieldChangeOrderLine.SkipSerializedItem	1221
Process Extensions for GeneratePlan	1222
GeneratePlan.SkipOrderLine	1222
Process Extensions for HandlingUnitBuilding	1223
HandlingUnitBuilding.CustomShipmentLine	1223
whext.wmd0001.custom.shipment.line	1223
Process Extensions for InboundAdvice	1224
InboundAdvice.SkipPutawayAdvice	1224
Process Extensions for IntegrationTransaction	1225
IntegrationTransaction.SkipPrint	1225
Process Extensions for IntrastatTransaction	1225
IntrastatTransaction.CheckBlockedForReporting	1226
lpext.ita0001.check.intrastat.transaction.blocked.for.reporting	1226
Process Extensions for InventoryAgingAnalysis	1227
InventoryAgingAnalysis.Custom	1227
whext.dli0011.customize.inventory.aging.analysis	1228
InventoryAgingAnalysis.Skip	1228
Process Extensions for InventorySearchEngine	1229
InventorySearchEngine.CheckHandlingUnit	1229
whext.dli0002.check.handling.unit	1230
InventorySearchEngine.CheckStockPoint	1231

whext.dlI0001.check.stock.point.....	1232
whext.dlI0001.stock.point.selection	1233
Process Extensions for Invoice	1233
Invoice.CombineAdvancesIsAllowed	1234
ciext.sli0004.combine.advances.is.allowed.....	1234
Invoice.CustomComposingCriteriaMet.....	1235
ciext.sli0003.invoice.custom.composing.criteria.met	1235
Invoice.CustomXMLInvoicing	1237
ciext.sli0001.define.custom.xml.data.elements.....	1238
ciext.sli0001.get.value.for.custom.xml.data.element	1240
Invoice.SkipCreditNoteLinking	1242
Invoice.SkipPrintInvoice	1243
Process Extensions for InvoiceFromBP	1244
InvoiceFromBP.SkipPrintAgingAnalysis.....	1245
InvoiceFromBP.SkipPrintOpenEntries	1245
InvoiceFromBP.SkipWriteOffPaymentDifference.....	1246
Process Extensions for InvoiceToBP	1246
InvoiceToBP.SkipPrintAgingAnalysis.....	1247
InvoiceToBP.SkipPrintOpenEntries	1247
Process Extensions for ItemMasterPlan	1248
ItemMasterPlan.SkipCopyToScenario	1248
ItemMasterPlan.SkipCopyToScenarioForPeriod	1249
Process Extensions for LoadBuilding.....	1249
LoadBuilding.CustomBuildingConstraints	1249
whext.dlI0004.custom.load.building.constraints	1250
Process Extensions for LocationSearchEngine	1251
LocationSearchEngine.CheckLocation	1251
whext.dlI0015.override.location	1252
whext.dlI0015.skip.location.....	1254
Process Extensions for OutboundAdvice	1255
OutboundAdvice.SkipGeneratePickingList	1255
OutboundAdvice.SkipPrint	1256
OutboundAdvice.SkipReleaseAdvice.....	1256
Process Extensions for OutboundOrderLine.....	1256
OutboundOrderLine.SkipGenerateAdvice.....	1257
Process Extensions for PackingList	1257
PackingList.SkipPrint.....	1257

Process Extensions for PackingSlip	1258
PackingSlip.SkipPrint	1258
Process Extensions for PaymentAdvice	1258
PaymentAdvice.CustomAuditPayments	1258
tfext.cmg0002.custom.audit.composed.payment	1259
tfext.cmg0002.custom.audit.payment.advice.line	1259
Process Extensions for PaymentReceipt	1260
PaymentReceipt.CustomXMLHandling	1260
tfext.cmg0001.define.custom.payment.elements	1260
tfext.cmg0001.define.custom.receipt.elements	1262
tfext.cmg0001.get.value.for.custom.payment.element	1263
tfext.cmg0001.get.value.for.custom.receipt.element	1265
Process Extensions for PerformInventoryValuation	1266
PerformInventoryValuation.SkipItem	1266
Process Extensions for PickingMission	1267
PickingMission.HandleAfterGenerate	1267
whext.dll0005.picking.mission.handle.after.generate	1267
Process Extensions for PlannedInventoryTransaction	1268
PlannedInventoryTransaction.SkipPrint	1268
Process Extensions for PlannedOrder	1269
PlannedOrder.SkipConfirmOrder	1269
PlannedOrder.SkipPrint	1269
PlannedOrder.SkipTransfer	1270
Process Extensions for PlannedServiceActivity	1271
PlannedServiceActivity.SkipSwitchStatus	1271
Process Extensions for Pricing	1272
Pricing.DefaultPriceDateType	1272
tdext.pcg0002.get.customer.determined.purchase.price.date.type.parameter	1273
tdext.pcg0002.get.customer.determined.purchase.price.date.type.setting	1273
tdext.pcg0002.get.customer.determined.sales.price.date.type.parameter	1274
tdext.pcg0002.get.customer.determined.sales.price.date.type.setting	1275
tdext.pcg0002.get.customer.determined.service.price.date.type.parameter	1275
tdext.pcg0002.get.customer.determined.service.price.date.type.setting	1276
Pricing.RetrievePriceAndDiscounts	1277
tdext.pcg0001.get.customer.determined.price.and.discounts	1277
Process Extensions for Procurement	1282
Procurement.DefaultPurchaseOffice	1282

tdext.pur0001.get.customer.determined.default.purchase.office	1283
Procurement.HandleGeneralLedgerCodeDefaulting	1284
tdext.pur0005.skip.general.ledger.code.defaulting	1284
Process Extensions for ProductionOrder	1285
ProductionOrder.PrintCustomReports	1286
tiext.sfc0001.check.print.condition	1286
tiext.sfc0001.print.custom.report	1288
tiext.sfc0001.register.custom.reports	1289
tiext.sfc0001.update.print.status	1290
ProductionOrder.SkipCalculateIntermediateResults	1292
ProductionOrder.SkipClose	1293
ProductionOrder.SkipCreateOrderGroup	1293
ProductionOrder.SkipGenerateSubcDocuments	1294
ProductionOrder.SkipPrintMaterialShortage	1294
ProductionOrder.SkipReportComplete	1295
ProductionOrder.SkipWarehouseForMaterialShortage	1295
Process Extensions for ProductVariant	1296
ProductVariant.AddCustomCPQIntegrationParameters	1296
tiext.pcf0003.get.cpq.integration.parameter.value	1297
tiext.pcf0003.register.cpq.integration.parameters	1298
ProductVariant.AddToCPQConfiguration	1299
tiext.pcf0002.add.to.cpq.configuration	1299
ProductVariant.AdjustPrice	1300
tiext.pcf0001.adjust.variant.price	1300
ProductVariant.SkipDeleteOrExpireJSBOM	1301
tiext.pcs0002.product.variant.skip.delete.or.expire.jsbom	1302
Process Extensions for ProjectBudget	1302
ProjectBudget.SkipGenerateServiceOrders	1303
Process Extensions for ProjectControl	1303
ProjectControl.SkipPrint	1303
Process Extensions for ProjectNetworkPlanning	1304
ProjectNetworkPlanning.SkipGenerate	1304
Process Extensions for ProjectPCS	1304
ProjectPCS.SkipClose	1304
tiext.pcs0001.project.pcs.skip.close	1305
tiext.pcs0001.project.pcs.skip.close.implemented	1305
ProjectPCS.SkipGlobalStatusUpdate	1305

Process Extensions for ProjectPlannedPurchaseOrder	1306
ProjectPlannedPurchaseOrder.SkipApprove	1306
Process Extensions for ProjectPlannedWarehouseOrder	1307
ProjectPlannedWarehouseOrder.SkipApprove	1308
Process Extensions for ProjectStructuralElement	1308
ProjectStructuralElement.SkipGenerate	1308
Process Extensions for ProjectTransaction	1309
ProjectTransaction.SkipProcess	1309
Process Extensions for PurchaseOrder	1309
PurchaseOrder.SkipApprove	1310
PurchaseOrder.SkipPrintPurchaseInvoice	1310
PurchaseOrder.SkipPrintPurchaseOrder	1311
PurchaseOrder.SkipPrintReceivableInvoice	1312
PurchaseOrder.SkipReleaseToWarehousing	1312
Process Extensions for PurchaseOrderGenerate	1313
PurchaseOrderGenerate.HandleCombining	1313
tdext.pur0003.determine.purchase.order.for.combining.order.lines.during.generation	1314
tdext.pur0003.suppress.combining.of.purchase.order.lines.to.existing.order.during.generation	1316
PurchaseOrderGenerate.SuppressCommingleLine	1318
tdext.pur0002.suppress.commingling.of.purchase.order.line.during.generation	1318
Process Extensions for PurchaseOrderLine	1319
PurchaseOrderLine.SkipPrintPurchaseInvoice	1320
PurchaseOrderLine.SkipPrintPurchaseOrder	1320
PurchaseOrderLine.SkipPrintPurchaseOrderReminder	1321
PurchaseOrderLine.SkipPrintReceivableInvoice	1322
Process Extensions for PurchaseSelfBilledInvoice	1323
PurchaseSelfBilledInvoice.CustomCompose	1323
tfext.acp0001.set.custom.composing.fields	1323
Process Extensions for QualityFailureDocument	1324
QualityFailureDocument.SkipPrint	1324
Process Extensions for QualityManagement	1325
QualityManagement.DefaultStandardTestProcedure	1325
qmext.ptc0001.get.customer.determined.default.standard.test.procedure	1325
Process Extensions for QualityNonConformance	1326
QualityNonConformance.SkipPrint	1327

Process Extensions for ReceivableInvoice	1327
ReceivableInvoice.SkipPrintSpecification	1327
Process Extensions for ReceivedCustomerOrderProcess	1328
ReceivedCustomerOrderProcess.Validate	1328
tdext.sls0005.validate.received.customer.order.during.processing	1329
Process Extensions for RequestForQuoteConvert	1330
RequestForQuoteConvert.CustomSorting	1330
tdext.pur0004.get.order.by.for.convert	1330
Process Extensions for ReservedApprovedCommissionsRebates	1331
ReservedApprovedCommissionsRebates.SkipPrint	1331
Process Extensions for ReservedCommissionsRebates	1332
ReservedCommissionsRebates.SkipReservationApproval	1332
Process Extensions for Sales	1333
Sales.SkipRetroBilledAdviceLineGeneration	1333
Process Extensions for SalesCheckInventory	1335
SalesCheckInventory.CustomSorting	1335
tdext.sls0001.get.customer.defined.order.by	1336
SalesCheckInventory.DefaultInventoryShortageOption	1336
tdext.sls0002.get.customer.determined.default.automatic.inventory.shortage.option	1337
SalesCheckInventory.SkipShortageLine	1338
Process Extensions for SalesContract	1340
SalesContract.SkipPrintSalesContract	1340
Process Extensions for SalesContractDeliveryScheme	1341
SalesContractDeliveryScheme.SkipGenerateOrder	1341
Process Extensions for SalesOrder	1342
SalesOrder.DetermineSupplier	1342
tdext.sls0006.sales.order.determine.supplier.for.direct.delivery	1342
SalesOrder.SkipAdditionalCostSet	1344
SalesOrder.SkipApprove	1346
SalesOrder.SkipPrint	1347
SalesOrder.SkipPrintAcknowledgement	1348
SalesOrder.SkipReleaseToWarehousing	1348
Process Extensions for SalesOrderGenerate	1350
SalesOrderGenerate.DetermineOrderForCombining	1350
tdext.sls0004.determine.sales.order.for.combining.order.lines.during.generation	1351
SalesOrderGenerate.SuppressCombiningOrderLines	1352

tdext.sls0003.suppress.combining.sales.order.lines.during.generation	1353
Process Extensions for SalesOrderInvoiceLine	1354
SalesOrderInvoiceLine.SkipReleaseToInvoicing	1354
Process Extensions for SalesOrderLine.....	1355
SalesOrderLine.AllowDisplaySelectedContract	1355
tdext.sls0008.sales.order.line.allow.display.selected.contract.....	1356
SalesOrderLine.AllowSaveInCaseOfInsufficientATP	1356
tdext.sls0007.sales.order.line.allow.save.in.case.of.insufficient.atp	1357
SalesOrderLine.SkipGenerateProjectStructure	1357
SalesOrderLine.SkipPrint	1358
SalesOrderLine.SkipPrintAcknowledgement	1359
Process Extensions for SalesQuote	1360
SalesQuote.SkipPrintSalesQuote	1360
Process Extensions for SalesQuoteLine	1360
SalesQuoteLine.SkipPrintSalesQuote	1361
Process Extensions for SelfBilledInvoiceLine	1361
SelfBilledInvoiceLine.SkipMatching	1362
Process Extensions for ServiceOrderActivity	1362
ServiceOrderActivity.SkipComplete	1363
Process Extensions for ShiftsByWorkcell.....	1363
ShiftsByWorkcell.SkipCompleteShift.....	1364
Process Extensions for ShipmentBuilding.....	1364
ShipmentBuilding.CustomBuildingConstraints.....	1364
whext.dli0003.custom.shipment.building.constraints	1364
Process Extensions for StandardCost.....	1366
StandardCost.SkipActualize	1366
tiext.cpr0001.standard.cost.skip.actualize	1366
StandardCost.SkipCalculate	1367
tiext.cpr0002.standard.cost.skip.calculate	1367
tiext.cpr0002.standard.cost.skip.calculate.item	1370
Process Extensions for Statement	1371
Statement.SkipPrintStatements	1371
Process Extensions for StorageMission.....	1372
StorageMission.HandleAfterGenerate	1372
whext.dli0006.storage.mission.handle.after.generate	1372
Process Extensions for TargetPrices	1373

TargetPrices.SkipCalculateOrderLines	1373
Process Extensions for TaxDeclaration	1375
TaxDeclaration.GetXMLTextTagContent	1376
tfext.gld0001.get.xml.text.tag.content.for.tax.declaration	1376
Process Extensions for WarehouseOrder	1378
WarehouseOrder.CreateNewHeader	1378
whext.dll0012.create.new.warehouse.order	1378

About this guide

Intended audience

This guide is intended for IT professionals working in implementation projects or IT optimization phases for Infor LN. Basic knowledge about the Infor LN software structure and Infor LN's 4GL programming language is a pre-requisite.

Related documents

You can find these documents on docs.infor.com:

- *Infor LN Studio Application Development Guide*
- *Infor LN Studio Integration Development Guide*
- *Infor LN Extensions Development Guide*

You can find the *Infor ES Programmer's Guide* in [KB2924522](https://kb.infor.com/kb2924522). The content of this guide is also available in the help pages of Infor LN Studio.

Contacting Infor

If you have questions about Infor products, go to Infor Concierge at <https://concierge.infor.com> and create a support incident.

For the latest documentation, go to Documentation Central at docs.infor.com. We recommend that you check this website periodically for updated documentation.

Chapter 1 Introduction

Infor LN is a standard ERP application with rich functionality. With its built-in flexibility with parameters, workflows, dynamic processes, it can be adjusted to serve the business processes in the industries Infor LN is designed for. To close the small gaps between the standard functionality and the specific business needs, Infor LN offers a variety of extensibility possibilities. The main goal of extensibility is to develop the last-mile functionality for your organization without changing the core standard software components and using only the public interfaces of the standard application. In this way, you can develop the extensions fully separated from the standard components and upgrading the standard software will therefore not result in additional efforts and costs for upgrading the customizations. Extensions are not influenced by the upgrade process.

Public Interfaces

Public Interfaces are methods with LN application functionality that can be called from extensions.

Public Interfaces explained

The development of extensions can be made easier if methods from the LN Application can be used. This can be done by using the so-called LN Public Interfaces. LN Public Interfaces are functions in the LN Application that are available for anyone who develops extensions on LN. The available LN Public Interfaces are visible in LN Studio and in the Extension Modeler. LN Public Interfaces are generic functions with a certain level of complexity that will likely be used by multiple customers. Simple read actions, for example, needs to be developed by customers or implementation partners themselves. Moreover, LN Public Interfaces will perform something what cannot easily be achieved with one or more standard sessions or processes in LN.

How to request a new Public Interface

Apply the following process if you need a new LN Public Interface:

- 1 Evaluate if the new LN Public Interface is generic and contains a certain level of complexity. Make sure the required feature cannot be achieved with personalizing a standard session or process.

- 2 Create an incident and clearly describe the required LN Public Interface. Use the Excel Sheet "Request Template for LN Public Interfaces & Process Extensions)" which is attached to [KB2003722](#) in the Infor Customer Portal.
- 3 Infor Support will create a defect for this incident.
- 4 Infor Development will review the requested LN Public Interface and will either develop the new LN Public Interface or reject the request. If the LN Public Interface is developed it will be released via the regular delivery process.
- 5 After the new solution has been installed the customer can use the new LN Public Interface in the extension modeler and/or in LN Studio.
- 6 Infor Support will complete the incident.

Public Interface example

Find below an example of one of the LN Public Interfaces. This Public Interface converts an amount to another currency.

```
long Common.ConvertAmount(
    domain    tcncmp    iFinancialCompany,
    domain    tcamnt    iSourceAmount,
    domain    tcccur    iSourceCurrency,
    domain    tcrtyp    iExchangeRateType,
    domain    tcdate    iRateDateUTC,
    domain    tcccur    iTargetCurrency,
    ref       domain    tcamnt    oTargetAmount,
    ref       domain    tcmcs.s999m oExceptionMessage mb,
    ref       long      oExceptionID)
```

It is very important to use LN Public Interfaces properly and to catch the errors. For this reason, each Public Interface has the output arguments 'oExceptionMessage' and 'oExceptionID'.

If the Public Interface returns a value unequal to zero, then argument oExceptionMessage is filled for sure with the latest exception message and argument oExceptionID is a reference to an XML-object that contains some more information about the exception. This extra information can be retrieved by using the Public Interface 'Exception' in otcextxtapi. It is also important to free up memory by calling Public Interface Exception.Delete(...).

If the Public Interface returns zero, arguments oExceptionMessage and oExceptionID could be filled due to information messages. So, in fact only the return value indicates if a public interface is successful or not.

The next example shows how to implement error handling when using LN Public Interfaces, in this case Common.ConvertAmount to convert an amount to another currency.

```
#pragma used dll "otcextextapi"
#pragma used dll "otcextemmapapi"

        long          exception.id, i
domain    tcmcs.s999m    exception.message

if Common.ConvertAmount(..., exception.message, exception.id) <> 0 then
    |* Exception(s) found.
    for i = 1 to Exception.NumberOfMessages(exception.id)
        dal.set.error.message("@"& Exception.GetMessage(i))
    endfor
    Exception.Delete(exception.id)
    return (DALHOOKERROR)
else
    |* Call was successful. Exception messages could exist!
    Exception.Delete(exception.id)
endif
```

To be able to use the Public Interface in your extensions, you need to add a “#pragma used dll” statement for the DLL that contains the Public Interface. The DLL names can be found in the documentation of the available Public Interfaces. Note that the DLL name needs to be preceded by an “o” in the #pragma-statement.

Available Public Interfaces

The next chapters of this document describe the available Public Interfaces for Infor LN (10.8) and what their usage is.

If the Public Interfaces described in this document are not shown in the Extension Modeler or Infor LN Studio in your environment, it may be necessary to apply a Knowledge Base article (KB) that can be found in the Infor Customer Portal. The applicable KB number is mentioned in the Public Interface description.

Public Interface to call BDE methods

BDEs (Business Data Entities) are components in Infor LN that are the base for Infor LN's web services. A special Public Interface is available to call the BDE methods within extensions: BDE.ExecuteMethod. Note that this will not be a real (SOAP) web service call, but the BDE method will be executed internally in LN.

The available BDEs are shown in session “Business Objects” (ttadv7500m000). The BDEs that are available to be called as Public Interface must satisfy the following conditions:

- The name must not end with “BOD”
- The Type of Business Object must be ‘Public’

- The Maintained in Studio checkbox must be checked
- The action Download WSDL must be available for the BDE.

Proxy DLL

BDE methods are functions that have an XML document as input (the request), and the output is either a response XML document (in case the method succeeded) or a result XML document (in case the method failed). To build the request or to retrieve data from the response or result, you need to use a Proxy DLL that is generated based on the WSDL (Web Service Description Language). To be able to generate a proxy DLL you need to have a Development license (product ID 10146).

Perform the following steps to generate a Proxy DLL:

- 1 Select the BDE you want to call in your extension in session “Business Objects” (ttadv7500m000).
- 2 Click **Actions>Download WSDL**.
- 3 Save the WSDL in a folder on your PC.
- 4 Start Infor LN Studio. At least version 10.7.0.389 is needed. For more information about Infor LN Studio see the Related Documents section of this document.
- 5 If you don’t have an Activity yet, create a new one.
- 6 Create a new software component of type Library. Enter a name and a description and click **Finish**.
- 7 A new library is created, and the editor is opened for it. Check out the library.
- 8 Right-click in the editor and select **Generate Source from WSDL**.
- 9 The “Generate Library from WSDL file” displays. Browse to the WSDL file you saved on your PC in step 3.
- 10 Click **OK**.
- 11 The source is generated. Click **Save** button.
- 12 If you need to use the proxy DLL in another LN Studio Activity or another Extensibility Activity, the proxy DLL must be committed by (partially) ending the current Activity.

Coding example

This program creates and releases a Service Order using the Create and ReleaseOrder methods of the ServiceOrder_v4 BDE. The numbers behind |# refer to remarks after the example program.

```
#pragma used dll "otcextextapi" |# 1
#pragma used dll "otcextbdeapi" |# 2

function create.and.release.service.order()
{
```

Introduction

```
long    request, response, result
long    so.order, so.act, so.ass
long    ro
long    da
long    ret
long    exceptionid
string  exceptionmessage(1000) mb

domain  tcorno  service.order

db.retry.point()                                     |# 3

request = Create_CreateRequest.New()                 |# 4

so.order = Create_ServiceOrder_v4.New("JJD")          |# 5
Create_ServiceOrder_v4.SetserviceCenter(so.order, "100") |# 6

da = Create_DataArea.New()                           |# 7
ret = Create_DataArea.AddServiceOrder_v4(da, so.order)
Create_CreateRequest.SetDataArea(request, da)

so.act = Create_ServiceOrderActivity.New()            |# 8
ret = Create_ServiceOrder_v4.AddServiceOrderActivity(so.order, so.act) |# 9
|# 6
Create_ServiceOrderActivity.SetserviceOrderActivityPlannedStartTime(so.act, utc.num())
Create_ServiceOrderActivity.SetserviceOrderActivityPlannedFinishTime(
    so.act, utc.num() + 3600)

so.ass = Create_ActivityEngineer.New()                |# 10
ret = Create_ServiceOrderActivity.AddActivityEngineer(so.act, so.ass) |# 11
Create_ActivityEngineer.SetassignmentEngineer(so.ass, "0099101") |# 6

ret = BDE.ExecuteMethod("ServiceOrder_v4",           |# 12
    "Create",
    request,
    response,
    result,
    exceptionmessage,
    exceptionid)

if ret <> 0 then                                     |# 13
    abort.transaction()
    message(exceptionmessage)
    Exception.Delete(exceptionid)
    return
endif

ret = Create_CreateResponse.GetDataArea(response, da) |# 14
```

```

so.order = Create_DataArea.GetServiceOrder_v4(da)
service.order = Create_ServiceOrder_v4.GetServiceOrderCode(so.order)

xmlDelete(request)
xmlDelete(response)

request = ReleaseOrder_ReleaseOrderRequest.New() |# 15

ro = ReleaseOrder_ServiceOrder_v4.New(service.order)

da = ReleaseOrder_DataArea.New()
ret = ReleaseOrder_DataArea.AddServiceOrder_v4(da, ro)
ReleaseOrder_ReleaseOrderRequest.SetDataArea(request, da)

ret = BDE.ExecuteMethod("ServiceOrder_v4", |# 16
                        "ReleaseOrder",
                        request,
                        response,
                        result,
                        exceptionmessage,
                        exceptionid)

if ret <> 0 then |# 13
    abort.transaction()
    message(exceptionmessage)
    Exception.Delete(exceptionid)
    return
endif

Exception.Delete(exceptionid) |# 17

commit.transaction() |# 18
message(sprintf$("Service Order %s has been created and released.", service.order))
}

```

Explanation:

- 1 ottextxtapi contains the functions for exception handling.
- 2 ottextbdeapi contains the function BDE.ExecuteMethod().
- 3 Transaction handling must always be in the calling program of the BDE method. Note that when you execute a BDE method in an extension hook (for example in the Before Save hook in a table extension, your hook must not start an own transaction, because it needs to be executed in the transaction of the program that calls the table extension.
- 4 This is the initialization of a new request that must be built up. Use always the functions from the proxy DLL with the method name followed by an “_” as prefix.
- 5 The example creates a Service Order. The root component is the Service Order itself and the component name is the same as the BDE name, ServiceOrder_v4 in this case. Because the

Service Order number is a mandatory attribute of the Service Order, it must be passed as an argument. In this case the series in which the Service Order must be created.

- 6 Other attributes of the Service Order can be set as well. This also applies to the attributes of other components of the Service Order (Activity and Assignment Engineer) which are added later.
- 7 The Service Order is not directly connected to the request, but within the DataArea of the request. Here the DataArea is created, the Service Order is linked to it and the DataArea is connected to the request.
- 8 An Activity is created and
- 9 linked to the Service Order.
- 10 An Assignment Engineer is created and
- 11 linked to the Activity.
- 12 This is the call of the Public Interface to execute the BDE method to create the Service Order including the Activity and the Assignment Engineer. The XML request that has been built up by calling the functions is the proxy DLL looks like:

```
<Create.CreateRequest>
  <DataArea>
    <ServiceOrder_v4>
      <serviceOrderCode>JJD</serviceOrderCode>
      <serviceCenter>100</serviceCenter>
      <ServiceOrderActivity>
        <serviceOrderActivityPlannedStartTime>2019-05-03T07:22:35+02:00
        </serviceOrderActivityPlannedStartTime>
        <serviceOrderActivityPlannedFinishTime>2019-05-03T08:22:35+02:00
        </serviceOrderActivityPlannedFinishTime>
        <ActivityEngineer>
          <assignmentEngineer>0099101</assignmentEngineer>
        </ActivityEngineer>
      </ServiceOrderActivity>
    </ServiceOrder_v4>
  </DataArea>
</Create.CreateRequest>
```

- 13 If the method fails, the transaction must be aborted. When the transaction is started outside your code (for example when your code is part of a table extension), you don't need to abort, but return the error (DALHOOKERROR). The *exceptionmessage* field contains the most important message from the BDE method call. If you need more messages, you can use the Exception object (see public Interfaces for Extensibility) to retrieve the other messages.
- 14 Retrieve the generated Service Order Number from the response by getting the DataArea of the response XML document, getting the Service Order from the DataArea and then getting the Service Order Number attribute of the Service Order. The XML response of the Create method looks like (the actual response contains much more elements, but they are left out for readability):


```

<CreateResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <DataArea>
    <ServiceOrder_v4>
      <serviceOrderCode>JJD000235</serviceOrderCode>
      <customerOrderReference></customerOrderReference>
      ...
    <ServiceOrderActivity>
      <serviceOrderActivityLineNumber>10</serviceOrderActivityLineNumber>
      <activityStatus>free</activityStatus>
      <serviceOrderActivityPlannedStartTime>2019-05-03T05:22:35Z
      </serviceOrderActivityPlannedStartTime>
      <serviceOrderActivityPlannedFinishTime>2019-05-03T06:22:35Z
      </serviceOrderActivityPlannedFinishTime>

      ...
    <ActivityEngineer>
      <assignmentLineNumber>1</assignmentLineNumber>
      <assignmentStatus>assigned</assignmentStatus>
      <assignmentEngineer>0099101</assignmentEngineer>
      ...
    </ActivityEngineer>
  </ServiceOrderActivity>
</ServiceOrder_v4>
</DataArea>
</CreateResponse>

```

- 15 Use this Service Order Number to build up the ReleaseOrder request.
- 16 Execute the ReleaseOrder method. If this method fails, the transaction needs to be aborted. In this case the created Service Order will be reverted as well.
- 17 Free the Exception object, which may contain information messages.
- 18 Commit the transaction. See also note 3 and 11.

Process Extensions

Process Extensions are the opposite of Public Interfaces. Instead of calling standard LN methods (Public Interfaces) from the extensions, the standard LN application calls the Process Extensions, provided that they are implemented.

Process Extensions explained

With the extension types table extension, session extension, etc. the behavior and functionality of Infor LN can be changed in many ways (see the Infor LN Extensions Development Guide). But there

may be additional requirements where intervening the standard functionality is required, which cannot be achieved with extending the table, session or any other component. Examples are processing and/or print sessions where additional selection ranges are required or core algorithms for composing invoices where the standard criteria are not sufficient, especially when customer defined fields need to be included in the selection and/or algorithm.

How to request a new Process Extension

The possibility to skip objects in processing and/or print sessions is a generic approach and can be used in every session where this Process Extension type has been implemented in the standard application. If you need to additional criteria whether or not objects may be processed by a session and that session has not been prepared for this Process Extension type, you can request this.

The other types of Process Extensions, where core algorithms can be extended, can be requested as well.

Apply the following process if you need a new Process Extension:

- 1 Evaluate if the request is generic. Make sure the required feature cannot be achieved with personalizing a standard session or process or by using one of the other extension points (for example a table extension).
- 2 In case of a skip Process Extension type, create an incident in the Infor Customer Portal. In case of another Process Extension type, create an enhancement request in Infor's ERS system.
- 3 Infor Development will review the requested Process Extension and will either include it in the standard product or reject the request. If the Process Extension is developed it will be released via the regular delivery process.
- 4 After the new solution has been installed the customer can implement the new Process Extension using the extension modeler.
- 5 If an incident was created, Infor Support will complete the incident.

Available Process Extensions

The last chapter of this document describes the available Process Extensions for Infor LN (10.8) and how they must be implemented.

If the Process Extensions described in this document are not shown in the Extension Modeler in your environment, it may be necessary to apply a Knowledge Base article (KB) that can be found in the Infor Customer Portal. The applicable KB number is mentioned in the Process Extension description.

Chapter 2 Public Interfaces for Extensibility

Public Interfaces for Exception

The following functions are available:

[Exception.Delete](#)

[Exception.GetMessage](#)

[Exception.NumberOfMessages](#)

Exception.Delete

DLL:	tcextextapi
Syntax:	Exception.Delete(ref long ioExceptionID)
Usage:	Expl: This function deletes the exception to free memory. Pre: ioExceptionID should refer to an Exception. Post: None Input: ioExceptionID - the exception id. Output: ioExceptionID - the exception id. Return: None

Exception.GetMessage

DLL:	tcextextapi
Syntax:	Exception.GetMessage(long iExceptionID, long iMessageIndex, ref domain tcmcs.s999m oMessageDescription mb)

Usage:	Expl: This function reads message description from all messages in the XML identified by iExceptionID for a certain index. Pre: iExceptionID should refer to an Exception. Post: None Input: - iExceptionID - the exception id. - iMessageIndex - the index for the message to be returned. - iMessageIndex should be greater than zero and less than the number of messages. Output: - oMessageDescription - The found message. Return: None

Exception.NumberOfMessages

DLL:	tcextextapi
Syntax:	long Exception.NumberOfMessages (long iExceptionID)
Usage:	Expl: This function determines the number of messages that have been stored in the XML where iExceptionID refers to. Pre: iExceptionID should refer to a valid XML with the structure as described above. Post: None Input: - iExceptionID - the exception id that points to the XML. Output: - None Return: The number of found messages. If iExceptionID does not refer to a valid XML the return value is 0.

Public Interfaces for ProcessingOptionSet

The following functions are available:

[ProcessingOptionSet.Create](#)

[ProcessingOptionSet.Delete](#)

ProcessingOptionSet.Create

DLL:	tcextextapi														
	This function is available from KB3500234 .														
Syntax:	<pre> long ProcessingOptionSet.Create(ref long oProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>														
Usage:	Expl: Create a Processing Option Set. The created Processing Option Set can be used to control Public Interfaces that support this functionality. Name/Value pairs must be given in the variable arguments. Example flow for Item.Copy(): <pre> ret = ProcessingOptionSet.Create(my.copy.item.processing.option.set, my.exception.message1, my.exception.id1, "copyItemsBySiteAndItemsByOffice", tcyesno.no, "copyItemText", tcyesno.yes, "copyReferenceDesignators", tcyesno.no, "targetItemDescription", "This is the copied item", "projectPartQuantity", 751.12) . . ret = Item.Copy(source.item, target.item, my.copy.item.processing.option.set, my.exception.message2, my.exception.id2) . . * release memory ret = ProcessingOptionSet.Delete(my.copy.item.processing.option.set) . . </pre> Pre: - Post: Use ProcessingOptionSet.Delete() to release memory for the allocated Processing Option Set. Input: Variable Arguments - repetition of pairs: <table> <tr> <td>OptionName</td> <td>- option name of type string</td> </tr> <tr> <td>OptionValue</td> <td>- option value, type as defined by Public Interface usage.</td> </tr> </table> Output: <table> <tr> <td>oProcessingOptionSet</td> <td>- reference to the processing option set</td> </tr> <tr> <td>oExceptionMessage</td> <td>- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.</td> </tr> <tr> <td>oExceptionID</td> <td>- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.</td> </tr> </table> Return: <table> <tr> <td>0</td> <td>- successfully created</td> </tr> <tr> <td><> 0</td> <td>- failed.</td> </tr> </table>	OptionName	- option name of type string	OptionValue	- option value, type as defined by Public Interface usage.	oProcessingOptionSet	- reference to the processing option set	oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.	oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.	0	- successfully created	<> 0	- failed.
OptionName	- option name of type string														
OptionValue	- option value, type as defined by Public Interface usage.														
oProcessingOptionSet	- reference to the processing option set														
oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.														
oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.														
0	- successfully created														
<> 0	- failed.														

ProcessingOptionSet.Delete

DLL:	tcextextapi This function is available from KB3500234 .
Syntax:	<pre>long ProcessingOptionSet.Delete(ref long ioProcessingOptionSet)</pre>
Usage:	Expl: Release memory for a created Processing Option Set/ Input/Output: ioProcessingOptionSet - reference to Processing Option Set Return: 0 - successfully deleted <> 0 - failure.

Chapter 3 Public Interfaces for Common

Public Interfaces for Address

The following functions are available:

[Address.Create](#)

[Address.StartDetail](#)

[Address.StartPrintByBusinessPartner](#)

Address.Create

DLL:	ttextcomapi		
	This function is available from KB2314506 .		
Syntax:	<pre> long Address.Create(domain tcnama iAddressName mb, domain tcccty iCountry, domain tcdsca iCityName mb, boolean iSetStateProvince, domain tcmcs.cste iStateProvince, boolean iSetName2, domain tcnama iName2 mb, boolean iSetCity2, domain tcnamf iCity2 mb, boolean iSetPostalCode, domain tcpstc iPostalCode mb, boolean iSetStreet, domain tcnamc iStreet mb, boolean iSetStreet2, domain tcnamd iStreet2 mb, boolean iSetHouseNumber, domain tccom.nmbr iHouseNumber mb, boolean iSetPOBoxNumber, domain tccom.nmbr iPOBoxNumber mb, boolean iSetBuilding, domain tccom.bldg iBuilding mb, boolean iSetBuildingFloor, domain tccom.blfl iBuildingFloor mb, </pre>		

	<pre> boolean iSetBuildingUnit, domain tccom.blun iBuildingUnit mb, boolean iSetAddressLine1, domain tcmcs.str100m iAddressLine1 mb, boolean iSetAddressLine2, domain tcmcs.str100m iAddressLine2 mb, boolean iSetAddressLine3, domain tcmcs.str100m iAddressLine3 mb, boolean iSetAddressLine4, domain tcmcs.str100m iAddressLine4 mb, boolean iSetAddressLine5, domain tcmcs.str100m iAddressLine5 mb, boolean iSetAddressLine6, domain tcmcs.str100m iAddressLine6 mb, ref domain tccom.cadr oAddressCode, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function creates a new address. If an existing address code is found based on the given input information then the existing address code is returned. Otherwise a new address code is created and returned. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: <pre> iAddressName - Name of the Address (This is a Mandatory field.) iCountry - Country (This is a Mandatory field.) iCityName - City Name (This is a Mandatory field.) iSetStateProvince - Set State/Province iStateProvince - State/Province iSetName2 - Set Name 2 (True/False) iName2 - Name 2 iSetCity2 - Set City 2 (True/False) iCity2 - City 2 iSetPostalCode - Set ZIP Code/Postal Code (True/False) iPostalCode - ZIP Code/Postal Code iSetStreet - Set Street (True/False) iStreet - Street iSetStreet2 - Set Street 2 (True/False) iStreet2 - Street 2 iSetHouseNumber - Set House Number (True/False) iHouseNumber - House Number iSetPOBoxNumber - Set P.O. Box Number (True/False) iPOBoxNumber - P.O. Box Number iSetBuilding - Set Building (True/False) iBuilding - Building iSetBuildingFloor - Set Building Floor (True/False) iBuildingFloor - Building Floor iSetBuildingUnit - Set Building Unit (True/False) iBuildingUnit - Building Unit iSetAddressLine1 - Set Address Line 1 (True/False) iAddressLine1 - Address Line 1 iSetAddressLine2 - Set Address Line 2 (True/False) iAddressLine2 - Address Line 2 iSetAddressLine3 - Set Address Line 3 (True/False) iAddressLine3 - Address Line 3 iSetAddressLine4 - Set Address Line 4 (True/False) iAddressLine4 - Address Line 4 iSetAddressLine5 - Set Address Line 5 (True/False) iAddressLine5 - Address Line 5 iSetAddressLine6 - Set Address Line 6 (True/False) </pre>

	iAddressLine6 Output: oAddressCode oExceptionMessage oExceptionID Return: 0 <> 0	- Address Line 6 - Address Code - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. - Address Found/Create - Error

Address.StartDetail

DLL:	ttextcomapi This function is available from KB3502076 .	
Syntax:	<pre> long Address.StartDetail(long iStartMode, domain tccom.cadr iAddress, boolean iReadOnly, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>	
Usage:	Expl: This function starts session Addresses (tccom4130s000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iAddress: Mandatory iReadOnly: Optional: Session is started in readonly-mode when iReadOnly is True Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - Session started - Error	

Address.StartPrintByBusinessPartner

DLL:	tcextcomapi																																																																				
	This function is available from KB3566667 .																																																																				
Syntax:	long Address.StartPrintByBusinessPartner(domain tccom.bpid iBusinessPartnerFrom, domain tccom.bpid iBusinessPartnerTo, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)																																																																				
Usage:	Expl: This function starts session 'Print Addresses by Business Partner' (ttccom4430m100). Input: iBusinessPartnerFrom - Business Partner From iBusinessPartnerTo - Business Partner To: Mandatory iProcessingOptionSet - Optional, if 0, the session is started with regular defaulting logic (user defaults or session defaults), except selection range for Business Partner, which is defaulted with iBusinessPartnerFrom and iBusinessPartnerTo. A Processing Option Set can be created via a call to ProcessingOptionSet.Create(). Processing Options have a direct relationship with the form fields on session Print Addresses by Business Partner (tccom4430m100) and are not explained in further detail here. Please refer to the session help for additional information. Options which are not available as Processing Options will get defaulted with the value in column DEFAULT below. Processing Options that are set while a required Implemented Software Component is not available are ignored. Explanation about setting of defaults: Minimum Value: Minimum value of domain is taken as default value. Maximum Value: Maximum value of domain is taken as default value. <table><tr><td>NAME</td><td>TYPE</td><td>DEFAULT</td></tr><tr><td>BusinessPartnerFrom</td><td>domain tccom.bpid</td><td>iBusinessPartnerFrom</td></tr><tr><td>BusinessPartnerTo</td><td>domain tccom.bpid</td><td>iBusinessPartnerTo</td></tr><tr><td>SearchKeyFrom</td><td>domain tcseak</td><td>minimum value</td></tr><tr><td>SearchKeyTo</td><td>domain tcseak</td><td>maximum value</td></tr><tr><td>ParentFrom</td><td>domain tccom.bpid</td><td>minimum value</td></tr><tr><td>ParentTo</td><td>domain tccom.bpid</td><td>maximum value</td></tr><tr><td>CountryFrom</td><td>domain tcccty</td><td>minimum value</td></tr><tr><td>CountryTo</td><td>domain tcccty</td><td>maximum value</td></tr><tr><td>ZIPCodeFrom</td><td>domain tcpstc</td><td>minimum value</td></tr><tr><td>ZIPCodeTo</td><td>domain tcpstc</td><td>maximum value</td></tr><tr><td>CityFrom</td><td>domain tccity</td><td>minimum value</td></tr><tr><td>CityTo</td><td>domain tccity</td><td>maximum value</td></tr><tr><td>SortBy</td><td>domain tccom.sortbpad</td><td>tccom.sortbpad.bpartner</td></tr><tr><td>BusinessPartners</td><td>domain tcyesno</td><td>tcyesno.no</td></tr><tr><td>SoldTo</td><td>domain tcyesno</td><td>tcyesno.no</td></tr><tr><td>ShipTo</td><td>domain tcyesno</td><td>tcyesno.no</td></tr><tr><td>InvoiceTo</td><td>domain tcyesno</td><td>tcyesno.no</td></tr><tr><td>PayBy</td><td>domain tcyesno</td><td>tcyesno.no</td></tr><tr><td>BuyFrom</td><td>domain tcyesno</td><td>tcyesno.no</td></tr><tr><td>ShipFrom</td><td>domain tcyesno</td><td>tcyesno.no</td></tr><tr><td>InvoiceFrom</td><td>domain tcyesno</td><td>tcyesno.no</td></tr></table>			NAME	TYPE	DEFAULT	BusinessPartnerFrom	domain tccom.bpid	iBusinessPartnerFrom	BusinessPartnerTo	domain tccom.bpid	iBusinessPartnerTo	SearchKeyFrom	domain tcseak	minimum value	SearchKeyTo	domain tcseak	maximum value	ParentFrom	domain tccom.bpid	minimum value	ParentTo	domain tccom.bpid	maximum value	CountryFrom	domain tcccty	minimum value	CountryTo	domain tcccty	maximum value	ZIPCodeFrom	domain tcpstc	minimum value	ZIPCodeTo	domain tcpstc	maximum value	CityFrom	domain tccity	minimum value	CityTo	domain tccity	maximum value	SortBy	domain tccom.sortbpad	tccom.sortbpad.bpartner	BusinessPartners	domain tcyesno	tcyesno.no	SoldTo	domain tcyesno	tcyesno.no	ShipTo	domain tcyesno	tcyesno.no	InvoiceTo	domain tcyesno	tcyesno.no	PayBy	domain tcyesno	tcyesno.no	BuyFrom	domain tcyesno	tcyesno.no	ShipFrom	domain tcyesno	tcyesno.no	InvoiceFrom	domain tcyesno	tcyesno.no
NAME	TYPE	DEFAULT																																																																			
BusinessPartnerFrom	domain tccom.bpid	iBusinessPartnerFrom																																																																			
BusinessPartnerTo	domain tccom.bpid	iBusinessPartnerTo																																																																			
SearchKeyFrom	domain tcseak	minimum value																																																																			
SearchKeyTo	domain tcseak	maximum value																																																																			
ParentFrom	domain tccom.bpid	minimum value																																																																			
ParentTo	domain tccom.bpid	maximum value																																																																			
CountryFrom	domain tcccty	minimum value																																																																			
CountryTo	domain tcccty	maximum value																																																																			
ZIPCodeFrom	domain tcpstc	minimum value																																																																			
ZIPCodeTo	domain tcpstc	maximum value																																																																			
CityFrom	domain tccity	minimum value																																																																			
CityTo	domain tccity	maximum value																																																																			
SortBy	domain tccom.sortbpad	tccom.sortbpad.bpartner																																																																			
BusinessPartners	domain tcyesno	tcyesno.no																																																																			
SoldTo	domain tcyesno	tcyesno.no																																																																			
ShipTo	domain tcyesno	tcyesno.no																																																																			
InvoiceTo	domain tcyesno	tcyesno.no																																																																			
PayBy	domain tcyesno	tcyesno.no																																																																			
BuyFrom	domain tcyesno	tcyesno.no																																																																			
ShipFrom	domain tcyesno	tcyesno.no																																																																			
InvoiceFrom	domain tcyesno	tcyesno.no																																																																			

PayTo	domain	tctyesno	tctyesno.no
AllAddressesPerRole	domain	tctyesno	tctyesno.no
Text	domain	tctyesno	tctyesno.no
Output: oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.		
oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.		
Return: 0	- Session started		
<> 0	- Error		

Public Interfaces for Warehouse

The following functions are available:

[Warehouse.StartOverview](#)

Warehouse.StartOverview

DLL:	tcextemapi		
	This function is available from KB3501694 .		
Syntax:	<pre>long Warehouse.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcncmp iOperationalCompany, domain tccwar iWarehouse, domain tcemm.grid iEnterpriseUnit, domain tcemm.clus iPlanningCluster, domain tcsite iSite, boolean iInEnterprisePlanningOnly, ref domain tccwar oWarehouse, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>		
Usage:	Expl: This function starts session Warehouses (tcemm1112m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be		

	MODELESS - started as a zoom session. Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Optional: Specifies the start filter Possible values are: "byCompanyCluster": Filter on specified variables iOperationalCompany and iPlanningCluster "byCompanyWarehouse": Filter on specified variables iOperationalCompany and iWarehouse "byCompany": Filter on specified variable iOperationalCompany iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend Optional: Specific query to be used in session. iOperationalCompany Optional: Filter on Operational Company with filled iStartFilter iWarehouse Optional: Filter on Warehouse with filled iStartFilter iEnterpriseUnit Optional: Filter on Enterprise Unit iPlanningCluster Optional: Filter on Planning Cluster with filled iStartFilter iSite Optional: Filter on Site iInEnterprisePlanningOnly Optional: Filter on Include in EP when iInEpOnly is True Output: oWarehouse - for iStartMode MODAL: Selected Warehouse oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

Public Interfaces for Common

The following functions are available:

[Common.CalculateDiscountAmounts](#)[Common.CalculateDueDates](#)[Common.ConvertAmount](#)[Common.ConvertAmountToWords](#)[Common.ConvertISODurationToSeconds](#)[Common.ConvertLeadTimeToISODuration](#)[Common.ConvertPrice](#)[Common.ConvertQuantity](#)[Common.ConvertTime](#)[Common.GenerateFirstFreeNumber](#)[Common.GetAdditionalEntityInfo](#)[Common.GetConversionFactor](#)[Common.GetCurrencyRates](#)[Common.GetFinancialCompanyOfEntity](#)[Common.GetFormattedAddress](#)[Common.GetISOCODEOfLNLanguage](#)[Common.GetParameters](#)[Common.RoundAmount](#)[Common.RoundQuantity](#)[Common.UpdateApprovedConversionFactor](#)[Common.ValidateNumberGroupSeries](#)

Common.CalculateDiscountAmounts

DLL:	tcextmcsapi		
	This function is available from KB2305449 .		
Syntax:	<pre>long Common.CalculateDiscountAmounts(domain tcncmp iFinancialCompany, domain tccpay iPaymentTerms, domain tcccur iInvoiceCurrency, domain tcamnt iInvoiceAmount, domain tcamnt iTaxAmount, ref domain tcmcs.long oNumberOfScheduleLines, ref domain tcamnt oScheduleLineAmountArray(), ref domain tcamnt oFirstDiscountAmountArray(),</pre>		

	<pre> ref domain tcamnt oSecondDiscountAmountArray(), ref domain tcamnt oThirdDiscountAmountArray(), ref domain tcmcs.s250m oErrorMessage mb, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface is used to calculate the schedule amounts and the discount amounts for a certain payment terms. If a schedule is linked to the payment terms, the calculated amounts are based on the schedule line details of tcmcs221. If no schedule is linked to the payment terms, the calculated amounts are based on the payment terms details of tcmcs013. Pre: The array output arguments should be declared as BASED. Post: Free the allocated memory of the arrays. Input: - iFinancialCompany - Financial Company This is a Mandatory field. - iPaymentTerms - Payment Terms This is a Mandatory field. - iInvoiceCurrency - Invoice Currency This is a Mandatory field. - iInvoiceAmount - Invoice Amount - iTaxAmount - Tax Amount Output: - oNumberOfScheduleLines - Number of Schedule Lines - oScheduleLineAmountArray - Array of Schedule Line Amount - oFirstDiscountAmountArray - Array of First Discount Amount - oSecondDiscountAmountArray - Array of Second Discount Amount - oThirdDiscountAmountArray - Array of Third Discount Amount - oErrorMessage - Error Message - oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Success <> 0 - Error calculating the discount amounts

Common.CalculateDueDates

DLL:	tcextmcsapi This function is available from KB3540996.
Syntax:	<pre> long Common.CalculateDueDates (domain tcncmp iFinancialCompany, domain tccpay iPaymentTerms, domain tcccp.ccal iCalendar, domain tcccp.ract iAvailabilityType, domain tcgld.date iInvoiceDate, ref domain tcmcs.long oNumberOfScheduleLines, ref domain tcgld.date oDueDateArray(), </pre>

	<pre> ref domain tcgld.date oFirstDiscountDateArray(), ref domain tcgld.date oSecondDiscountDateArray(), ref domain tcgld.date oThirdDiscountDateArray(), ref domain tcmcs.s250m oErrorMessage mb, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface is used to calculate the due dates and the discount dates for a certain payment terms. If a schedule is linked to the payment terms, the calculated dates are based on the schedule line details of tcmcs221. If no schedule is linked to the payment terms, the calculated dates are based on the payment terms details of tcmcs013. Pre: The array output arguments should be declared as BASED. Post: Free the allocated memory of the arrays. Input: - iFinancialCompany - Financial Company This is a Mandatory field. - iPaymentTerms - Payment Terms This is a Mandatory field. - iCalendar - Calendar - iAvailabilityType - Availability Type - iInvoiceDate - Invoice Date Default set to Current Date if passed empty. Output: - oNumberOfScheduleLines - Number of Schedule Lines - oDueDateArray - Array of Due Dates - oFirstDiscountDateArray - Array of First Discount Date - oSecondDiscountDateArray - Array of Second Discount Date - oThirdDiscountDateArray - Array of Third Discount Date - oErrorMessage - Error Message - oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Success <> 0 - Error calculating the due dates

Common.ConvertAmount

DLL:	tcextemmap
Syntax:	<pre> long Common.ConvertAmount(domain tcncmp iFinancialCompany, domain tcamnt iSourceAmount, domain tcccur iSourceCurrency, domain tcrtyp iExchangeRateType, domain tcdate iRateDateUTC, domain tcccur iTargetCurrency, ref domain tcamnt oTargetAmount, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function converts an amount in one currency to an amount in another currency.

	The rates are read based on the given iRateDateUTC for the given iExchangeRateType. If the iExchangeRateType is not passed, the function will use the Internal Rate Type defined for the company. Pre: Company must be a valid company. Post: None Input: <table> <tr> <td>iFinancialCompany</td><td>- Financial Company: Mandatory</td></tr> <tr> <td>iSourceAmount</td><td>- Source Amount</td></tr> <tr> <td>iSourceCurrency</td><td>- Source Currency: Mandatory</td></tr> <tr> <td>iExchangeRateType</td><td>- Source Rate Type; if not filled, the Internal Rate Type will be used.</td></tr> <tr> <td>iRateDateUTC</td><td>- Rate Date (UTC)</td></tr> <tr> <td>iTargetCurrency</td><td>- Target Currency: Mandatory</td></tr> </table> Output: <table> <tr> <td>oTargetAmount</td><td>- Target Amount, result of the conversion.</td></tr> <tr> <td>oExceptionMessage</td><td>- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID</td></tr> <tr> <td>oExceptionID</td><td>- An ID that refers to all error information. Use the functions in Exception to get all relevant information.</td></tr> </table> Return: 0 - Amount is converted. <> 0 - Otherwise.	iFinancialCompany	- Financial Company: Mandatory	iSourceAmount	- Source Amount	iSourceCurrency	- Source Currency: Mandatory	iExchangeRateType	- Source Rate Type; if not filled, the Internal Rate Type will be used.	iRateDateUTC	- Rate Date (UTC)	iTargetCurrency	- Target Currency: Mandatory	oTargetAmount	- Target Amount, result of the conversion.	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
iFinancialCompany	- Financial Company: Mandatory																		
iSourceAmount	- Source Amount																		
iSourceCurrency	- Source Currency: Mandatory																		
iExchangeRateType	- Source Rate Type; if not filled, the Internal Rate Type will be used.																		
iRateDateUTC	- Rate Date (UTC)																		
iTargetCurrency	- Target Currency: Mandatory																		
oTargetAmount	- Target Amount, result of the conversion.																		
oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID																		
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.																		

Common.ConvertAmountToWords

DLL:	tcextmcsapi This function is available from KB2274799.								
Syntax:	<pre>long Common.ConvertAmountToWords (domain tcamnt iAmount, domain tclang iSystemLanguage, ref domain tcmcs.s130m oAmountInWordsBeforeDecimalSign mb, ref domain tcmcs.st65m oAmountInWordsAfterDecimalSign mb, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>								
Usage:	Expl: This function converts an amount into words in the given language. Pre: NA Post: NA Input: <table> <tr> <td>iAmount</td><td>- Amount to be converted into written text</td></tr> <tr> <td>iSystemLanguage</td><td>- SystemLanguage in which oAmountInWordsBeforeDecimalSign and oAmountInWordsAfterDecimalSign must be expressed.(mandatory)</td></tr> </table> Output: <table> <tr> <td>oAmountInWordsBeforeDecimalSign</td><td>- iAmount before decimal sign in written text</td></tr> <tr> <td>oAmountInWordsAfterDecimalSign</td><td>- iAmount after decimal sign in written text</td></tr> </table>	iAmount	- Amount to be converted into written text	iSystemLanguage	- SystemLanguage in which oAmountInWordsBeforeDecimalSign and oAmountInWordsAfterDecimalSign must be expressed.(mandatory)	oAmountInWordsBeforeDecimalSign	- iAmount before decimal sign in written text	oAmountInWordsAfterDecimalSign	- iAmount after decimal sign in written text
iAmount	- Amount to be converted into written text								
iSystemLanguage	- SystemLanguage in which oAmountInWordsBeforeDecimalSign and oAmountInWordsAfterDecimalSign must be expressed.(mandatory)								
oAmountInWordsBeforeDecimalSign	- iAmount before decimal sign in written text								
oAmountInWordsAfterDecimalSign	- iAmount after decimal sign in written text								

	NOTE: For the most languages the decimals are written in digits. Like: 'und 43/100' for German. Only if there is a legal requirement for a language, decimals are written in words. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - iAmount is converted into words. - on Errors

Common.ConvertISODurationToSeconds

DLL:	tcextcomapi This function is available from KB2324302.												
Syntax:	<pre>long Common.ConvertISODurationToSeconds(domain tcmcs.st24 iISODuration, ref double oNumberOfSeconds, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>												
Usage:	Expl. : This Public Interface converts an ISO duration string to a time in seconds. Example 1: <table> <tr> <td>ISO Duration</td> <td>"P5Y2M9DT2H30M5S"</td> </tr> <tr> <td>Number of seconds</td> <td>163650605</td> </tr> </table> Example 2: <table> <tr> <td>ISO Duration</td> <td>"P9D"</td> </tr> <tr> <td>Number of seconds</td> <td>777600</td> </tr> </table> Example 3: <table> <tr> <td>ISO Duration</td> <td>"PT2.5H3.1M4.5S"</td> </tr> <tr> <td>Number of seconds</td> <td>9190.5</td> </tr> </table> Pre : NA Post : NA Input : iISODuration Output : oNumberOfSeconds oExceptionMessage oExceptionID return : 0 <> 0 - The ISO duration (mandatory) - The number of seconds The last message if any message is found. If more than one message is found, these are present in the oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Succesfully converted to ISO Duration Otherwise	ISO Duration	"P5Y2M9DT2H30M5S"	Number of seconds	163650605	ISO Duration	"P9D"	Number of seconds	777600	ISO Duration	"PT2.5H3.1M4.5S"	Number of seconds	9190.5
ISO Duration	"P5Y2M9DT2H30M5S"												
Number of seconds	163650605												
ISO Duration	"P9D"												
Number of seconds	777600												
ISO Duration	"PT2.5H3.1M4.5S"												
Number of seconds	9190.5												

Common.ConvertLeadTimeToISODuration

DLL:	tcextcomapi
	This function is available from KB2275138 .
Syntax:	<pre> long Common.ConvertLeadTimeToISODuration(long iYears, long iMonths, domain tcmcs.double iDays, domain tcmcs.double iHours, domain tcmcs.double iMinutes, domain tcmcs.double iSeconds, ref domain tcmcs.st24 oISODuration, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: The purpose of this function is to convert a specified number of years, months, days, hours, minutes and/or seconds to an ISO duration in the format PnYnMnDTnHnMnS or -PnYnMnDTnHnMnS in case of a negative duration. A duration is negative if the specified values have a negative value. A combination of negative values and positive values is not allowed. A duration is positive if the values are 0 or greater than zero. Years and months are without decimals. Days, hours, minutes and seconds can be specified with decimals. <pre> Pre : NA Post : NA Input : iYears - Years iMonths - Months iDays - Days iHours - Hours iMinutes - Minutes iSeconds - Seconds Output : oISODuration - The ISO Duration oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. return : 0 Successfully converted to ISO Duration <> 0 Otherwise </pre>

Common.ConvertPrice

DLL:	tcextcomapi																																							
Syntax:	<pre>long Common.ConvertPrice(domain tcitem iItem, domain tccitg iItemGroup, domain tccuni iInventoryUnit, domain tccuni iFromUnit, domain tcconv iConversionFactorBetweenFromUnitAndInventoryUnit, domain tccuni iToUnit, domain tcconv iConversionFactorBetweenInventoryUnitAndToUnit, domain tcpric iPriceInFromUnit, ref domain tcpric oPriceInToUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>																																							
Usage:	Expl: This method converts a price expressed in "iFromUnit" to a price expressed in "iToUnit". Input: If the "iInventoryUnit" or one of the conversion factors (iConversionFactorBetweenFromUnitAndInventoryUnit / iConversionFactorBetweenInventoryUnitAndToUnit) are not filled the "iItem" (and possibly "iItemGroup") must be filled so the function can search for this information by itself. <table><tr><td>iItem</td><td>Item Code</td><td></td></tr><tr><td>iItemGroup</td><td>Item Group</td><td></td></tr><tr><td>iInventoryUnit</td><td>Inventory Unit</td><td></td></tr><tr><td>iFromUnit</td><td>From Unit</td><td>Mandatory</td></tr><tr><td>iConversionFactorBetweenFromUnitAndInventoryUnit</td><td>Conversion Factor between "iFromUnit" and "iInventoryUnit".</td><td></td></tr><tr><td>iToUnit</td><td>To Unit</td><td>Mandatory</td></tr><tr><td>iConversionFactorBetweenInventoryUnitAndToUnit</td><td>Conversion factor between "iInventoryUnit" and "iToUnit".</td><td></td></tr><tr><td>iPriceInFromUnit</td><td>Price in "iFromUnit"</td><td></td></tr><tr><td>Output: oPriceInToUnit</td><td>Price in "iToUnit"</td><td></td></tr><tr><td>oExceptionMessage</td><td>The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.</td><td></td></tr><tr><td>oExceptionID</td><td>An ID that refers to all error information. Use the functions in Exception to get all relevant information.</td><td></td></tr><tr><td>Return: 0</td><td>Succesfully Converted Price</td><td></td></tr><tr><td><> 0</td><td>Otherwise</td><td></td></tr></table>	iItem	Item Code		iItemGroup	Item Group		iInventoryUnit	Inventory Unit		iFromUnit	From Unit	Mandatory	iConversionFactorBetweenFromUnitAndInventoryUnit	Conversion Factor between "iFromUnit" and "iInventoryUnit".		iToUnit	To Unit	Mandatory	iConversionFactorBetweenInventoryUnitAndToUnit	Conversion factor between "iInventoryUnit" and "iToUnit".		iPriceInFromUnit	Price in "iFromUnit"		Output: oPriceInToUnit	Price in "iToUnit"		oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.		oExceptionID	An ID that refers to all error information. Use the functions in Exception to get all relevant information.		Return: 0	Succesfully Converted Price		<> 0	Otherwise	
iItem	Item Code																																							
iItemGroup	Item Group																																							
iInventoryUnit	Inventory Unit																																							
iFromUnit	From Unit	Mandatory																																						
iConversionFactorBetweenFromUnitAndInventoryUnit	Conversion Factor between "iFromUnit" and "iInventoryUnit".																																							
iToUnit	To Unit	Mandatory																																						
iConversionFactorBetweenInventoryUnitAndToUnit	Conversion factor between "iInventoryUnit" and "iToUnit".																																							
iPriceInFromUnit	Price in "iFromUnit"																																							
Output: oPriceInToUnit	Price in "iToUnit"																																							
oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.																																							
oExceptionID	An ID that refers to all error information. Use the functions in Exception to get all relevant information.																																							
Return: 0	Succesfully Converted Price																																							
<> 0	Otherwise																																							

Common.ConvertQuantity

DLL:	tcextcomapi
Syntax:	<pre> long Common.ConvertQuantity(</pre>

	<pre> domain tcitem iItem, domain tccuni iFromUnit, domain tcqstl iFromQuantity, domain tccuni iToUnit, ref domain tcqstl oToQuantity, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function converts a quantity specified in a unit to a quantity specified in another unit. Rounding will not be done. Common.RoundQuantity can be used for rounding. Post: None Input: iItem - Item iFromUnit - From Unit: Mandatory iFromQuantity - From Quantity iToUnit - To Unit: Mandatory Output: oToQuantity - To Quantity oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Quantity is converted. <> 0 - Otherwise.

Common.ConvertTime

DLL:	tcextcomapi
Syntax:	<pre> long Common.ConvertTime(domain tccuni iFromUnit, domain tcqstl iFromTime, domain tccuni iToUnit, ref domain tcqstl oToTime, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function converts a time specified in a unit to a time specified in another unit. Rounding will not be done. Common.RoundQuantity can be used for rounding. Post: None Input: iFromUnit - From Unit: Mandatory iFromTime - From Time iToUnit - To Unit: Mandatory Output: oToTime - To Time oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in

	Exception to get all relevant information. Return: 0 - Time is converted. <> 0 - Otherwise.

Common.GenerateFirstFreeNumber

DLL:	tctxmcsapi																
Syntax:	<pre>long Common.GenerateFirstFreeNumber(domain tcseri iSeries, domain tcnrgr iNumberGroup, domain tcmcs.byte iMaximumLength, ref domain tcorno oFirstFreeNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>																
Usage:	Expl.: This function generates a new first free number based upon the series, number group and the first free number length. The function checks the entered series. Blocked series and series being full are rejected. Example: If the iSeries is "SLS" and the iMaximumLength = 7 the oFirstFreeNumber will be "SLS9999". Pre: Transaction handling must be done when no caching is used. Post: Transaction handling must be done when no caching is used. Input: <table> <tr> <td>iSeries</td><td>- The series for which the first free number must be generated.</td></tr> <tr> <td>iNumberGroup</td><td>- The number group (parameters) for which the first free number must be generated.(mandatory)</td></tr> <tr> <td>iMaximumLength</td><td>- This is the total length of the output, Inclusive used series. iMaximumLength > 0 and iMaximumLength =< domain length of domain tcorno (standard = 9). (mandatory)</td></tr> </table> Output: <table> <tr> <td>oFirstFreeNumber</td><td>- The first free number based on the series, numbergroup and Maximum length</td></tr> <tr> <td>oExceptionMessage</td><td>- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID</td></tr> <tr> <td>oExceptionID</td><td>- An ID that refers to all error information. Use the functions in Exception to get all relevant information.</td></tr> </table> Return values: <table> <tr> <td>0</td><td>- First Free Number is generated</td></tr> <tr> <td><> 0</td><td>- on errors</td></tr> </table>	iSeries	- The series for which the first free number must be generated.	iNumberGroup	- The number group (parameters) for which the first free number must be generated.(mandatory)	iMaximumLength	- This is the total length of the output, Inclusive used series. iMaximumLength > 0 and iMaximumLength =< domain length of domain tcorno (standard = 9). (mandatory)	oFirstFreeNumber	- The first free number based on the series, numbergroup and Maximum length	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.	0	- First Free Number is generated	<> 0	- on errors
iSeries	- The series for which the first free number must be generated.																
iNumberGroup	- The number group (parameters) for which the first free number must be generated.(mandatory)																
iMaximumLength	- This is the total length of the output, Inclusive used series. iMaximumLength > 0 and iMaximumLength =< domain length of domain tcorno (standard = 9). (mandatory)																
oFirstFreeNumber	- The first free number based on the series, numbergroup and Maximum length																
oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID																
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.																
0	- First Free Number is generated																
<> 0	- on errors																

Common.GetAdditionalEntityInfo

DLL:	tcextemmap
Syntax:	<pre> long Common.GetAdditionalEntityInfo(domain tcomm.enty iEntityType, domain tcomm.enio iEntity, ref domain tcncmp oOperationalCompany, ref domain tcscite oSite, ref domain tcomm.grid oEnterpriseUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl : This function determines the Operational Company, Site and Enterprise Unit for a department, warehouse or project Entity Type/Entity combination. Pre : - Post : - Input : iEntityType - Entity type (Mandatory). Entity types 'Not Applicable' and 'Contract' are not valid. iEntity - Entity (Mandatory)- a department, warehouse or project code. Output : oOperationalCompany - Operational Company oSite - Site, for entity type 'Project' Site is not applicable and returns always empty. oEnterpriseUnit - Enterprise Unit oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return values: 0 - Additional entity information read <> 0 - Otherwise

Common.GetConversionFactor

DLL:	tcextcomapi
Syntax:	<pre> long Common.GetConversionFactor(domain tcitem iItem, domain tccitg iItemGroup, domain tccuni iFromUnit, domain tccuni iToUnit, ref domain tcconv oConversionFactor, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function returns the conversion factor between the given units. The item argument allows for a search on item level. The Item, Item Group and To Unit arguments will all speed up the search if entered.

Pre:	-
Post:	-
Input:	- iItem - Item (Optional). - iItemGroup - Item Group (Optional). - iFromUnit - From Unit (Mandatory if iItem is empty). - iToUnit - To Unit (Mandatory).
Output:	- oConversionFactor - Conversion Factor. - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information.
Return:	0 - Conversion Factor successfully retrieved. <> 0 - Otherwise.

Common.GetCurrencyRates

DLL:	tcextemmap
Syntax:	<pre>long Common.GetCurrencyRates(domain tcncmp iFinancialCompany, domain tcccur iCurrency, domain tcdate iRateDateUTC, domain tcrtyp iExchangeRateType, ref domain tcratc oRates(), ref domain tcratf oRateFactors(), ref domain tccrnd oRoundingFactors(), ref domain tcyesno oExpressInBaseCurrency(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Get the rate data between a given transaction currency and the home currencies. The home currencies are defined in session "Companies" (tceemm1170m000). To determine the rate data between two currencies a date and an exchange rate type is needed. So these two values also need to be provided. Depending on the number of home currencies (with a maximum of three) the rate data is filled. The rate data consists of the following: - rates - rate factors - rounding factors - express in base currency Pre: Currency system needs to be setup in Companies (tceemm1170m000) Rates must be defined between the currency and home currencies.

	Post: - Input: - iFinancialCompany - Financial Company: Mandatory - iCurrency - Transaction currency for which the rates to the home currencies have to be retrieved: Mandatory - iRateDateUTC - UTC date for which the rates have to be retrieved: Mandatory - iExchangeRateType - Exchange rate type: Mandatory Output: - oRates - Array of rates. - oRateFactors - Array of rate factors. - oRoundingFactors - Array of rounding factors. - oExpressInBaseCurrency - Array of flags (tcyesno) which indicate if the currency is expressed in the currency base. - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Currency rates are found. <> 0 - Otherwise.

Common.GetFinancialCompanyOfEntity

DLL:	tcextemapi
Syntax:	<pre>long Common.GetFinancialCompanyOfEntity(domain tcncmp iLogisticCompany, domain tcomm.enty iEntityType, domain tcomm.enio iEntity, ref domain tcncmp oFinancialCompany, ref domain tmcsc.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function gets the financial company of the logistic company linked to entity. Pre: Post: Input: - i.logistic.company - Logistic Company: Mandatory - i.entity.type - Entity Type: Mandatory - i.entity - Entity: Mandatory Output: - oFinancialCompany - Financial company of the enterprise unit linked to the entity. - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - oExceptionID - An ID that refers to all error information. Use the functions in

	Return: 0 <> 0 Exception to get all relevant information. - Financial company is found. - Otherwise.

Common.GetFormattedAddress

DLL:	tcextcomapi
Syntax:	<pre> long Common.GetFormattedAddress(domain tccom.cadr iAddress, long iNumberOfLines, domain tcccty iFromCountry, ref string oFormattedAddress(,) fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This functions formats an given addresscode into a formatted address array of a given number of lines. Pre: iAddress must exists in tccoml30 Multibyte Array used for oFormattedAddress must be declared and initialized. Post: NA Input: - iAddress - The address code for which the formatted address must be generated. (mandatory) - iNumberOfLines - The maximum number of lines of the formatted address.(mandatory) - iFromCountry - When iFromCountry is filled, the country information will only be used in the formatted address when it differs from iFromCountry. When left empty always the country information is used in the formatted address. *Example if i.from.country = "NLD" the Dutch addresses will not have country information in the formatted address. This can be used when a company located in "NLD" sends invoices or mailings to addresses in "NLD". <pre> Moonen Shipyards BV Kade 35 1780 AA Den Helder The addresses outside "NLD" will have country information in the formatted address: Moonen F&A&T-Indertechnik GmbH Balhorner Feld 1 33106 Paderborn Deutschland if i.from.country = empty both addresses will have country information in the formatted address: </pre>

	Moonen Shipyards BV Kade 35 1780 AA Den Helder Nederland Moonen Fördertechnik GmbH Balhorner Feld 1 33106 Paderborn Deutschland
	Output: oFormattedAddress - Array with formatted address lines. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return values: 0 - Formatted address is returned <> 0 - on errors

Common.GetISOCodeOfLNLanguage

DLL:	tcextmcsapi
Syntax:	<pre>long Common.GetISOCodeOfLNLanguage(domain tcclan iLNLanguage, boolean iForceReading, ref domain tcilng oISOLanguage, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl : This function gets the ISO code of the LN language code. Pre : - Post : - Input : iLNLanguage - The LN language: Mandatory iForceReading - When false, the function uses cache if possible. Output : oISOLanguage - The ISO language code. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return values: 0 - ISO code read <> 0 - on Errors

Common.GetParameters

DLL:	tcextcomapi
Syntax:	<pre> long Common.GetParameters(domain tcncmp iCompany, domain tctabl.c iTableCode, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function reads one or more parameters from a standard LN parameter table. Note that this function uses variable arguments, for input and output. Data is retrieved via name value pairs: the field mnemonic and the field value. Several parameter-fields are stored as arrays. Individual elements of these parameter-fields can be retrieved by adding the element number to the field-mnemonic. Example: "qipo(28)", "mvst(2)". Example: When the 'Minimal Picks' and 'Quarantine Inventory Available for Planning by Origin' are needed from the Inventory Handling parameters, the call of this public interface is as follows: <pre> if Common.GetParameters(/* Fixed arguments: company, --> input "whinh000", --> input exception.message, --> output exception.id, --> output /* Variable arguments: "minp", --> input minimal.picks, --> output "qipo(1)", --> input l.qipo(1), --> output "qipo(2)", --> input l.qipo(2), --> output "qipo(3)", --> input l.qipo(3) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif </pre> Pre: None Post: None Input: - iCompany - Company: Mandatory Note: for reading "tccom000" always the current company is taken. - iTableCode - Code of the parameter table: Mandatory. Example: "tpctm000" ... - The field mnemonic of the required parameters. Output: - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required parameters.

	Return: 0 - Parameter(s) read. <> 0 - Error occurred.

Common.RoundAmount

DLL:	ttextcomapi
Syntax:	<pre> long Common.RoundAmount(double iUnroundedAmount, domain tcmcs.st14 iDomainName, domain tcccur iCurrency, ref double oRoundedAmount, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This functions rounds an amount with a given domain and currency. Pre: NA Post: NA Input : iUnroundedAmount - The amount that must be rounded. iDomainName - The domain name. (mandatory) iCurrency - The currency. (mandatory) Output: oRoundedAmount - The rounded amount. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return values: 0 - Rounded amount is returned <> 0 - on errors

Common.RoundQuantity

DLL:	ttextcomapi
Syntax:	<pre> long Common.RoundQuantity(double iUnroundedQuantity, domain tcmcs.st14 iDomainName, domain tccuni iUnit, ref double oRoundedQuantity, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function rounds a quantity depending on given domain and unit. Pre: NA Post: NA Input : iUnroundedQuantity - The quantity that must be rounded.

	iDomainName - The domain name. (mandatory) iUnit - The unit. (mandatory) Output: oRoundedQuantity - The rounded quantity. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return values: 0 - Rounded quantity is returned <> 0 - on errors

Common.UpdateApprovedConversionFactor

DLL:	tcextcomapi
Syntax:	<pre>long Common.UpdateApprovedConversionFactor(domain tcitem iItem, domain tccity iItemGroup, domain tccuni iFromUnit, domain tccuni iToUnit, domain tcconv iConversionFactor, domain tcrpow iConversionFactorExponent, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This Public Interface updates an approved conversion factor. The given Item cannot have Inventory, a Package Definition, Requisition Lines or a Quotation for the Conversion Factor to be updated. The given Item Group cannot have Inventory for the Conversion Factor to be changed. As generic Conversion Factors cannot be changed, either iItem, iItemGroup or both need to be filled. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iItem Item Code (Mandatory if iItemGroup is empty). iItemGroup Item Group (Mandatory if iItem is empty). iFromUnit From Unit (Mandatory). iToUnit To Unit (Mandatory). iConversionFactor New Conversion Factor between "iFromUnit" and "iToUnit". Should be positive (Mandatory). iConversionFactorExponent New Conversion Factor exponent. Output: oExceptionMessage The last message if the return

	value is not equal to 0. If more than one message is given, these are present in the oExceptionID. An ID that refers to all error information. Use the functions in Exception to get all relevant information. Successfully updated the Conversion Factor. Otherwise.
oExceptionID	
Return: 0	
<> 0	

Common.ValidateNumberGroupSeries

DLL:	tcextmcsapi
Syntax:	<pre>long Common.ValidateNumberGroupSeries(domain tcseri iSeries, domain tcnrgr iNumberGroup, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl.: This function checks if a series has been defined for the specified number group. Pre: - Post: - Input: iSeries - The series. iNumberGroup - The number group (mandatory). Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return values: 0 If the order series is defined and not blocked. 1 If the order series is not defined 2 If the order series is blocked. 3 If the series length does not correspond to number group 4 If the series for a numbergroup is full.

Public Interfaces for Contact

The following functions are available:

[Contact.StartOverview](#)

Contact.StartOverview

DLL:	tcextcomapi
	This function is available from KB2277108 .
Syntax:	<pre> long Contact.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccom.ccnt iContact, domain tcseak iSearchKey mb, ref domain tccom.ccnt oContact, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session Contacts - tccom4540m000 (Overview). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "" (Session has no start filter) iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend Optional A specific query to be used when zooming to this session. iContact - Contact Not Mandatory iSearchKey - Search Key Not Mandatory Output: for iStartMode MODAL: oContact - Selected Contact oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Otherwise.

Public Interfaces for BusinessPartner

The following functions are available:

[BusinessPartner.CheckOverdueInvoices](#)

[BusinessPartner.GetBuyFromData](#)

[BusinessPartner.GetCreditLimitAndBalances](#)

[BusinessPartner.GetGeneralData](#)

[BusinessPartner.GetInvoiceFromData](#)

[BusinessPartner.GetInvoiceToData](#)

[BusinessPartner.GetPayByData](#)

[BusinessPartner.GetPayToData](#)

[BusinessPartner.GetShipFromData](#)

[BusinessPartner.GetShipToData](#)

[BusinessPartner.GetSoldToData](#)

[BusinessPartner.StartEasyEntry](#)

[BusinessPartner.StartMultiMain](#)

[BusinessPartner.StartOverview](#)

BusinessPartner.CheckOverdueInvoices

DLL:	tcextcomapi
Syntax:	<pre> long BusinessPartner.CheckOverdueInvoices (domain tcncmp iFinancialCompany, domain tccom.bpid iInvoiceToBusinessPartner, domain tccwoc iDepartment, domain tcyesno iIncludeAnticipatedReceipts, domain tcdate iDueDate, ref domain tcyesno oOverdueAmountExceedsTolerance, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl. :

	This function checks if the business partner has overdue invoices. If so, this function will compare the total overdue amount with the tolerance amount (tccom112.tamd) or percentage of the credit limit (tccom112.tpcd). If the overdue amount is above the tolerance, then o.overdue.amout.exceeds.tolerance becomes tcyesno.yes When both fields (tccom112.tamd and tccom112.tpcd) are 0, then it has to give the signal every time the due amount is greater than 0. If the user wants to avoid the signal, the percentage has to be filled with 100% and then the full credit limit will be used. This of course requires that the credit limit has been set. Pre : NA Post : NA Input : iFinancialCompany - financial company number (mandatory) iInvoiceToBusinessPartner - invoice-to business partner (mandatory) iDepartment - department (optional) iIncludeAnticipatedReceipts - anticipated receipts yes/no (mandatory) iDueDate - date till which overdue invoices are calculated (mandatory) Output : oOverdueAmountExceedsTolerance - 'Yes' when total overdue amount exceeds tolerance else 'No'. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Total overdue amount determined. <> 0 - Error occurred.

BusinessPartner.GetBuyFromData

DLL:	tcextcomapi
Syntax:	<pre>long BusinessPartner.GetBuyFromData(domain tcncomp iLogisticCompany, domain tccom.bpid iBuyFromBusinessPartner, domain tccwoc iDepartment, boolean iForceRead, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...)</pre>
Usage:	Expl: This function retrieves data from Buy-from Business Partners. Individual fields are retrieved, multiple fields can be retrieved in one call. Data is read from general level or department level. Depending on the input argument iDepartment Note: this function uses variable arguments, for input and

	<pre> output. Data is retrieved via value pairs: specify the field and the field value. Example: if BusinessPartner.GetBuyFromData(* Fixed arguments: 575, --> input "RLX000002", --> input "", --> input true, --> input oExceptionMessage, --> output oExceptionID, --> output * Variable arguments: "cadr", --> input 1.tccom120.cadr, --> output "ccnt", --> input 1.tccom120.ccnt) <> 0 then --> output * Error, do something Exception.Delete(exception.id) Pre: none Post: none Input: iLogisticCompany - Logistic Company (mandatory) iBuyFromBusinessPartner - Buy From Business Partner (mandatory) iDepartment - Department iForceRead - True/False: Option to force new query instead of using cached information ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - An error occurred </pre>

BusinessPartner.GetCreditLimitAndBalances

DLL:	ttextcomapi
Syntax:	<pre> long BusinessPartner.GetCreditLimitAndBalances(domain tccom.bpid iInvoiceToBusinessPartner, domain tccwoc iDepartment, domain tcyesno iIncludeOpenOrdersInFinancialRisk, ref domain tcamnt oCreditLimit, ref domain tcamnt oOrderBalance, ref domain tcamnt oInvoiceBalance, ref domain tcamnt oBillingRequestBalance, </pre>

	<pre> ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl. : This function calculates the credit limit and the balances of an invoice-to business partner. Pre : NA Post : NA Input : iInvoiceToBusinessPartner iDepartment - invoice-to business partner - mandatory iIncludeOpenOrdersInFinancialRisk - department for checking credit limit (optional) - Include open orders in financial risk y/n (tcmcs064.ioso) - mandatory Output : oCreditLimit - total credit limit oOrderBalance - open order balance oInvoiceBalance - open invoice balance oBillingReqBalance - open billing request balance Return: 0 - Credit Limit and Balances determined successfully. <> 0 - Error occurred. </pre>

BusinessPartner.GetGeneralData

DLL:	tcextcomapi
Syntax:	<pre> long BusinessPartner.GetGeneralData(domain tcncmp iCompany, domain tccom.bpid iBusinessPartner, boolean iForceRead, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	<pre> Expl: This function retrieves data from Business Partners. Individual fields are retrieved, multiple fields can be retrieved in one call. Note: this function uses variable arguments, for input and output. Data is retrieved via value pairs: specify the field and the field value. Example: if BusinessPartner.GetGeneralData(/* Fixed arguments: 575, --> input "RLX000002", --> input true, --> input oExceptionMessage, --> output oExceptionID, --> output /* Variable arguments: "cadr", --> input 1.tccom100.cadr, --> output "ccnt", --> input 1.tccom100.ccnt) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) Pre: none </pre>

	Post: none Input: iCompany iBusinessPartner iForceRead ... Output: oExceptionMessage oExceptionID ... Return: 0 <> 0	- Company (mandatory) - Business Partner (mandatory) - True/False: Option to force new query instead of using cached information - The field mnemonic of the required field. - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - The value of the required field. - Data read - An error occurred

BusinessPartner.GetInvoiceFromData

DLL:	ttextcomapi	
Syntax:	<pre> long BusinessPartner.GetInvoiceFromData(domain tcncmp iFinancialCompany, domain tccom.bpid iInvoiceFromBusinessPartner, domain tccwoc iDepartment, boolean iForceRead, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>	
Usage:	Expl: This function retrieves data from Invoice-from Business Partners. Individual fields are retrieved, multiple fields can be retrieved in one call. Data is read from general level or department level. Depending on the input argument iDepartment Note: this function uses variable arguments, for input and output. Data is retrieved via value pairs: specify the field and the field value. Example: <pre> if BusinessPartner.GetInvoiceFromData(/* Fixed arguments: 575, --> input "RLX000002", --> input "", --> input true, --> input oExceptionMessage, --> output oExceptionID, --> output /* Variable arguments: "cadr", --> input 1.tccom122.cadr, --> output </pre>	

	<pre> "ccnt", 1.tccom122.ccnt) <> 0 then * Error, do something Exception.Delete(exception.id) Pre: none Post: none Input: iFinancialCompany - Financial Company (mandatory) iInvoiceFromBusinessPartner iDepartment - Invoice from Business Partner (mandatory) iForceRead - Department ... - True/False: Option to force new query instead of using cached information - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - An error occurred </pre>

BusinessPartner.GetInvoiceToData

DLL:	tcextcomapi
Syntax:	<pre> long BusinessPartner.GetInvoiceToData(domain tcncomp iFinancialCompany, domain tccom.bpid iInvoiceToBusinessPartner, domain tccwoc iDepartment, boolean iForceRead, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves data from Invoice-to Business Partners. Individual fields are retrieved, multiple fields can be retrieved in one call. Data is read from general level or department level. Depending on the input argument iDepartment Note: this function uses variable arguments, for input and output. Data is retrieved via value pairs: specify the field and the field value. Example: <pre> if BusinessPartner.GetInvoiceToData(* Fixed arguments: 575, --> input </pre>

	<pre> "RLX000002", --> input "", --> input true, --> input oExceptionMessage, --> output oExceptionID, --> output /* Variable arguments: "cadr", --> input 1.tccom112.cadr, --> output "ccnt", --> input 1.tccom112.ccnt) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) Pre: none Post: none Input: iFinancialCompany - Financial Company (mandatory) iInvoiceToBusinessPartner - Invoice to Business Partner (mandatory) iDepartment - Department iForceRead - True/False: Option to force new query instead of using cached information ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - An error occurred </pre>

BusinessPartner.GetPayByData

DLL:	ttextcomapi
Syntax:	<pre> long BusinessPartner.GetPayByData(domain tcncmp iFinancialCompany, domain tccom.bpid iPayByBusinessPartner, domain tccwoc iDepartment, boolean ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves data from Pay-by Business Partners. Individual fields are retrieved, multiple fields can be retrieved in one call. Data is read from general level or department level. Depending on the input argument iDepartment

	Note: this function uses variable arguments, for input and output. Data is retrieved via value pairs: specify the field and the field value. Example: <pre> if BusinessPartner.GetPayByData(* Fixed arguments: 575, --> input "RLX000002", --> input "", --> input true, --> input oExceptionMessage, --> output oExceptionID, --> output * Variable arguments: "cadr", --> input l.tccom114.cadr, --> output "ccnt", --> input l.tccom114.ccnt) <> 0 then --> output * Error, do something Exception.Delete(exception.id) </pre> Pre: none Post: none Input: iFinancialCompany - Financial Company (mandatory) iPayByBusinessPartner - Pay-by Business Partner (mandatory) iDepartment - Department iForceRead - True/False: Option to force new query instead of using cached information ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - An error occurred

BusinessPartner.GetPayToData

DLL:	tcextcomapi
Syntax:	<pre> long BusinessPartner.GetPayToData(domain tcncmp iFinancialCompany, domain tccom.bpid iPayToBusinessPartner, domain tccwoc iDepartment, boolean iForceRead, ref domain tcmcs.s999m oExceptionMessage mb, </pre>

	ref	long		oExceptionID, ...)	
Usage:	Expl:	This function retrieves data from Pay-to Business Partners. Individual fields are retrieved, multiple fields can be retrieved in one call. Data is read from general level or department level. Depending on the input argument iDepartment Note: this function uses variable arguments, for input and output. Data is retrieved via value pairs: specify the field and the field value. Example: if BusinessPartner.GetPayToData(/* Fixed arguments: 575,			

BusinessPartner.GetShipFromData

DLL:	tcextcomapi
Syntax:	<pre> long BusinessPartner.GetShipFromData(domain tcncmp iLogisticCompany, domain tccom.bpid iShipFromBusinessPartner, domain tccom.bpid iBuyFromBusinessPartner, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves data from Ship-from Business Partners. Individual fields are retrieved, multiple fields can be retrieved in one call. Data is read from general level or site level. Depending on the input argument iSite Note: this function uses variable arguments, for input and output. Data is retrieved via value pairs: specify the field and the field value. Example: <pre> if BusinessPartner.GetShipFromData(/* Fixed arguments: 575, --> input "RLX000002", --> input "", --> input "", --> input true, --> input false --> input oExceptionMessage, --> output oExceptionID, --> output /* Variable arguments: "cadr", --> input 1.tccom121.cadr, --> output "ccnt", --> input 1.tccom121.ccnt) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) </pre> Pre: none Post: none Input: - iLogisticCompany - Logistic Company (mandatory) - iShipFromBusinessPartner - Ship-from Business Partner (mandatory) - iBuyFromBusinessPartner - Buy-from Business Partner This input field is only used for reading the fields: "Delivery Terms"(cdec), "Point of Title Passage"(ptpa) and "Return Delivery Terms"(rdec). If those are empty for the given Ship-from Business Partner the system will try to retrieve the data from the Buy-from Business Partner - iSite - Site - iForceRead - True/False: Option to force new query instead of using cached information

	<pre> iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... Return: 0 - The value of the required field. <> 0 - Data read - An error occurred </pre>

BusinessPartner.GetShipToData

DLL:	tcextcomapi
Syntax:	<pre> long BusinessPartner.GetShipToData(domain tcncmp iLogisticCompany, domain tccom.bpid iShipToBusinessPartner, domain tccom.bpid iSoldToBusinessPartner, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves data from Ship-to Business Partners. Individual fields are retrieved, multiple fields can be retrieved in one call. Data is read from general level or site level. Depending on the input argument iSite Note: this function uses variable arguments, for input and output. Data is retrieved via value pairs: specify the field and the field value. Example: <pre> if BusinessPartner.GetShipToData(/* Fixed arguments: 575, --> input "RLX000002", --> input "", --> input "", --> input true, --> input false, --> input oExceptionMessage, --> output oExceptionID, --> output /* Variable arguments: </pre>

	<pre> "cdr", l.tccom111.cdr, "ccnt", l.tccom111.ccnt) <> 0 then * Error, do something Exception.Delete(exception.id) Pre: none Post: none Input: iLogisticCompany - Logistic Company (mandatory) iShipToBusinessPartner - Ship-to Business Partner (mandatory) iSoldToBusinessPartner - Sold-to Business Partner This input field is only used for reading the fields: "Automatically Process Sales Schedule Releases"(apsr), "Delivery Terms"(cdec), "Point of Title Passage"(ptpa) and "Return Delivery Terms"(rdec). If those are empty for the given Ship-to Business Partner the system will try to retrieve the data' from the Sold-to business partner. iSite - Site iForceRead - True/False: Option to force new query instead of using cached information iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. Only used for "Site" ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - An error occurred </pre>

BusinessPartner.GetSoldToData

DLL:	tcextcomapi
Syntax:	<pre> long BusinessPartner.GetSoldToData(domain tcncmp iLogisticCompany, domain tccom.bpid iSoldToBusinessPartner, domain tccwoc iDepartment, boolean iForceRead, </pre>

	<pre> ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves data from Sold-to Business Partners. Individual fields are retrieved, multiple fields can be retrieved in one call. Data is read from general level or department level. Depending on the input argument iDepartment Note: this function uses variable arguments, for input and output. Data is retrieved via value pairs: specify the field and the field value. Example: <pre> if BusinessPartner.GetSoldToData(* Fixed arguments: 575, --> input "RLX000002", --> input "", --> input true, --> input oExceptionMessage, --> output oExceptionID, --> output * Variable arguments: "cadr", --> input l.tccom110.cadr, --> output "ccnt", --> input l.tccom110.ccnt) <> 0 then --> output * Error, do something Exception.Delete(exception.id) </pre> Pre: none Post: none Input: - iLogisticCompany - Logistic Company (mandatory) - iSoldToBusinessPartner - Sold to Business Partner (mandatory) - iDepartment - Department - iForceRead - True/False: Option to force new query instead of using cached information ... - The field mnemonic of the required field. Output: - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - An error occurred

BusinessPartner.StartEasyEntry

DLL:	tcextcomapi
Syntax:	<pre> long BusinessPartner.StartEasyEntry(long iStartMode, domain tccom.bpid iBusinessPartnerSerie, domain tcnama iBusinessPartnerName mb, domain tcsern iBusinessPartnerDefaults, domain tcbprl iBusinessPartnerRole, domain tcyesno iCreateSoldToRole, domain tcyesno iCreateBuyFromRole, domain tcyesno iCreateShipToRole, domain tcyesno iCreateShipFromRole, domain tcyesno iCreateInvoiceToRole, domain tcyesno iCreateInvoiceFromRole, domain tcyesno iCreatePayByRole, domain tcyesno iCreatePayToRole, domain tccom.cadr iAddressCode, domain tcnama iAddressName mb, domain tcccty iCountry, domain tcpstc iZipCode mb, domain tcmcs.cste iStateProvince, domain tcnama iCounty mb, domain tccity iCity, domain tcnamc iStreet mb, domain tccom.nmbr iHouseNumber mb, domain tctax.txnb iTaxNumber, domain tccom.ccnt iContact, domain tcmcs.st80m iContactName mb, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Easy Entry Business Partners - (tccom4201m000). Input: iStartMode Specifies the start mode for the session. Possible value is: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. (mandatory) iBusinessPartnerSerie - Business Partner Serie (mandatory) iBusinessPartnerName - Business Partner Name (mandatory) iBusinessPartnerDefaults - Business Partner Defaults (mandatory) iBusinessPartnerRole - Business Partner Role Note: The session Easy Entry Business Partners will not always use all iCreate<*>Role input field. This depends on the company type eg: In a logistic company the financial roles will not be created iCreateSoldToRole - Create Sold To Role (customer/logistic) iCreateBuyFromRole - Create Buy From Role (supplier/logistic) iCreateShipToRole - Create Ship To Role (customer/logistic) iCreateShipFromRole - Create Ship From Role (supplier/logistic) iCreateInvoiceToRole - Create Invoice To Role (customer/financial) iCreateInvoiceFromRole - Create Invoice From Role (supplier/financial)

	<code>iCreatePayByRole</code> - Create Pay By Role (customer/financial) <code>iCreatePayToRole</code> - Create Pay To Role (supplier/financial) <code>iAddressCode</code> - Address Code or address series Note: if <code>iAddressCode</code> is an existing address instead of a series code, the values of the address related input fields are ignored. <code>iAddressName</code> - Address Name <code>iCountry</code> - Country <code>iZipCode</code> - Zip Code/ Postal code <code>iStateProvince</code> - State or Province <code>iCounty</code> - County <code>iCity</code> - City <code>iStreet</code> - Street <code>iHouseNumber</code> - House number <code>iTaxNumber</code> - Tax Number <code>iContact</code> - Contact or contact serie Note: if <code>iContact</code> is an existing Contact instead of a series code, the value of the <code>iContactName</code> is ignored. <code>iContactName</code> - Full Contact Name Output: <code>oExceptionMessage</code> - The last message if the return value is not equal to 0. If more than one message is given, these are present in the <code>oExceptionID</code> <code>oExceptionID</code> - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Otherwise.

BusinessPartner.StartMultiMain

DLL:	ttextcomapi
Syntax:	<pre>long BusinessPartner.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tccom.bpid iBusinessPartner, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the details session Business Partners - (tccom4100s000). Input: <code>iStartMode</code> Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel

	sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "" (Session has no start filter) iQueryExtend Optional A specific query to be used when zooming to this session. iBusinessPartner - Business Partner Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Otherwise.

BusinessPartner.StartOverview

DLL:	tcextcomapi
Syntax:	<pre> long BusinessPartner.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccom.bpid iBusinessPartner, domain tcseak iSearchKey mb, domain tccom.bpid iParentBusinessPartner, domain tccom.cadr iAddress, ref domain tccom.bpid oBusinessPartner, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session Business Partners - (tccom4500m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "" (Session has no start filter) iSessionIndex

	Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend Optional A specific query to be used when zooming to this session. iBusinessPartner - Business Partner Not Mandatory iSearchKey - Search Key Not Mandatory iParentBusinessPartner - Parent BP Not Mandatory iAddress - Address Not Mandatory Output: for iStartMode MODAL: oBusinessPartner- Selected Business Partner oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Otherwise.

Public Interfaces for BuyFromBusinessPartner

The following functions are available:

[BuyFromBusinessPartner.StartOverview](#)

BuyFromBusinessPartner.StartOverview

DLL:	tcextcomapi This function is available from KB2277106.
Syntax:	<pre>long BuyFromBusinessPartner.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccom.bpid iBuyFromBusinessPartner, domain tcseak iSearchKey mb, domain tccom.cadr iAddress, ref domain tccom.bpid oBuyFromBusinessPartner, ref domain tcmcs.s999m oExceptionMessage mb,</pre>

	ref	long	oExceptionID)
Usage:	Expl:	This function starts session Buy-from Business Partners - (tccom4520m000).	
	Input:	iStartMode	Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously.
		iStartFilter	Specifies the start filter that is to be applied to the started session. Possible values are: "" (Session has no start filter)
		iSessionIndex	Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable.
		iQueryExtend	Optional A specific query to be used when zooming to this session.
		iBuyFromBusinessPartner	- Business Partner Not Mandatory
		iSearchKey	- Search Key Not Mandatory
		iAddress	- Address Not Mandatory
	Output:	for iStartMode MODAL:	
		oBuyFromBusinessPartner	- Selected Buy-from Business Partner
		oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
		oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
	Return:	0	- Session started
		<> 0	- Otherwise.

Public Interfaces for InvoiceToBusinessPartner

The following functions are available:

[InvoiceToBusinessPartner.StartDetail](#)

InvoiceToBusinessPartner.StartDetail

DLL:	tcextcomapi
	This function is available from KB3524683 .
Syntax:	<pre> long InvoiceToBusinessPartner.StartDetail(long iStartMode, domain tccom.bpid iInvoiceToBusinessPartner, domain tccwoc iDepartment, boolean iReadOnly, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts Detail session Invoice-to Business Partner-(tccom4112s000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS- Parent and child are parallel sessions that can be manipulated simultaneously. iInvoiceToBusinessPartner - Invoice-to Business Partner (Mandatory) iDepartment - Department (Not Mandatory) iReadOnly - Session is started in readonly-mode when iReadOnly is True (Optional) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Otherwise.

Public Interfaces for HoldReason

The following functions are available:

[HoldReasons.StartOverview](#)

HoldReasons.StartOverview

DLL:	tcextmcsapi
	This function is available from KB2257724 .
Syntax:	<pre> long HoldReasons.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcmcs.hrea iHoldReason, ref domain tcmcs.hrea oHoldReason, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Hold Reasons in overview mode (tcmcs2110m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex The index that will be used. Supported value: 1: sort by Hold Reason iQueryExtend A specific query to be used when zooming to this session. iHoldReason Hold Reason Output: for iStartMode MODAL: oHoldReason Selected Hold Reason oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 Otherwise.

Public Interfaces for BillingCycle

The following functions are available:

[BillingCycle.GetInvoicedate](#)

BillingCycle.GetInvoicedate

DLL:	ttextmcsapi This function is available from KB3519213 .																						
Syntax:	<pre> long BillingCycle.GetInvoicedate(domain tcncmp iCompany, domain tcbicy iBillingCycle, domain tcdate iCutOffDate, domain tcdate iDeliveryDate, domain tcamnt iBillableLineAmount, domain tcccur iBillableLineCurrency, domain tcdate iRateDate, domain tcrtyp iRateType, ref domain tcdate oInvoiceDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>																						
Usage:	Expl: This public interface is used to determine the next invoice date for a billing cycle (including time). The next invoice date is determines using the recurrence from the billing cycle of the billable line. Note: 1) If the input Billing Cycle is empty then the input Cut off Date is returned as the Invoice Date. In this case if Cut off Date is passed as 0 then Invoice Date 0 is returned. 2) If Billing Cycle is filled and if input Cut-off Date is 0 then current date is used. 3) If Delivery date is 0 then current date is used. Pre: preconditions Post: postconditions Input: <table border="0"> <tr><td>iCompany</td><td>- Company Number</td></tr> <tr><td>iBillingCycle</td><td>- Billing Cycle</td></tr> <tr><td>iCutOffDate</td><td>- Current Cut off Date / Invoice Date</td></tr> <tr><td>iDeliveryDate</td><td>- Delivery Date</td></tr> <tr><td>iBillableLineAmount</td><td>- Line Amount of the Billable line</td></tr> <tr><td>iBillableLineCurrency</td><td>- Currency of Billable Line</td></tr> <tr><td>iRateDate</td><td>- Rate Date of the Billable Line</td></tr> <tr><td>iRateType</td><td>- Rate Type of the Billable Line</td></tr> </table> Output: <table border="0"> <tr><td>oInvoiceDate</td><td>- New Invoice Date based on Billing Cycle</td></tr> <tr><td>oExceptionMessage</td><td>- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.</td></tr> <tr><td>oExceptionID</td><td>- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.</td></tr> </table> Return: 0 - New Invoice Date determined successfully <> 0 - Error	iCompany	- Company Number	iBillingCycle	- Billing Cycle	iCutOffDate	- Current Cut off Date / Invoice Date	iDeliveryDate	- Delivery Date	iBillableLineAmount	- Line Amount of the Billable line	iBillableLineCurrency	- Currency of Billable Line	iRateDate	- Rate Date of the Billable Line	iRateType	- Rate Type of the Billable Line	oInvoiceDate	- New Invoice Date based on Billing Cycle	oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.	oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.
iCompany	- Company Number																						
iBillingCycle	- Billing Cycle																						
iCutOffDate	- Current Cut off Date / Invoice Date																						
iDeliveryDate	- Delivery Date																						
iBillableLineAmount	- Line Amount of the Billable line																						
iBillableLineCurrency	- Currency of Billable Line																						
iRateDate	- Rate Date of the Billable Line																						
iRateType	- Rate Type of the Billable Line																						
oInvoiceDate	- New Invoice Date based on Billing Cycle																						
oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.																						
oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.																						

Public Interfaces for Series

The following functions are available:

[Series.StartOverview](#)

Series.StartOverview

DLL:	tcextmcsapi
	This function is available from KB3606156 .
Syntax:	<pre> long Series.StartOverview(long iStartMode, long iSessionIndex, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcnrgr iNumberGroup, ref domain tcseri oSeries, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session 'First Free Numbers' (tcmcs0150m000). Input: iStartMode Specifies the start mode for the session. Possible values are: - MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. - MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iSessionIndex - The index that will be used. Supported value: 1: sort by Number Group, Series. iStartFilter - Not Used. iQueryExtend - A specific query to be used when zooming to this session. iNumberGroup - Number Group. Output: oSeries - Selected Series. (For iStartMode = MODAL) oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - Session started. - Error.

Public Interfaces for Task

The following functions are available:

[Task.StartMultiMain](#)

Task.StartMultiMain

DLL:	tcontextcomapi This function is available from KB3607361 .
Syntax:	<pre> long Task.StartMultiMain(long iStartMode, long iSessionIndex, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tccom.acty iTask, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session 'Task' (tccom6600m300). Input: - iStartMode - Specifies the start mode for the session. - Possible values are: - MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. - MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. - iSessionIndex - The index that will be used. - Supported value: 1: sort by Activity. - iStartFilter - Not Used. - iQueryExtend - A specific query to be used when zooming to this session. - iTask - Task; this is an Activity of type Task. Output: - oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started. <> 0 - Error.

Chapter 4 Public Interfaces for Calendar

Public Interfaces for Calendar

The following functions are available:

[Calendar.CalculateDaysFromHours](#)

[Calendar.GetPeriodCapacity](#)

[Calendar.GetWorkingDays](#)

[Calendar.PlanLeadTimeBackward](#)

[Calendar.PlanLeadTimeForward](#)

Calendar.CalculateDaysFromHours

DLL:	ttextccpapi		
	This function is available from KB3568051 .		
Syntax:	<pre>long Calendar.CalculateDaysFromHours(domain tcccp.ract iAvailabilityType, double iHours, domain tcutcs iEffectiveDate, ref double oDays, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>		
Usage:	Expl: This public interface converts a given number of hours into the corresponding number of days, using the average number of hours per day as the conversion factor. Input: - iAvailabilityType - Availability Type (Mandatory). - iHours - Hours (Mandatory). - iEffectiveDate - Effective Date. Default value is the current date if the value is zero. Output: - oDays - Days - oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID.		

	oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.
	Return: 0	- Calculated the number of days Successfully.
	<> 0	- Error occurred.

Calendar.GetPeriodCapacity

DLL:	tcextccpapi
Syntax:	<pre> long Calendar.GetPeriodCapacity(domain tcncmp iLogisticCompany, domain tcccp.ccal iCalendarCode, domain tcccp.ract iAvailabilityType, domain tczone iTimeZone, domain tcdate iStartDate, domain tcdate iEndDate, ref domain cpcom.cato oCapacity, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Get the available capacity between the given start and end date, for the given logistic company and calendar parameters. In case the optional iTimeZone parameter is not given, the system timezone is assumed. Input: - iLogisticCompany Logistic Company (Mandatory) - iCalendarCode Calendar (Mandatory) - iAvailabilityType Availability Type (Mandatory) - iTimeZone Time zone (Optional) - iStartDate Start Date (Mandatory) - iEndDate End Date (Mandatory) Output: - oCapacity Capacity (in hours) - oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Function is executed successfully <> 0 - An error occurred

Calendar.GetWorkingDays

DLL:	tcextccpapi		
	This function is available from KB2297011 .		
Syntax:	<pre> long Calendar.GetWorkingDays(domain tcccp.ccal iCalendarCode, domain tcccp.ract iAvailabilityType, domain tczone iTimeZone, domain tcdate iStartDate, domain tcdate iEndDate, ref long oNumberOfWorkingDays, ref domain tcdate oWorkingDays(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>		
Usage:	Expl: This function retrieves the working days in the given interval from the actual calendar. Input: iCalendarCode - Calendar Code, Mandatory iAvailabilityType - Availability Type, Mandatory iTimeZone - Time Zone, Optional If empty, It gets the system time zone. This is usually the time zone of the user, as defined in User Data Template (ttams1110m000). iStartDate - Interval Start Date, Mandatory Must exist in the calendar. iEndDate - Interval End Date, Mandatory Must exist in the calendar. Output: oNumberOfWorkingDays - Number of days in the Working Days array. oWorkingDays - Array of working days. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Function is executed successfully. <> 0 - An error occurred.		

Calendar.PlanLeadTimeBackward

DLL:	tcextccpapi		
Syntax:	<pre> long Calendar.PlanLeadTimeBackward(domain tcccp.ccal iCalendarCode, </pre>		

	domain tcccp.ractiAvailabilityType, domain tcdateiFinishDate, domain tcccp.ccapiLeadTime, domain tcccp.ptmu iTimeUnit, domain tczone iTimeZone, ref domain tcdateoFinishDate, ref domain tcdateoStartDate, ref domain tcmcs.s999moExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function plans lead time backward. For the given finish date and leadtime, the start date is calculated for the in the given calendar (= calendar code / availability type combination). Depending on the Time Unit (iTimeUnit) the planning will be done in seconds, hours or days (not capacity). The constraint here is that the given lead time also must be in seconds, hours or days. Use the following table for finding the unit of the lead time: +-----+-----+ Time Unit Lead Time +-----+-----+ Seconds (tcccp.pltmu.seconds) Seconds Hours (tcccp.pltmu.hours) Hours Days (tcccp.pltmu.days) Days +-----+-----+ So if the plan time unit is seconds than also the lead time must be expressed in seconds, the same goes for hours and days. Time Unit Hours can have fractions (e.g. 1.5), The Time Units Seconds and Days must be passed into this functions without fractions (e.g. 5400 when seconds or 1 if days). Input: iCalendarCode - Calendar Code, Mandatory iAvailabilityType - Availability Type, Mandatory iFinishDate - Requested Finish date (utc), Mandatory iLeadTime - Lead Time iTimeUnit - Time Unit, Mandatory iTimeZone - Time Zone, Optional When not specified, the time zone of the company is used Output: oFinishDate - Actual finish date (= iFinishDate or if iFinishDate is between availability periods: iStartDate moved backward to the end of the previous availability period) oStartDate - Calculated Start Date oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information.

	Return: 0	-	Start Date is succesfully calculated
	<> 0	-	Start Date could not be calculated

Calendar.PlanLeadTimeForward

DLL:	tcextccpapi																																					
Syntax:	<pre>long Calendar.PlanLeadTimeForward(domain tcccp.ccal iCalendarCode, domain tcccp.ract iAvailabilityType, domain tcdate iStartDate, domain tcccp.ccap iLeadTime, domain tcccp.ptmu iTimeUnit, domain tczone iTimeZone, ref domain tcdate oStartDate, ref domain tcdate oFinishDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>																																					
Usage:	Expl: This function plans lead time forward. For the given start date and leadtime, the end date is calculated for the in the given calendar (= calendar code / availability type combination). Depending on the Time Unit (iTimeUnit) the planning will be done in seconds, hours or days (not capacity). The constraint here is that the given lead time also must be in seconds, hours or days. Use the following table for finding the unit of the lead time: <table><tr><td colspan="2">+-----+-----+</td></tr><tr><td> Time Unit</td><td> Lead Time</td></tr><tr><td colspan="2">+-----+-----+</td></tr><tr><td> Seconds (tcccp.pltmu.seconds)</td><td> Seconds</td></tr><tr><td> Hours (tcccp.pltmu.hours)</td><td> Hours</td></tr><tr><td> Days (tcccp.pltmu.days)</td><td> Days</td></tr><tr><td colspan="2">+-----+-----+</td></tr></table> So if the plan time unit is seconds than also the lead time must be expressed in seconds, the same goes for hours and days. Time Unit Hours can have fractions (e.g. 1.5), The Time Units Seconds and Days must be passed into this functions without fractions (e.g. 5400 when seconds or 1 if days). Input: <table><tr><td>iCalendarCode</td><td>-</td><td>Calendar Code, Mandatory</td></tr><tr><td>iAvailabilityType</td><td>-</td><td>Availability Type, Mandatory</td></tr><tr><td>iStartDate</td><td>-</td><td>Start date (utc), Mandatory</td></tr><tr><td>iLeadTime</td><td>-</td><td>Lead Time</td></tr><tr><td>iTimeUnit</td><td>-</td><td>Time Unit, Mandatory</td></tr><tr><td>iTimeZone</td><td>-</td><td>Time Zone, Optional</td></tr></table> When not specified, the time zone of the company is used Output: <table><tr><td>oStartDate</td><td>-</td><td>Actual start date (= iStartDate or</td></tr></table>			+-----+-----+		Time Unit	Lead Time	+-----+-----+		Seconds (tcccp.pltmu.seconds)	Seconds	Hours (tcccp.pltmu.hours)	Hours	Days (tcccp.pltmu.days)	Days	+-----+-----+		iCalendarCode	-	Calendar Code, Mandatory	iAvailabilityType	-	Availability Type, Mandatory	iStartDate	-	Start date (utc), Mandatory	iLeadTime	-	Lead Time	iTimeUnit	-	Time Unit, Mandatory	iTimeZone	-	Time Zone, Optional	oStartDate	-	Actual start date (= iStartDate or
+-----+-----+																																						
Time Unit	Lead Time																																					
+-----+-----+																																						
Seconds (tcccp.pltmu.seconds)	Seconds																																					
Hours (tcccp.pltmu.hours)	Hours																																					
Days (tcccp.pltmu.days)	Days																																					
+-----+-----+																																						
iCalendarCode	-	Calendar Code, Mandatory																																				
iAvailabilityType	-	Availability Type, Mandatory																																				
iStartDate	-	Start date (utc), Mandatory																																				
iLeadTime	-	Lead Time																																				
iTimeUnit	-	Time Unit, Mandatory																																				
iTimeZone	-	Time Zone, Optional																																				
oStartDate	-	Actual start date (= iStartDate or																																				

		if iStartDate is between availability periods: iStartDate moved forward to the start of the next availability period)
	oFinishDate -	Calculated Finish Date
	oExceptionMessage -	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.
	oExceptionID -	An ID that refers to all error information. Use the functions in Exception to get all relevant information.
	Return: 0 -	Finish Date is successfully calculated
	<> 0 -	Finish Date could not be calculated

Chapter 5 Public Interfaces for IntercompanyTrade

Public Interfaces for IntercompanyTradeOrder

The following functions are available:

[IntercompanyTradeOrder.PresentForWarehouseOrderLine](#)

IntercompanyTradeOrder.PresentForWarehouseOrderLine

DLL:	tcextitrap
	This function is available from KB2321477 .
Syntax:	<pre> long IntercompanyTradeOrder.PresentForWarehouseOrderLine(domain tcncmp iWarehouseOrderCompany, domain tckoor iWarehouseKindOfOrder, domain tcorno iWarehouseOrder, domain tcpono iWarehouseOrderLine, domain tcpono iWarehouseOrderSequence, domain tcyesno iMultiCompanyTransfer, domain tcncmp iShipFromCompany, domain tcyesno iSalesTrade, domain tcyesno iPurchaseTrade, boolean iCheckSecurity, domain tcyesno iExcludeCancelled, domain tcmcs.str132 i.calling.api, ref boolean oIntercompanyTradeOrderPresent, ref domain tcncmp oTradeOrderCompany, ref domain tcorno oTradeOrder, ref domain tcpono oTradeOrderLine, ref domain tcyesno oTradeTransactions, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function checks if a intercompany trade order is present for the Warehouse Order Line. Pre: N.A. Post: N.A. Input: iWarehouseOrderCompany - Warehouse Order Company (Mandatory) iWarehouseKindOfOrder - Warehouse Kind of Order (Mandatory)

Public Interfaces for IntercompanyTrade

	iWarehouseOrder - Warehouse Order (Mandatory) iWarehouseOrderLine - Warehouse Order Line iWarehouseOrderSequence - Warehouse Order Sequence iMultiCompanyTransfer - Multi Company Transfer (Mandatory) iShipFromCompany - ShipFromCompany (Mandatory when iMultiCompanyTransfer has value Yes) iSalesTrade - Sales Trade (Mandatory) iPurchaseTrade - Purchase Trade (Mandatory) iCheckSecurity - Check Security (Mandatory) iExcludeCancelled - Exclude Cancelled (Mandatory) Output: oIntercompanyTradeOrderPresent - Intercompany Trade Order Present oTradeOrderCompany - Trade Order Company oTradeOrder - Trade Order oTradeOrderLine - Trade Order Line oTradeTransactions - Trade Transactions oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant Return: 0: OK, <> 0: Error

Chapter 6 Public Interfaces for Item

Public Interfaces for Item

The following functions are available:

[Item.ConvertItemToPlanItem](#)

[Item.ConvertPlanItemToItem](#)

[Item.Copy](#)

[Item.CopySiteData](#)

[Item.CreateSerialNumber](#)

[Item.GetCostingData](#)

[Item.GetData](#)

[Item.GetLotSerialRegistration](#)

[Item.GetOrderingData](#)

[Item.GetPlanningTimeFenceDate](#)

[Item.GetProductionData](#)

[Item.GetPurchaseData](#)

[Item.GetRevision](#)

[Item.GetSalesData](#)

[Item.GetServiceData](#)

[Item.GetSupplier](#)

[Item.GetWarehousingData](#)

[Item.InsertSerialNumber](#)

[Item.ProjectPegAllowed](#)

[Item.StartCopy](#)

[Item.StartDetail](#)

[Item.StartMultiMain](#)

[Item.StartOverview](#)[Item.StartWhereUsedComponent](#)

Item.ConvertItemToPlanItem

DLL:	cpextrpdapi
Syntax:	<pre> long Item.ConvertItemToPlanItem(domain tcncmp iLogisticCompany, domain tcitem iItem, domain tcomm.clus iPlanningCluster, ref domain cpitem oPlanItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function determines the plan item code for the given general item and cluster. Pre: None Post: None Input: iLogisticCompany Logistic Company (Mandatory) iItem General Item (Mandatory) iPlanningCluster Planning Cluster (Mandatory) Output: oPlanItem Plan Item code oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Function is executed successfully <> 0 An error occurred

Item.ConvertPlanItemToItem

DLL:	cpextrpdapi
Syntax:	<pre> long Item.ConvertPlanItemToItem(domain tcncmp iLogisticCompany, domain cpitem iPlanItem, ref domain tcitem oItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function determines the tcitem code for the given plan-item. Pre: None Post: None

Input:	iLogisticCompany	Logistic Company (Mandatory)
	iPlanItem	Plan Item (Mandatory)
Output:	oItem	Item
	oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
	oExceptionID	An ID that refers to all error information. Use the functions in Exception to get all relevant information.
Return:	0	Function is executed successfully
	<> 0	An error occurred

Item.Copy

DLL:	tcextibdapi												
	This function is available from KB2302509 .												
Syntax:	<pre>long Item.Copy(domain tcitem iSourceItem, domain tcitem iTargetItem, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>												
Usage:	Expl: Copy the Source Item to the Target Item using the defaults or copy options as provided in the input Processing Option Set. By default, transaction management is executed within the function, so no pending transactions should be present before calling this function. Processing Option 'transactionManagementByCaller' is available to have the transaction management under control of the caller. Pre: - Post: - Input: iSourceItem Mandatory iTargetItem Mandatory iProcessingOptionSet Optional, if 0, the default copy options are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create(). Processing Options have a direct relationship with the form fields on session Copy Item Data (tcibd0205m000) and are not explained in further detail here. Please refer to the session help for additional information. Copy Item Data options which are not available as Processing Options will get defaulted in accordance with the session logic. Processing Options that are set while a required Implemented Software Component is not available are ignored. <table><tr><th>NAME</th><th>TYPE</th><th>DEFAULT</th></tr><tr><td>targetItemDescription</td><td>string</td><td><from source item></td></tr><tr><td>copyAlternativeItems</td><td>domain tcyesno</td><td>tcyesno.yes</td></tr><tr><td>copyBillOfMaterial</td><td>domain tcvesno</td><td>tcvesno.yes</td></tr></table>	NAME	TYPE	DEFAULT	targetItemDescription	string	<from source item>	copyAlternativeItems	domain tcyesno	tcyesno.yes	copyBillOfMaterial	domain tcvesno	tcvesno.yes
NAME	TYPE	DEFAULT											
targetItemDescription	string	<from source item>											
copyAlternativeItems	domain tcyesno	tcyesno.yes											
copyBillOfMaterial	domain tcvesno	tcvesno.yes											

	<pre> copyRouting domain tyesno tyesno.yes copyEItemRelations domain tyesno tyesno.yes copySupplyingRelationships domain tyesno tyesno.yes copyTradeManagement domain tyesno tyesno.yes copyItemText domain tyesno tyesno.yes copyBusinessPartnersByItem domain tyesno tyesno.yes copyDocuments domain tyesno tyesno.no copyDocumentLinks domain tyesno tyesno.yes copyProjectActivities domain tyesno tyesno.yes copyReferenceDesignators domain tyesno tyesno.yes copyRequirementsByItem domain tyesno tyesno.yes copyAttributeSets domain tyesno tyesno.yes copyItemsBySiteAndItemsByOffice domain tyesno tyesno.yes copyConstraints domain tyesno tyesno.yes copyFeaturesAndOptions domain tyesno tyesno.yes copyItemGenerationSettings domain tyesno tyesno.yes copyGenericRouting domain tyesno tyesno.yes copyGenericBillsOfMaterial domain tyesno tyesno.yes copyGenericPriceLists domain tyesno tyesno.yes projectPartQuantity double 0.0 approveConversionFactors domain tyesno tyesno.yes </pre> Note: The session option to Calculate Standard Cost and Valuation Price is not included in the set and therefore this action is not supported. StandardCosts.CalculateForNewItem() should be used instead. Output: <pre> oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. </pre> Return: 0 - The item was copied. <> 0 - Otherwise.

Item.CopySiteData

DLL:	ttextibdapi This function is available from KB2302509 .
Syntax:	<pre> long Item.CopySiteData(domain tcitem iSourceItem, domain tcsite iSourceSite, domain tcitem iTargetItem, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl: Copy the Item by Site data from the Source Item to the Target Item. Pre: db.retry.point() must have been set. </pre>

	Post: Commit or abort the transaction. Input: iSourceItem Mandatory, item to copy Site Data from iSourceSite Mandatory, Site data to copy iTargetItem Mandatory, item to copy Site Data to iProcessingOptionSet Optional, if 0, the default copy options are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create(). This parameter is for future use, so currently no processing options are available for this Public Interface. Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Site Data was copied. <> 0 - Otherwise.

Item.CreateSerialNumber

DLL:	tcextibdapi
Syntax:	<pre>long Item.CreateSerialNumber(domain tcsite iSite, domain tcitem iItem, domain whinh.oorg iOrderOrigin, domain tcorno iOrder, domain tcwset iOrderSet, domain tcuef.mask iMask, ref domain tcibd.sern oSerialNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Create a Serial Number for a given Item. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iSite Site (Mandatory when the Site concept is Active) iItem Item (Mandatory) iOrderOrigin Order Origin (Optional) iOrder Order (Optional). iOrderSet Order Set (Optional) iMask Mask (Optional) Output: oSerialNumber The created Serial Number

	<code>oExceptionMessage</code> The last message if the return value is not equal to 0. If more than one message is given, these are present in the <code>oExceptionID</code>. <code>oExceptionID</code> An ID that refers to all error information. Use the functions in <code>Exception</code> to get all relevant information. Return: 0 - Serial Number has been created <> 0 - Error. A new Serial Number could not be created

Item.GetCostingData

DLL:	tiextcprapi																																																																											
Syntax:	<pre>long Item.GetCostingData(domain tcncmp iLogisticCompany, domain tcitem iItem, domain tcemm.grid iEnterpriseUnit, boolean iForceRead, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...)</pre>																																																																											
Usage:	Expl: This function retrieves data from Item - Costing Data. Individual fields are retrieved, multiple fields can be retrieved in one call. The enterprise.unit argument is only used when the parameter Standard Cost by Enterprise Unit is set to Active in Implemented Software Components. In that case it is necessary to supply an Enterprise Unit when the item-type requires this. For item types Cost or Service, no Enterprise Unit is expected. The set of supported fields for retrieval is limited. The ones marked with EU only have relevance when Standard Cost by Enterprise Unit is Active. <table><tr><td>Mnemonic</td><td>-</td><td>Description</td><td>-</td><td>Scope</td></tr><tr><td>type</td><td>-</td><td>Item Costing Type</td><td>-</td><td>EU</td></tr><tr><td>base</td><td>-</td><td>Standard Cost Base</td><td>-</td><td>EU</td></tr><tr><td>cwar</td><td>-</td><td>Warehouse</td><td>-</td><td>EU</td></tr><tr><td>cofc</td><td>-</td><td>Supplying Purchase Office</td><td>-</td><td>EU</td></tr><tr><td>scos</td><td>-</td><td>Costing Source</td><td>-</td><td>EU</td></tr><tr><td>sueu</td><td>-</td><td>Supplying Enterprise Unit</td><td>-</td><td>EU</td></tr><tr><td>spit</td><td>-</td><td>Surcharges by Item</td><td>-</td><td></td></tr><tr><td>vpwh</td><td>-</td><td>Surcharges by Warehouse</td><td>-</td><td></td></tr><tr><td>ccur</td><td>-</td><td>Item Costing Currency</td><td>-</td><td></td></tr><tr><td>ltcp</td><td>-</td><td>Last Calculation Date</td><td>-</td><td></td></tr><tr><td>chrt</td><td>-</td><td>Standard Cost Component Scheme</td><td>-</td><td></td></tr><tr><td>coyn</td><td>-</td><td>Combined Ownership Allowed</td><td>-</td><td></td></tr><tr><td>inlc</td><td>-</td><td>Included Landed Costs</td><td>-</td><td></td></tr><tr><td>lcst</td><td>-</td><td>Landed Costs Set</td><td>-</td><td></td></tr></table> Note: this function uses variable arguments, for input and output. Data is retrieved via value pairs: specify the field and the field value. Example: when item costing warehouse and currency are needed,	Mnemonic	-	Description	-	Scope	type	-	Item Costing Type	-	EU	base	-	Standard Cost Base	-	EU	cwar	-	Warehouse	-	EU	cofc	-	Supplying Purchase Office	-	EU	scos	-	Costing Source	-	EU	sueu	-	Supplying Enterprise Unit	-	EU	spit	-	Surcharges by Item	-		vpwh	-	Surcharges by Warehouse	-		ccur	-	Item Costing Currency	-		ltcp	-	Last Calculation Date	-		chrt	-	Standard Cost Component Scheme	-		coyn	-	Combined Ownership Allowed	-		inlc	-	Included Landed Costs	-		lcst	-	Landed Costs Set	-	
Mnemonic	-	Description	-	Scope																																																																								
type	-	Item Costing Type	-	EU																																																																								
base	-	Standard Cost Base	-	EU																																																																								
cwar	-	Warehouse	-	EU																																																																								
cofc	-	Supplying Purchase Office	-	EU																																																																								
scos	-	Costing Source	-	EU																																																																								
sueu	-	Supplying Enterprise Unit	-	EU																																																																								
spit	-	Surcharges by Item	-																																																																									
vpwh	-	Surcharges by Warehouse	-																																																																									
ccur	-	Item Costing Currency	-																																																																									
ltcp	-	Last Calculation Date	-																																																																									
chrt	-	Standard Cost Component Scheme	-																																																																									
coyn	-	Combined Ownership Allowed	-																																																																									
inlc	-	Included Landed Costs	-																																																																									
lcst	-	Landed Costs Set	-																																																																									

	the function must be called as follows: <pre> if Item.GetCostingData(/* Fixed arguments: company, --> input item, --> input enterprise.unit, --> input force.read, --> input exception.message, --> output exception.id, --> output /* Variable arguments: "cwar", --> input warehouse, --> output "ccur", --> input currency) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif </pre> Pre: None Post: None Input: <table> <tr> <td>iLogisticCompany</td><td>- Logistic Company: Mandatory</td></tr> <tr> <td>iItem -</td><td>- Item: Mandatory</td></tr> <tr> <td>iEnterpriseUnit</td><td>- Enterprise Unit - not mandatory</td></tr> <tr> <td>iForceRead</td><td>- Option to force new query instead of using cached information</td></tr> <tr> <td>...</td><td>- The field mnemonic of the required field.</td></tr> </table> Output: <table> <tr> <td>oExceptionMessage</td><td>- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID</td></tr> <tr> <td>oExceptionID</td><td>- An ID that refers to all error information. Use the functions in Exception to get all relevant information.</td></tr> <tr> <td>...</td><td>- The value of the required field.</td></tr> </table> Return: 0 <> 0 <table> <tr> <td></td><td>- Data read</td></tr> <tr> <td></td><td>- Otherwise.</td></tr> </table>	iLogisticCompany	- Logistic Company: Mandatory	iItem -	- Item: Mandatory	iEnterpriseUnit	- Enterprise Unit - not mandatory	iForceRead	- Option to force new query instead of using cached information	...	- The field mnemonic of the required field.	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.	...	- The value of the required field.		- Data read		- Otherwise.
iLogisticCompany	- Logistic Company: Mandatory																				
iItem -	- Item: Mandatory																				
iEnterpriseUnit	- Enterprise Unit - not mandatory																				
iForceRead	- Option to force new query instead of using cached information																				
...	- The field mnemonic of the required field.																				
oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID																				
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.																				
...	- The value of the required field.																				
	- Data read																				
	- Otherwise.																				

Item.GetData

DLL:	tcextibdapi
Syntax:	<pre> long Item.GetData(domain tcncomp iLogisticCompany, domain tcitem iItem, domain tcsite iSite, boolean iForceRead, ref domain tcmcs.s999m oExceptionMessage mb, </pre>

	ref	long	oExceptionID, ...)
Usage:	Expl: Retrieve data from Items The required field is specified by the field mnemonic, the output value is filled in the output argument depending on the data type. This function is Multi Site aware, data is read from general level or site level, depending on the input arguments Site and implementation phase of Multi Site. Note: this function uses variable arguments, for input and output. Data is retrieved via value pairs: specify the field and the field value. Example: when item project and customized are needed, the function must be called as follows: <pre> if Item.GetData(/* Fixed arguments: company, --> input item, --> input site, --> input force.read, --> input exception.message, --> output exception.id, --> output /* Variable arguments: "cprj", --> input project, --> output "cust", --> input customized) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif Pre: None Post: None Input: iLogisticCompany - Logistic Company: Mandatory. iItem - Item: Mandatory. iSite - Site: Not Mandatory. iForceRead - Option to force new query in stead of using cached information. ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read. <> 0 - Otherwise. </pre>		

Item.GetLotSerialRegistration

DLL:	whextwmdapi
Syntax:	<pre> long Item.GetLotSerialRegistration(domain tcitem iItem, domain whkorg iKindOfOrigin, domain whinh.oorg iOrderOrigin, domain whinh.ittp iTransactionType, ref boolean oLotRegistration, ref boolean oSerialRegistration, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface determines if Lot and Serial registration is required for the given Item, Kind of Origin, Order Origin and Transaction Type. Pre: N.A. Post: N.A. Input: iItem - Item (Mandatory) iKindOfOrigin - Kind of Origin (Mandatory) iOrderOrigin - Order Origin (Mandatory) iTransactionType - TransactionType (Mandatory) Output: oLotRegistration - Lot registration required oSerialRegistration - Serial registration required oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Success <> 0 - Error

Item.GetOrderingData

DLL:	tcextibdapi
Syntax:	<pre> long Item.GetOrderingData(domain tcncmp iLogisticCompany, domain tcitem iItem, domain tcsite iSite, boolean iForceRead, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: Retrieves data from Items - Ordering. The required field is specified by the field menomonic, the output value is filled in the output argument depending on the data type. This function is Multi Site aware, data is read from general level or site level, depending on the input arguments Site and implementation phase of Multi Site.

	Note: this function uses variable arguments, for input and output. Data is retrieved via value pairs: specify the field and the field value. Example: when item ordering warehouse and order interval are needed, the function must be called as follows: <pre> if Item.GetOrderingData(/* Fixed arguments: company, --> input item, --> input site, --> input force.read, --> input exception.message, --> output exception.id, --> output /* Variable arguments: "cwar", --> input warehouse, --> output "oint", --> input order.interval) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif Pre: None Post: None Input: iLogisticCompany - Logistic Company: Mandatory iItem - - Item: Mandatory iSite - - Site: Not Mandatory iForceRead - Option to force new query in stead of using cached information. ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read. <> 0 - Otherwise. </pre>

Item.GetPlanningTimeFenceDate

DLL:	cpextrpdapi
Syntax:	long Item.GetPlanningTimeFenceDate(domain tcncmp iLogisticCompany,

	domain cpcom.plnc iScenario, domain cpitem iPlanItem, ref domain tcdate oTimeFenceDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function determines the time fence date for a specific company and plan item. Pre: None Post: None Input: iLogisticCompany Logistic Company (Mandatory) iScenario Scenario (Mandatory) iPlanItem Plan Item (Mandatory) Output: oTimeFenceDate Time Fence Date oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Function is executed successfully <> 0 An error occurred

Item.GetProductionData

DLL:	tiextipdapi
Syntax:	<pre>long Item.GetProductionData (domain tcncmp iLogisticCompany, domain tcitem iItem, domain tcsite iSite, boolean iForceRead, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...)</pre>
Usage:	Expl: This function retrieves data from Item - Production Data Individual fields are retrieved, multiple fields can be retrieved in one call. Two levels of data exist, global - which is the relevant scope unless site specific data exists. When the Job Shop by Site or the Sites concept is active, some fields have modified relevance or scope. STANDARD: Fields are always read from global data or the specified Site. When the Sites concept is not active, the iSite parameter is ignored: cpha - Phantom cick - Critical for Inventory iimf - Receipt Inspection iima - Outbound Inspection cncd - Conformance Reporting

	<pre> rgrp - Routing Group txta - Item Production Text Ignore-Site: Fields are always read from global data. seak - Search Key jsst - Job Shop Site Site-NA: Fields cannot be read when the Sites concept is active. repi - Repetitive Item scdl - Schedule Code pcrp - Rate Factor for Planning JSBS-NA: Fields cannot be read when the Job Shop by Site concept is active. oqdr - Quantity-Dependent Routing phst - Use Phantom Inventory swoc - Warehouse is Mandatory on BOM line JSBS-Site-Required: Fields must be read with iSite specified when the Job Shop by Site concept is active. iuma - Turn operation 0 into First of bfcp - Backflush if Material bfep - Backflush Materials bfhr - Backflush Hours drin - Direct Initiate Inventory Issue dris - Direct Proc. Wareh. Order Line nsfc - Net Change JSC sfpl - Shop Floor Planner JSBS-NA-for-Product: Field cannot be read when Job Shop by Site concept is active and the item is of type Product. For other item-types, the field may only be read when iSite is not specified. unom - BOM Quantity runi - Routing Quantity scpf - Scrap Percentage scpq - Scrap Quantity oltm - Order Lead Time oltu - Order Lead Time Unit smfs - Subcontracting with Material Flow Note: this function uses variable arguments, for input and output. Data is retrieved via value pairs: specify the field and the field value. Example: when Shop Floor Planner and the Critical for Inventory fields are needed, the function must be called as follows: if Item.GetProductionData(/* Fixed Arguments company, --> input item, --> input site, --> input force.read, --> input exception.message, --> output exception.id, --> output /* Variable arguments: </pre>

	<pre> "sfpl", --> input sfc.planner, --> output "cick", --> input critical.for.inventory) <> 0 --> output then /* Error, do something Exception.Delete(exception.id) endif Pre: None Post: None Input: iLogisticCompany - Logistic Company: Mandatory iItem - Item: Mandatory iSite - Site - not mandatory iForceRead - Option to force new query instead of using cached information ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the requested field. Return: 0 - Data read <> 0 - Otherwise. </pre>

Item.GetPurchaseData

DLL:	tdextipuapi
Syntax:	<pre> long Item.GetPurchaseData(domain tcncmp iLogisticCompany, domain tcitem iItem, domain tcsite iSite, domain tccwoc iPurchaseOffice, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves data from Item - Purchase. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, data is read from general level, or office, or site level, depending on the input arguments Site and Office and implementation phase of Multi Site. Note: this function uses variable arguments, for input and

	output. Data is retrieved via value pairs: specify the field and the field value. Example: when item purchase price and currency are needed, the function must be called as follows: <pre> if Item.GetPurchaseData(* Fixed arguments: company, --> input item, --> input site, --> input purchase office, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output * Variable arguments: "ccur", --> input purchase.currency, --> output "prip", --> input purchase.price) <> 0 then --> output * Error, do something Exception.Delete(exception.id) endif </pre> Pre: None Post: None Input: - iLogisticCompany - Logistic Company: Mandatory - iItem - Item: Mandatory - iSite - Site: Not Mandatory - iPurchaseOffice - Purchase Office: Not Mandatory - iForceRead - Option to force new query in stead of using cached information - iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. Output: - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - An error occurred

Item.GetRevision

DLL:	tcextibdapi
	This function is available from KB2295055 .
Syntax:	<pre> long Item.GetRevision(domain tcncmp iLogisticCompany, domain tcitem iItem, domain tcdate iDate, ref domain tiedm.revi oRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function retrieves the Item Revision on the given date. Pre: None Post: None Input: - iLogisticCompany - Logistic Company: Mandatory - iItem - Item: Mandatory - iDate - Date: Mandatory Output: - oRevision - Item Revision - oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Data read. <> 0 - Otherwise.

Item.GetSalesData

DLL:	tdextisaapi
Syntax:	<pre> long Item.GetSalesData(domain tcncmp iLogisticCompany, domain tcitem iItem, domain tccwoc iSalesOffice, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves data from Item - Sales. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, data is read from general level, or office, or site level, depending on the input arguments Site and Office and implementation phase of Multi Site.

	Note: this function uses variable arguments, for input and output. Data is retrieved via value pairs: specify the field and the field value. Example: when item sales price and currency are needed, the function must be called as follows: <pre> if Item.GetSalesData(/* Fixed arguments: company, --> input item, --> input site, --> input sales office, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output /* Variable arguments: "ccur", --> input sales.currency, --> output "pris", --> input sales.price) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif </pre> Pre: None Post: None Input: <table> <tr> <td>iLogisticCompany</td><td>- Logistic Company: Mandatory</td></tr> <tr> <td>iItem -</td><td>- Item: Mandatory</td></tr> <tr> <td>iSite -</td><td>- Site: Not Mandatory</td></tr> <tr> <td>iSalesOffice</td><td>- Sales Office: Not Mandatory</td></tr> <tr> <td>iForceRead</td><td>- Option to force new query in stead of using cached information</td></tr> <tr> <td>iContextIsMasterData</td><td>- True/False: Indicates if call is done to retrieve data for master data or transactional data.</td></tr> <tr> <td>...</td><td>- The field mnemonic of the required field.</td></tr> </table> Output: <table> <tr> <td>oExceptionMessage</td><td>- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID</td></tr> <tr> <td>oExceptionID</td><td>- An ID that refers to all error information. Use the functions in Exception to get all relevant information.</td></tr> <tr> <td>...</td><td>- The value of the required field.</td></tr> </table> Return: 0 <> 0 <table> <tr> <td>0</td><td>- Data read</td></tr> <tr> <td><> 0</td><td>- Otherwise.</td></tr> </table>	iLogisticCompany	- Logistic Company: Mandatory	iItem -	- Item: Mandatory	iSite -	- Site: Not Mandatory	iSalesOffice	- Sales Office: Not Mandatory	iForceRead	- Option to force new query in stead of using cached information	iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.	...	- The field mnemonic of the required field.	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.	...	- The value of the required field.	0	- Data read	<> 0	- Otherwise.
iLogisticCompany	- Logistic Company: Mandatory																								
iItem -	- Item: Mandatory																								
iSite -	- Site: Not Mandatory																								
iSalesOffice	- Sales Office: Not Mandatory																								
iForceRead	- Option to force new query in stead of using cached information																								
iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.																								
...	- The field mnemonic of the required field.																								
oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID																								
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.																								
...	- The value of the required field.																								
0	- Data read																								
<> 0	- Otherwise.																								

Item.GetServiceData

DLL:	tsextmdmapi
Syntax:	<pre> long Item.GetServiceData(domain tcncmp iLogisticCompany, domain tcitem iItem, domain tccwoc iServiceOffice, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves data from Item - Service. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, data is read from general level, or office, or site level, depending on the input arguments Site and Office and implementation phase of Multi Site. Note: this function uses variable arguments, for input and output. Data is retrieved via value pairs: specify the field and the field value. Example: when item sales price and currency are needed, the function must be called as follows: <pre> if Item.GetServiceData(* Fixed arguments: company, --> input item, --> input site, --> input service.office, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output * Variable arguments: "ccur", --> input currency, --> output "pris", --> input sales.price) <> 0 then --> output * Error, do something Exception.Delete(exception.id) endif </pre> Pre: None Post: None Input: - iLogisticCompany - Logistic Company: Mandatory - iItem - Item: Mandatory - iSite - Site: Not Mandatory - iServiceOffice - Service Office: Not Mandatory - iForceRead - Option to force new query in stead of using cached information - iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional

	<pre> ... Output: oExceptionMessage oExceptionID Return: 0 <> 0 </pre>	<pre> data. - The field mnemonic of the required field. - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - The value of the required field. - Data read - An error occurred </pre>

Item.GetSupplier

DLL:	tdextipuapi
Syntax:	<pre> long Item.GetSupplier(domain tcncmp iLogisticCompany, domain tccitg iItemGroup, domain tcitem iItem, domain tcsite iSite, domain tcefex.date iEffectiveDate, domain tcqrd2 iQuantity, boolean iForceRead, ref domain tccom.bpid oBuyFromBusinessPartner, ref domain tccom.bpid oShipFromBusinessPartner, ref boolean oBusinessPartnerFoundOnItemBPLevel, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function returns the first valid Buy-from Business Partner and Ship-from Business Partner based on priority for the specified Item/Item Group. Notes: - In this context, Item - Purchase BP level refers to Item - Purchase BP or Item - Purchase BP by Site. Item - Purchase level refers to Item - Purchase or Item - Purchase by Site. - Search is performed in Item - Purchase BP tables and Item - Purchase tables. - If i.site is filled, then the search will first be performed in Item - Purchase (BP) by Site tables. If no entry is found there, then Item - Purchase (BP) tables are searched. - Only active BPs will be returned. Selecting data to retrieve BPs is done in the following order: 1) Item-level: Item - Purchase if approved on Item - Purchase BP level. 2) Item-level: Item - Purchase BP, approved single source BP. 3) Item-level: Item - Purchase BP, approved BP. 4) Item-group-level: Item - Purchase if approved on Item -

	Purchase BP level. 5) Item-group-level: Item - Purchase BP, approved single source BP. 6) Item-group-level: Item - Purchase BP, approved BP. If Sourcing from Approved Vendors is not checked on Item - Purchase: 7) Item-level: Item - Purchase. Pre: NA Post: NA Input: iLogisticCompany - Logistic Company: Mandatory iItemGroup - Item Group; Optional if Item is filled. iItem - Item; Mandatory if Item Group not filled. iSite - Site: Not Mandatory iEffectiveDate - Mandatory; use utc.num() if current date is needed. iQuantity - Optional; use 0.0 if no minimum order quantity must be considered. iForceRead - Option to force new query in stead of using cached information Output: oBuyFromBusinessPartner - Buy-from Business Partner oShipFromBusinessPartner- Ship-from Business Partner oBusinessPartnerFoundOnItemBPLevel - True: Found on Item BP level. False: No BP found or not found on Item - BP level. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Data read <> 0 - An error occurred

Item.GetWarehousingData

DLL:	whextwmdapi
Syntax:	<pre> long Item.GetWarehousingData(domain tcncmp iLogisticCompany, domain tcitem iItem, domain tcsite iSite, boolean iForceRead, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves data from Item - Warehousing. Individual fields are retrieved, multiple fields can be

retrieved in one call.

This function is Multi Site aware, data is read from general level or site level, depending on the input argument Site and implementation phase of Multi Site.

Note: this function uses variable arguments, for input and output. Data is retrieved via value pairs: specify the field and the field value.

Example: when the indication 'Cross-docking', Minimum Cross-dock quantity and Maximum Cross-dock quantity are needed, the function must be called as follows:

```
if Item.GetWarehousingData(
    /* Fixed arguments:

        company,                --> input
        item,                   --> input
        site,                   --> input
        force.read,             --> input
        exception.message,      --> output
        exception.id,           --> output

    /* Variable arguments:

        "dycd",                 --> input
        dynamic.cross.docking,  --> output
        "qcmi",                 --> input
        minimum.cross.dock.quantity, --> output
        "qcma",                 --> input
        maximum.cross.dock.quantity) --> output
    <> 0 then

    /* Error, do something
    Exception.Delete(exception.id)
endif
```

Pre: None

Post: None

Input:

```
iLogisticCompany  - Logistic Company: Mandatory
iItem             - Item: Mandatory
iSite             - Site: Not Mandatory
iForceRead        - Option to force new query in stead of
                    using cached information
...               - The field mnemonic of the required
                    field.
```

Output:

```
oExceptionMessage - The last message if the return
                    value is not equal to 0.
                    If more than one message is
                    given, these are present in the
                    oExceptionID
oExceptionID       - An ID that refers to all error
                    information. Use the functions in
                    Exception to get all relevant
                    information.
...               - The value of the required field.
```

Return: 0

```
<> 0
- Error
```

Item.InsertSerialNumber

DLL:	tcextibdapi
	This function is available from KB2259343 .
Syntax:	<pre> long Item.InsertSerialNumber(domain tcitem iItem, domain tcibd.sern iSerialNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Insert a Serial Number for a given Item. The combination of the Serial Number and the Item must not exist. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iItem Item (Mandatory) iSerialNumber The Serial Number to insert (Mandatory) Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - The Serial Number has been inserted <> 0 - Error. The Serial Number could not be created

Item.ProjectPegAllowed

DLL:	tcextibdapi
Syntax:	<pre> long Item.ProjectPegAllowed(domain tcitem iItem, ref boolean oProjectPegAllowed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Check whether it is allowed to use a project peg for an item. Pre: - Post: - Input: iItem Item (Mandatory). Output: oProjectPegAllowed - True if Project peg is allowed. - False if Project peg is not allowed. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.

	<code>oExceptionID</code> An ID that refers to all error information. Use the functions in <code>Exception</code> to get all relevant information. Return: 0 - Project peg allowed was successfully determined. <> 0 - Error. Project peg allowed could not be determined.

Item.StartCopy

DLL:	tcextibdapi
Syntax:	<pre>long Item.StartCopy(long iStartMode, domain tcitem iSourceItem, ref domain tcitem oTargetItem, ref domain tcitm.dscr oTargetItemDescription mb, ref boolean oItemCopied, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...)</pre>
Usage:	Expl: Starts the session Copy Item Data (tcibd0205m000). Input: <code>iStartMode</code> Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. <code>iItem</code> Mandatory ... Additional input, specifies the option in a string, followed by the required value in the proper format. Possible options are: targetItem A string containing the item code of the to be created item targetItemDescription A string containing the item description of the to be created item approveConversionFactors A boolean for the copy option approve conversion factors Note: This function uses variable arguments, for input and output. Data is retrieved via name-value pairs: specify the field and a variable for the field value. Example: when the session Copy Item Data has to be started with the new item code and description filled and approve conversion factors checked 'yes', you use the following arguments: <pre>Item.StartCopy(</pre>

	<pre> /* Fixed arguments: start.mode, --> input item, --> input target.item, --> output target.item.description --> output item.copied --> output exception.message, --> output exception.id, --> output /* Variable arguments: "targetItem", --> input target.item, --> input "targetItemDescription", --> input target.item.description, --> input "approveConversionFactors", --> input true) --> input </pre>
Output:	<pre> oTargetItem - The Item Code of the created Item oTargetItemDescription - The Item Description of the created Item oItemCopied - True, if the item has been copied - False, otherwise oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. </pre>
Return: 0	- Session started
<> 0	- Otherwise.

Item.StartDetail

DLL:	tcextibdapi
Syntax:	<pre> long Item.StartDetail(long iStartMode, domain tcitem iItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Starts the session Item General (tcibd0501m000) in detail mode. This function can only be used when the sites concept is NOT active. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iItem

	Mandatory Output: oExceptionMessage oExceptionID Return: 0 <> 0	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Session started - Otherwise.

Item.StartMultiMain

DLL:	tcextibdapi	
Syntax:	<pre>long Item.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: Starts the (multimain) session Item (tcibd0501m100). This function can only be used when the sites concept is active or in preparation. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used iSessionIndex Not used iQueryExtend Not used iItem Mandatory Output: oExceptionMessage oExceptionID Return: 0 <> 0	

Item.StartOverview

DLL:	tcextibdapi
Syntax:	<pre> long Item.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, domain tckitm iItemType, domain tccpva iProductVariant, domain tccprj iPCSPProject, domain tcosys iOrderSystem, ref domain tcitem oItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Starts the session Item General (tcibd0501m000) in overview mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used iSessionIndex Value: 1: Item 6: ItemType, Item 8: ProductVariant, Project, Item 10: OderSystem, Item Determines the sort order in the session. E.g. if iSessionIndex = 6, then Items are sorted by Item Type and Item, and the optional input arguments iItemType and iItem can be used to execute an automatic find action on start of the session. iQueryExtend A specific query to be used when starting the session. Use this to specfiy a filter, e.g. "tcibd001.kitm = tckitm.product" Using the query extend may lead to a "data not found, session not started" situation. iItem optional iItemType optional iProductVariant optional iPCSPProject optional iOrderSystem optional

	Output: for iStartMode MODAL: oItem - item of the selected row oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Otherwise.

Item.StartWhereUsedComponent

DLL:	tiextmfcapi
Syntax:	<pre>long Item.StartWhereUsedComponent(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, domain tcsite iSite, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the session Where-Used Component (timfc0620m000) Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used iSessionIndex Not used iQueryExtend Not used iItem The component for which the where-used info is requested. Mandatory iSite If specified: only where-used information for this site will be shown. Optional Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in

	Return: 0 <> 0	Exception to get all relevant information. - Session started - Otherwise.

Public Interfaces for ItemBySite

The following functions are available:

[ItemBySite.StartDetail](#)

[ItemBySite.StartOverview](#)

ItemBySite.StartDetail

DLL:	tcextibdapi	
Syntax:	<pre>long ItemBySite.StartDetail(long iStartMode, domain tcsite iSite, domain tcitem iItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: Starts the session Items by Site General (tcibd1550m000) in detail mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iSite Mandatory iItem Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started	

	<> 0	- Otherwise.

ItemBySite.StartOverview

DLL:	tcextibdapi
Syntax:	<pre> long ItemBySite.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcsite iSite, domain tcitem iItem, ref domain tcsite oSite, ref domain tcitem oItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Starts the session Items by Site General (tcibd1550m000) in overview mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used iSessionIndex Value: 1: Site, Item 2: Item, Site iQueryExtend A specific query to be used when starting the session. Using the query extend may lead to a "data not found, session not started" situation. iSite Mandatory when iSessionIndex = 1 iItem Mandatory when iSessionindex = 2 Output: for iStartMode MODAL: oSite - site of the selected row oItem - item of the selected row oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Otherwise.

Public Interfaces for ItemPurchase

The following functions are available:

[ItemPurchase.StartDetail](#)

[ItemPurchase.StartMultiMain](#)

[ItemPurchase.StartOverview](#)

ItemPurchase.StartDetail

DLL:	tdextipuapi
Syntax:	<pre> long ItemPurchase.StartDetail(long iStartMode, domain tcitem iItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the detail session Items - Purchase (tdipu0101m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Output: iItem Item (Mandatory; given item must exist) oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

ItemPurchase.StartMultiMain

DLL:	tdextipuapi
Syntax:	<pre> long ItemPurchase.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcitem iItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the Multi-Main session Item - Purchase (tdipu0601m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iQueryExtend A specific query to be used when zooming to this session. iItem Item (Mandatory) Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

ItemPurchase.StartOverview

DLL:	tdextipuapi
Syntax:	<pre> long ItemPurchase.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, domain tcseak iSearchKey mb, domain tcmno iBuyer, ref domain tcitem oItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Items - Purchase

	in overview mode (tdipu0101m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex The index that will be used. Supported values: 1: sort by Item 2: sort by Search Key/Item 3: sort by Buyer/Item iQueryExtend A specific query to be used when zooming to this session. iItem Item iSearchKey Search Key iBuyer Buyer Output: for iStartMode MODAL: oItem Selected Item oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 Session started An error occurred

Public Interfaces for ItemPurchaseBySite

The following functions are available:

[ItemPurchaseBySite.StartDetail](#)

ItemPurchaseBySite.StartDetail

DLL:	tdextipuapi
Syntax:	long ItemPurchaseBySite.StartDetail(

	<pre> long iStartMode, domain tcitem iItem, domain tcitem iSite, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>								
Usage:	Expl: This function starts the detail session Items - Purchase (tdipu0101m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory. If the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: <table> <tr> <td>iItem</td><td>Item</td></tr> <tr> <td>iSite</td><td>Site</td></tr> </table> Output: <table> <tr> <td>oExceptionMessage</td><td>The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID</td></tr> <tr> <td>oExceptionID</td><td>An ID that refers to all error information. Use the functions in Exception to get all relevant information.</td></tr> </table> Return: 0 Session started <> 0 An error occurred	iItem	Item	iSite	Site	oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID	oExceptionID	An ID that refers to all error information. Use the functions in Exception to get all relevant information.
iItem	Item								
iSite	Site								
oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID								
oExceptionID	An ID that refers to all error information. Use the functions in Exception to get all relevant information.								

Public Interfaces for ItemPurchaseBusinessPartner

The following functions are available:

[ItemPurchaseBusinessPartner.CalculateLeadTimes](#)

[ItemPurchaseBusinessPartner.GetSafetyTime](#)

[ItemPurchaseBusinessPartner.StartDetail](#)

[ItemPurchaseBusinessPartner.StartOverview](#)

ItemPurchaseBusinessPartner.CalculateLeadTimes

DLL:	tdextipuapi
Syntax:	<pre> long ItemPurchaseBusinessPartner.CalculateLeadTimes(domain tccom.bpid iBuyFromBusinessPartner, domain tccom.bpid iShipFromBusinessPartner, domain tccom.cadr iShipFromAddress, domain tccom.cadr iShipToAddress, domain tcitem iItem, domain tcncmp iLogisticCompany, domain tcsite iSite, domain tcwtm iSupplyTime, domain tcto iSupplyTimeUnit, domain tcwtm iFullSupplyTime, domain tcto iFullSupplyTimeUnit, domain tcwtm iInternalProcessTime, domain tcto iInternalProcessTimeUnit, domain tcwtm iSafetyTime, domain tcto iSafetyTimeUnit, domain tccfrw iCarrier, domain tcmcs.long6 iLeadTime, domain tcmcs.long6 iFullLeadTime, ref domain tcmcs.long6 oLeadTime, ref domain tcmcs.long6 oFullLeadTime, ref domain tcmcs.long6 oTransportationTime, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function calculates the lead times (normal lead time and full lead time) and transportation time in days for the given combination of purchased item and business partner. Pre: Not Applicable. Post: Not Applicable. Input: - iBuyFromBusinessPartner - Buy-from Business Partner (Mandatory) - iShipFromBusinessPartner - Ship-from Business Partner (not Mandatory) - iShipFromAddress - Ship-from Address. If filled this address will be used in transportation time calculation. - iShipToAddress - Ship-to Address. If filled this address will be used in transportation time calculation. - iItem - Item (Mandatory) - iLogisticCompany - Logistic company (Mandatory) - iSite - Site (not Mandatory) - iSupplyTime - Supply Time (not Mandatory) - iSupplyTimeUnit - Supply Time Unit (Mandatory) Possible values: - Hours - Days - iFullSupplyTime - Full Supply Time (not Mandatory) - iFullSupplyTimeUnit - Full Supply Time Unit (Mandatory) Possible values: - Hours - Days - iInternalProcessTime - Internal Process Time (not Mandatory) - iInternalProcessTimeUnit - Internal Process Time Unit (Mandatory) Possible values: - Hours - Days - iSafetyTime - Safety Time (not Mandatory) - iSafetyTimeUnit - Safety Time Unit (Mandatory)

	Possible values: - Hours - Days iCarrier - Carrier (not Mandatory) iLeadTime - Current Lead Time (not Mandatory) iFullLeadTime - Current Full Lead Time (not Mandatory) Output: oLeadTime - Calculated Lead Time oFullLeadTime - Calculated Full Lead Time oTransportationTime - Calculated Transit Time oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Calculate Leadtimes finished <> 0 - Not all mandatory input arguments are filled.

ItemPurchaseBusinessPartner.GetSafetyTime

DLL:	tdextipuapi
	This function is available from KB3514344 .
Syntax:	<pre> long ItemPurchaseBusinessPartner.GetSafetyTime(domain tccitg iItemGroup, domain tcitem iItem, domain tccom.bpid iBuyFromBusinessPartner, domain tccom.bpid iShipFromBusinessPartner, domain tcddate iEffectiveDate, domain tcsite iSite, ref domain tcwtm oSafetyTime, ref domain tcto oSafetyTimeUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function returns the safety time and safety time unit for the specified Buy-from/Ship-from Business Partner and Item combination. Pre: Not Applicable. Post: Not Applicable. Input : - iItemGroup - Item Group (Mandatory if Item is not filled) - iItem - Item (Mandatory if Item Group is not filled) - iBuyFromBusinessPartner - Buy-from Business Partner (Mandatory) - iShipFromBusinessPartner - Ship-from Business Partner (Optional) - iEffectiveDate - Effective Date (Mandatory, use utc.num() if the current date is needed) - iSite - Site (Optional) Output: - oSafetyTime - Safety Time - oSafetyTimeUnit - Safety Time Unit - oExceptionMessage - The last message if any message is

	found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - Data read - An error occurred

ItemPurchaseBusinessPartner.StartDetail

DLL:	tdextipuapi														
Syntax:	<pre>long ItemPurchaseBusinessPartner.StartDetail(long iStartMode, domain tccitg iItemGroup, domain tcitem iItem, domain tccom.bpid iBuyFromBusinessPartner, domain tccom.bpid iShipFromBusinessPartner, domain tcefex.date iEffectiveDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>														
Usage:	Expl: This function starts the detail session Items - Purchase Business Partner (tdipu0110m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key. If the primary key cannot be found in the table, an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: <table> <tr> <td>iItemGroup</td><td>Item Group</td></tr> <tr> <td>iItem</td><td>Item</td></tr> <tr> <td>iBuyFromBusinessPartner</td><td>Buy-from Business Partner</td></tr> <tr> <td>iShipFromBusinessPartner</td><td>Ship-from Business Partner</td></tr> <tr> <td>iEffectiveDate</td><td>Effective Date</td></tr> </table> Output: <table> <tr> <td>oExceptionMessage</td><td>The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID</td></tr> <tr> <td>oExceptionID</td><td>An ID that refers to all error information. Use the functions in Exception to get all relevant</td></tr> </table>	iItemGroup	Item Group	iItem	Item	iBuyFromBusinessPartner	Buy-from Business Partner	iShipFromBusinessPartner	Ship-from Business Partner	iEffectiveDate	Effective Date	oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID	oExceptionID	An ID that refers to all error information. Use the functions in Exception to get all relevant
iItemGroup	Item Group														
iItem	Item														
iBuyFromBusinessPartner	Buy-from Business Partner														
iShipFromBusinessPartner	Ship-from Business Partner														
iEffectiveDate	Effective Date														
oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID														
oExceptionID	An ID that refers to all error information. Use the functions in Exception to get all relevant														

	Return: 0 <> 0	information. Session started An error occurred

ItemPurchaseBusinessPartner.StartOverview

DLL:	tdextipuapi
Syntax:	<pre> long ItemPurchaseBusinessPartner.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccitg iItemGroup, domain tcitem iItem, domain tccom.bpid iBuyFromBusinessPartner, domain tccom.bpid iShipFromBusinessPartner, domain tcefex.date iEffectiveDate, ref domain tccitg oItemGroup, ref domain tcitem oItem, ref domain tccom.bpid oBuyFromBusinessPartner, ref domain tccom.bpid oShipFromBusinessPartner, ref domain tcefex.date oEffectiveDate, ref domain tcmcs.s999m oExceptionMessage mb, ref domain long oExceptionID) </pre>
Usage:	Expl: This function starts the session Items - Purchase Business Partner in overview mode (tdipu0110m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used iSessionIndex The index that will be used. Supported values: 1: sort by Item Group/Item 2: sort by Buy-from Business Partner/ Ship-from Business Partner 3: sort by Approved Supplier List iQueryExtend A specific query to be used when zooming to this session. iItemGroup Item Group iItem Item iBuyFromBusinessPartner Buy-from Business Partner (Mandatory if iStartMode = MODELESS and iSessionIndex = 2) iShipFromBusinessPartner Ship-from Business Partner iEffectiveDate Effective Date Output: Variables below contain the values of the selected record. They are only filled if iStartMode is MODAL and 1 record has been selected.

	oItemGroup	Item Group
	oItem	Item
	oBuyFromBusinessPartner	Buy-from Business Partner
	oShipFromBusinessPartner	Ship-from Business Partner
	oEffectiveDate	Effective Date
	oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
	oExceptionID	An ID that refers to all error information. Use the functions in Exception to get all relevant information.
Return: 0		Session started
<> 0		An error occurred

Public Interfaces for ItemsByManufacturerPartNumber

The following functions are available:

[ItemsByManufacturerPartNumber.StartOverview](#)

ItemsByManufacturerPartNumber.StartOverview

DLL:	tdextipuapi
	This function is available from KB3556200 .
Syntax:	<pre>long ItemsByManufacturerPartNumber.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcmpnr iManufacturerPartNumber, domain tcmcs.cmnf iManufacturer, domain tcitem iItem, ref domain tcmpnr oManufacturerPartNumber, ref domain tcmcs.cmnf oManufacturer, ref domain tcitem oItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>

Usage:	Expl: This function starts session Items by MPN (tdipu0149m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used iSessionIndex Specifies the session-index that is to be used. Supported values: 1: sort by Manufacturer Part Number, Manufacturer, Item (default) 2: sort by Item, Manufacturer Part Number, Manufacturer iQueryExtend A specific query to be used when zooming to this session. iManufacturerPartNumber Manufacturer Part Number Mandatory if iStartMode = MODELESS and session-index 1 is used. iManufacturer Manufacturer Mandatory if iStartMode = MODELESS and session-index 1 is used. iItem Item Mandatory if iStartMode = MODELESS and session-index 2 is used. Output: for iStartMode MODAL: oManufacturerPartNumber The selected Manufacturer Part Number oManufacturer The selected Manufacturer oItem The selected Item oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

Public Interfaces for ItemSalesBusinessPartner

The following functions are available:

[ItemSalesBusinessPartner.StartDetail](#)

[ItemSalesBusinessPartner.StartOverview](#)

ItemSalesBusinessPartner.StartDetail

DLL:	tdextisaapi
	This function is available from KB2300144 .
Syntax:	<pre> long ItemSalesBusinessPartner.StartDetail(long iStartMode, domain tccitg iItemGroup, domain tcitem iItem, domain tccom.bpid iSoldToBusinessPartner, domain tccom.bpid iShipToBusinessPartner, domain tcefex.date iEffectiveDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Items - Sales Business Partner Details (tdisa0510m000). Input: iStartMode Specifies the start mode for the session (Mandatory). Possible values are: MODAL The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS Parent and child are parallel sessions that can be manipulated simultaneously. iItemGroup Item Group (Mandatory if argument iItem is not specified). iItem Item (Mandatory if argument iItemGroup is not specified). iSoldToBusinessPartner Sold-to Business Partner (Mandatory) iShipToBusinessPartner Ship-to Business Partner (Optional). iEffectiveDate Effective date (Mandatory). Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

ItemSalesBusinessPartner.StartOverview

DLL:	tdextisaapi
	This function is available from KB2300144 .
Syntax:	<pre> long ItemSalesBusinessPartner.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, </pre>

Public Interfaces for Item

	<pre> const string iQueryExtend(), domain tccitg iItemGroup, domain tcitem iItem, domain tccom.bpid iSoldToBusinessPartner, domain tccom.bpid iShipToBusinessPartner, domain tcefex.date iEffectiveDate, ref domain tccitg oItemGroup, ref domain tcitem oItem, ref domain tccom.bpid oSoldToBusinessPartner, ref domain tccom.bpid oShipToBusinessPartner, ref domain tcefex.date oEffectiveDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Items - Sales Business Partner Overview (tdisa0510m000). Input: iStartMode Specifies the start mode for the session (Mandatory). Possible values are: MODAL The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Start Filter, not used. iSessionIndex The table index that will be used (Optional). Supported values: 1: sort by Item Group, Item 2: sort by Sold-to Business Partner, Ship-to Business Partner 3: sort by Item, Item Group iQueryExtend A specific query to be used when zooming to this session (Optional). iItemGroup Item Group (Optional). iItem Item (Optional). iSoldToBusinessPartner Sold-to Business Partner (Optional) iShipToBusinessPartner Ship-to Business Partner (Optional). iEffectiveDate Effective date (Optional). Output: Variables below contain the values of the selected record, they are only filled if iStartMode is MODAL and 1 record has been selected: oItemGroup Item Group oItem Item oSoldToBusinessPartner Sold-to Business Partner oShipToBusinessPartner Ship-to Business Partner oEffectiveDate Effective Date Variables for API error handling: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

Public Interfaces for ItemSalesByOffice

The following functions are available:

[ItemSalesByOffice.StartOverview](#)

ItemSalesByOffice.StartOverview

DLL:	tdextisaapi
	This function is available from KB3538855 .
Syntax:	<pre> long ItemSalesByOffice.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, domain tccwoc iSalesOffice, ref domain tcitem oItem, ref domain tccwoc oSalesOffice, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Items - Sales by Office (tdisa0181m000) in Overview mode. Input: iStartMode Specifies the start mode for the session (Mandatory). Possible values are: MODAL The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Start Filter, not used. iSessionIndex The table index that will be used (Optional). Supported values: 1: sort by Item, Sales Office 2: sort by Sales Office, Item iQueryExtend A specific query to be used when zooming to this session (Optional). iItem Item (Mandatory if iSessionIndex equals 1 and iStartMode equals MODELESS). iSalesOffice Sales Office (Optional) Output: Variables below contain the values of the selected record, they are only filled if iStartMode is MODAL and 1 record has been selected: oItem Item oSalesOffice Sales Office Variables for API error handling:

	oExceptionMessage	The last message if any message is found. If more than one message is given, these are present in the oExceptionID.
	oExceptionID	An ID that refers to the exception information. Use the functions in Exception to get all relevant information.
	Return: 0	Session started
	<> 0	An error occurred

Public Interfaces for ItemPlanning

The following functions are available:

[ItemPlanning.StartDetail](#)

[ItemPlanning.StartOverview](#)

ItemPlanning.StartDetail

DLL:	cpextrpdapi
	This function is available from KB2303593 .
Syntax:	<pre>long ItemPlanning.StartDetail(long iStartMode, domain cpitem iPlanItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the session Items Planning (cprpd1100m000) in deatil mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Primary Key Fields: iPlanItem Plan Item - Mandatory Output: oExceptionMessage The last message if any message is found. If more than one message is found, these are present in the

	oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Session started Otherwise.
	Return: 0 <> 0

ItemPlanning.StartOverview

DLL:	cpextrpdapi
	This function is available from KB2303593 .
Syntax:	<pre>long ItemPlanning.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain cpitem iPlanItem, domain cpcom.plvl iPlanLevel, domain tccprj iProject, domain tceemm.clus iCluster, domain tcitem iItem, ref domain cpitem oPlanItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the session Items Planning (cprpd1100m000) in overview mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Specifies the table-index that is to be used. Optional. Standard supported values: 1: sort by Plan Item (default) 2: sort by Plan Level, Plan Item 7: sort by Project, Plan Level, Plan Item 8: sort by Planning Cluster, Item 9: sort by Item, Planning Cluster iQueryExtend A specific query to be used when zooming to this session.(Optional) iPlanItem Plan Item - Optional, used when Session Index is 1. iPlanLevel Plan Level - Optional, used when Session Index is 2. iProject Project - Optional, used when Session Index is 7. iCluster Cluster - Optional, used when Session

	Index is 8. Item - Optional, used when Session Index is 9. Output: Variables below contain the values of the selected record. They are only filled if iStartMode is MODAL and 1 record has been selected. oPlanItem Plan Item oExceptionMessage The last message if any message is found. If more than one message is found, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started. <> 0 Otherwise.

Public Interfaces for ItemSerial

The following functions are available:

[ItemSerial.CreateOrUpdate](#)

[ItemSerial.SetStatus](#)

ItemSerial.CreateOrUpdate

DLL:	tcextibdapi
	This function is available from KB3628975 .
Syntax:	<pre>long ItemSerial.CreateOrUpdate(domain tcitem iItem, domain tcibd.sern iSerialNumber, domain tcclot iLot, domain tcuef.effn iEffectivityUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This Public Interface updates an exiting item serial number with the lot and effectivity unit. Only items with item type Product and Rental Product can be updated. If the serial does not exist it will be created. Pre: db.retry.point() Post: abort or commit transaction

	<pre> iItem - Item (Mandatory). iSerialNumber - Serial Number (Mandatory). iLot - Lot (Optional). iEffectivityUnit - Effectivity Unit (Optional). If Item is lot controlled, Effectivity Unit will be taken from Lot. In this case the value must be 0 (zero). Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Successfully updated. <> 0 - Otherwise. </pre>

ItemSerial.SetStatus

DLL:	tcextibdapi This function is available from KB2331329.
Syntax:	<pre> long ItemSerial.SetStatus(domain tcitem iItem, domain tcibd.sern iSerialNumber, domain tcibd.ssts iStatus, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface updates the status of Item Serial numbers. Pre: retry.point must be set Post: transaction must be committed or aborted Input: <pre> iItem - Item (Mandatory). iSerialNumber - Serial Number (Mandatory). iStatus - Status (Mandatory). Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Successfully updated. <> 0 - Otherwise. </pre>

Public Interfaces for LotAndSerialSet

The following functions are available:

[LotAndSerialSet.StartOverview](#)

LotAndSerialSet.StartOverview

DLL:	tcextibdapi	
	This function is available from KB2274414 .	
Syntax:	<pre> long LotAndSerialSet.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcguid iLotAndSerialSet, domain tcitem iItem, boolean iDisplayOnly, ref domain tcguid oLotAndSerialSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>	
Usage:	Expl: Starts the session Lot and Serial Set (tcibd4111m000) in overview mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used. iSessionIndex Not used. iQueryExtend Not used. iLotAndSerialSet Optional when iDisplayOnly is false. iItem Mandatory, must exist. iDisplayOnly Mandatory. Output: oLotAndSerialSet - The newly created set in the case that iDisplayOnly = false and iLotAndSerialSet was empty. Only has a value if iStartMode = MODAL. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is	

	given, these are present in the oExceptionID. - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Session started. - Otherwise.
	oExceptionID
	Return: 0 <> 0

Public Interfaces for PlanningBillOfCriticalMaterial

The following functions are available:

[PlanningBillOfCriticalMaterial.Generate](#)

PlanningBillOfCriticalMaterial.Generate

DLL:	cpextrpdapi				
	This function is available from KB3534322 .				
Syntax:	<pre>long PlanningBillOfCriticalMaterial.Generate(domain cpitem iPlanItem, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>				
Usage:	Expl: This function can be used to Generate Bill of Critical Material. This Public Interface is similar to Generate Bill of Critical Material option in session cprpd3220m000. This function makes use of a Processing Option Set, which can be created via a call to ProcessingOptionSet.Create(), and cleaned up after use, via a call to ProcessingOptionSet.Delete(). Transaction management will be handled by this public interface. Pre: N.A. Post: N.A. Input: <table> <tr> <td>iPlanItem</td><td>Plan Item</td></tr> <tr> <td>iProcessingOptionSet</td><td>A Processing Option Set can be created via a call to ProcessingOptionSet.Create(). If 0, then user default/session default values are applied.</td></tr> </table> Processing Options have a direct relationship with the form fields on session Generate Bill of Critical Material (cprpd3220m000) and are not explained in further detail here. Please refer to the session help for additional information.	iPlanItem	Plan Item	iProcessingOptionSet	A Processing Option Set can be created via a call to ProcessingOptionSet.Create(). If 0, then user default/session default values are applied.
iPlanItem	Plan Item				
iProcessingOptionSet	A Processing Option Set can be created via a call to ProcessingOptionSet.Create(). If 0, then user default/session default values are applied.				

Browse Order Pegging options which are not available as Processing Options will get defaulted in accordance with the session logic.		
NAME	TYPE	DEFAULT
PrintingDevice	domain tcmcs.str14	""
PrintingFileoutPathAndName	domain tcmcs.str100	""
ErrorReport	domain tcyesno	tcyesno.no
ProcessAndErrorReport	domain tcyesno	If ErrorReport is Yes then ProcessAndErrorReport is set No.
Output: oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.	
oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.	
Return: 0	- Successfully Generated.	
<> 0	- Otherwise.	

Public Interfaces for PlanningBillOfCriticalCapacities

The following functions are available:

[PlanningBillOfCriticalCapacities.Generate](#)

PlanningBillOfCriticalCapacities.Generate

DLL:	cpextrpdapi
	This function is available from KB3534322 .
Syntax:	<pre>long PlanningBillOfCriticalCapacities.Generate(domain citem iPlanItem, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function can be used to Generate Bill of Critical Capacities. This Public Interface is similar to Generate Bill of Critical Capacities option in session cprpd3220m000. This function makes use of a Processing Option Set, which can be created via a call to ProcessingOptionSet.Create(), and cleaned up after use, via a call to ProcessingOptionSet.Delete(). Transaction management will

be handled by this public interface.		
Pre:	N.A.	
Post:	N.A.	
Input:		
	iPlanItem	Plan Item
	iProcessingOptionSet	A Processing Option Set can be created via a call to ProcessingOptionSet.Create(). If 0, then user default/session default values are applied.
Processing Options have a direct relationship with the form fields on session Generate Bill of Critical Material (cprpd3220m000) and are not explained in further detail here. Please refer to the session help for additional information. Browse Order Pegging options which are not available as Processing Options will get defaulted in accordance with the session logic.		
NAME	TYPE	DEFAULT
PrintingDevice	domain tcmcs.str14	""
PrintingFileoutPathAndName	domain tcmcs.str100	""
ErrorReport	domain tcyesno	tcyesno.no
ProcessAndErrorReport	domain tcyesno	If ErrorReport is Yes then ProcessAndErrorReport is set No.
Output: oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.	
	oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.
Return: 0	- Successfully Generated.	
<> 0	- Otherwise.	

Chapter 7 Public Interfaces for Engineering Data Management

Public Interfaces for EngineeringItemRevision

The following functions are available:

[EngineeringItemRevision.ApproveByProduction](#)

[EngineeringItemRevision.CopyToItem](#)

EngineeringItemRevision.ApproveByProduction

DLL:	tiextedmapi This function is available from KB2300366 .
Syntax:	<pre> long EngineeringItemRevision.ApproveByProduction(domain tcitem iEngineeringItem, domain tibmr iRevision, boolean iCopyRelationsFromPreviousRevision, boolean iOverwriteDescription, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function is used to approve by production the specified Engineering Item Revision. Note that this function will continue processing when: - no relation is present for the Item Engineering Revision - the Effective Date of the Item Engineering Revision is before the current date. Also note that this function will do <code>abort.transaction()</code> in some cases. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: <code>iEngineeringItem</code> - Engineering Item (Mandatory). <code>iRevision</code> - Revision (Mandatory). <code>iCopyRelationsFromPreviousRevision</code> - Control for copying relations from previous revision. <code>iOverwriteDescription</code> - Control for overwriting descriptions

Output: oExceptionMessage	and/or search keys.
oExceptionID	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.
Return: 0	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.
<> 0	- Engineering Item Revision is Approved by Production.
	- Engineering Item Revision is not Approved by Production.

EngineeringItemRevision.CopyToItem

DLL:	tiextedmapi										
	This function is available from KB2300366 .										
Syntax:	<pre> long EngineeringItemRevision.CopyToItem(domain tcitem iEngineeringItem, domain tiedm.revi iRevision, domain tcitem iTargetItem, domain tccitg iTargetItemGroup, boolean iOverwriteItem, boolean iOverwriteDescription, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>										
Usage:	Expl: This function is used to copy E-Item to Item (tcibd001 Items - General Data). The Item data is created if the Item is not already present. If the Item is present, then it is overwritten by the E-Item Revision data. Note: Below checks are executed before copying engineering item to items: - Engineering Data Management should be active and Engineering Revision must be implemented in Implemented Software Components and Engineering Data Management Parameters must be present. - Engineering Item Revision status must be Approved by Production. - Engineering Item to copied can only have relations with items for the same project. - If Target Item is present in Items then the target Item must be a Product or an Engineering Module. - If Target Item is not present in Items then valid Item Defaults must be present for Target Item to be created. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: <table> <tr> <td>iEngineeringItem</td><td>- Engineering Item (Mandatory).</td></tr> <tr> <td>iRevision</td><td>- Revision (Mandatory).</td></tr> <tr> <td>iTargetItem</td><td>- Target Item (Mandatory).</td></tr> <tr> <td>iTargetItemGroup</td><td>- Target Item Group (Mandatory).</td></tr> <tr> <td>iOverwriteItem</td><td>- Control for overwriting existing item</td></tr> </table>	iEngineeringItem	- Engineering Item (Mandatory).	iRevision	- Revision (Mandatory).	iTargetItem	- Target Item (Mandatory).	iTargetItemGroup	- Target Item Group (Mandatory).	iOverwriteItem	- Control for overwriting existing item
iEngineeringItem	- Engineering Item (Mandatory).										
iRevision	- Revision (Mandatory).										
iTargetItem	- Target Item (Mandatory).										
iTargetItemGroup	- Target Item Group (Mandatory).										
iOverwriteItem	- Control for overwriting existing item										

	iOverwriteDescription Output: oExceptionMessage oExceptionID Return: 0 <> 0	data. - Control for overwriting descriptions and/or search keys. - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. - Copy E-Item to Item is successful. - Copy E-Item to Item is not successful.

Chapter 8 Public Interfaces for CRM

Public Interfaces for Opportunity

The following functions are available:

[Opportunity.StartMultiMain](#)

Opportunity.StartMultiMain

DLL:	tdextsmiapi
Syntax:	<pre> long Opportunity.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcopty iOpportunity, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the Multi-Main session Opportunity (tdsmi1610m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iQueryExtend A specific query to be used when zooming to this session. iOpportunity Opportunity Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in

	Return: 0 <> 0	Exception to get all relevant information. Session started An error occurred

Public Interfaces for Opportunities

The following functions are available:

[Opportunities.StartOverview](#)

Opportunities.StartOverview

DLL:	tdextsmiapi
	This function is available from KB3522389 .
Syntax:	<pre> long Opportunities.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcopty iOpportunity, domain tcemno iAssignedTo, domain tccom.bpid iSoldToBusinessPartner, ref domain tcopty oOpportunity, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session Opportunities Overview (tdsm1110m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used iSessionIndex Specifies the table-index that is to be used. Supported values: 1: sort by Opportunity (default) 5: sort by Assigned to 6: sort by Sold-to Business Partner

	iQueryExtend A specific query to be used when zooming to this session. iOpportunity Opportunity (Optional) iAssignedTo Assigned to (Optional) iSoldToBusinessPartner Sold-to Business Partner (Optional) Output: for iStartMode MODAL: oOpportunity The selected Opportunity oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

Chapter 9 Public Interfaces for Sales

Public Interfaces for Sales

The following functions are available:

[Sales.CalculatePlannedDeliveryDate](#)

[Sales.GenerateRetrobilledPriceChangeAdvices](#)

[Sales.GetContractSettings](#)

[Sales.GetGeneralSettings](#)

[Sales.GetOrderSettings](#)

[Sales.GetQuoteSettings](#)

[Sales.GetScheduleSettings](#)

[Sales.StartProcessProFormalInvoices](#)

Sales.CalculatePlannedDeliveryDate

DLL:	tdextslsapi		
	This function is available from KB2310725 .		
Syntax:	<pre>long Sales.CalculatePlannedDeliveryDate(domain tcdte iPlannedReceiptDate, domain tcdte iOrderDate, domain tcitem iItem, domain tcsite iSite, domain tccwoc iSalesOffice, domain tccom.bpid iSoldToBusinessPartner, domain tccom.bpid iShipToBusinessPartner, domain tccwar iWarehouse, domain tccwoc iWorkcenter, domain tccwar iShipToWarehouse, domain tccom.cadr iShipToAddress, domain tcorno iSalesOrder, domain tcpono iSalesOrderLine,</pre>		

	domain tcpono iSalesOrderLineSequence, domain tcqsl1 iOrderQuantity, domain tcconv iOrderQuantityConvFactorToInvUnit, domain tccrte iRoute, domain tcmcs.serv iServiceLevel, domain tccdec iTermsOfDelivery, domain tcptpa iPointOfTitlePassage, domain tccfrw iPreferredCarrier, domain tccfrw iCarrier, ref domain tcdate oPlannedDeliveryDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function calculates the sales planned delivery date based on the planned receipt date (date at customer's site). For the calculation, the planned receipt date is reduced with the transportation time between the address of the sales office or supplier warehouse and the ship-to address. If the item is filled, applicable if a calculation for a sales order line is done, a check is done if a pattern can be found in Terms and conditions. If so the planned delivery date is updated by the first preceeding delivery moment of the pattern. Finally, the planned delivery date is shifted backward to the availability of the warehouse address calendar (calculated planned delivery date must be a workable date in the warehouse calendar). Pre: Not applicable Post: Not applicable Input: iPlannedReceiptDate - Planned Receipt Date (mandatory) iOrderDate - Order Date iItem - Item iSite - Site iSalesOffice - Sales Office iSoldToBusinessPartner - Sold-to Business Partner iShipToBusinessPartner - Ship-to Business Partner iWarehouse - Warehouse iWorkcenter - Workcenter iShipToWarehouse - Ship-to Warehouse iShipToAddress - Ship-to Address iSalesOrder - Sales Order Needed for direct delivery iSalesOrderLine - Sales Order Line Mandatory if Sales Order is filled. Needed for direct delivery. iSalesOrderLineSequence - Sales Order Line Sequence iOrderQuantity - Order Quantity iOrderQuantityConvFactorToInvUnit - Order Quantity Conversion Factor to Inventory Unit (mandatory). This field must be filled with 1.0 if no quantity is given. iRoute - Route iServiceLevel - Service Level iTermsOfDelivery - Terms of Delivery iPointOfTitlePassage - Point of Title Passage iPreferredCarrier - Preferred Carrier iCarrier - Carrier Output: oPlannedDeliveryDate - The calculated Planned Delivery Date. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception

	information. Use the functions in Exception to get all relevant information. Return: 0 -> No error <> 0 -> Error occurred

Sales.GenerateRetrobilledPriceChangeAdvices

DLL:	tdextslsapi
Syntax:	<pre> long Sales.GenerateRetrobilledPriceChangeAdvices (domain tdsls.updn iAdviceNumber, domain tcyesno iSeparateAdviceLinesForReturnOrders, boolean iGenerateForOrders, boolean iGenerateForSchedules, boolean iGenerateForShipments, domain tcorno iSalesOrderFrom, domain tcorno iSalesOrderTo, domain tcorno iSalesScheduleFrom, domain tcorno iSalesScheduleTo, domain tdshpm iShipmentFrom, domain tdshpm iShipmentTo, domain tccono iContractFrom, domain tccono iContractTo, domain tcpono iContractLineFrom, domain tcpono iContractLineTo, domain tccom.bpid iSoldToBusinessPartnerFrom, domain tccom.bpid iSoldToBusinessPartnerTo, domain tccom.bpid iShipToBusinessPartnerFrom, domain tccom.bpid iShipToBusinessPartnerTo, domain tccom.bpid iInvoiceToBusinessPartnerFrom, domain tccom.bpid iInvoiceToBusinessPartnerTo, domain tcdate iDateFrom, domain tcdate iDateTo, domain tdsls.corg iOrderOriginFrom, domain tdsls.corg iOrderOriginTo, domain tcitem iItemFrom, domain tcitem iItemTo, domain tcccur iCurrency, domain tdsls.utyp iUpdateType, domain tcamnt iUpdateAmount, domain tcprcg iUpdatePercentage, domain tcyesno iOnlyRetrobillingItems, ref domain tdsls.updn oAdviceNumber, ref long oNumberOfAdviceLines, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function will generate Retro-Billed Advices* and Advice Lines based on the given input parameters. The price change is calculated from the given Update Type, Update Amount and Update Percentage. * If the given Advice Number (iAdviceNumber) is a series, then a new Advice header will be created based upon that series. If it is an existing Advice, then the given Advice will be used. Pre: Caller must set retry-point

Post:	Caller must commit/abort transaction	
Input:	- iAdviceNumber	- The ID of the Advice Header Can be a series or an existing Advice ID.
	- iSeparateAdviceLinesForReturnOrders	-> Yes/No Create separate advice lines for linked sales return sales order (invoice) lines.
	- iGenerateForOrders	- True/False Indicates if Retro Billed advices must be generated for orders
	- iGenerateForSchedules	- True/False Indicates if Retro Billed advices must be generated for schedules
	- iGenerateForShipments	- True/False Indicates if Retro Billed advices must be generated for receipts
	- iSalesOrderFrom	- Sales Order from
	- iSalesOrderTo	- Sales Order to
	- iSalesScheduleFrom	- Sales Schedule from
	- iSalesScheduleTo	- Sales Schedule to
	- iShipmentFrom	- Shipment from
	- iShipmentTo	- Shipment to
	- iContractFrom	- Contract from
	- iContractTo	- Contract to
	- iContractLineFrom	- Contract Line from
	- iContractLineTo	- Contract Line to
	- iSoldToBusinessPartnerFrom	- Sold-to Business Partner from
	- iSoldToBusinessPartnerTo	- Sold-to Business Partner to
	- iShipToBusinessPartnerFrom	- Ship-to Business Partner from
	- iShipToBusinessPartnerTo	- Ship-to Business Partner to
	- iInvoiceToBusinessPartnerFrom	- Invoice-to Business Partner from
	- iInvoiceToBusinessPartnerTo	- Invoice-to Business Partner to
	- iDateFrom	- Date from
	- iDateTo	- Date to
	- iOrderOriginFrom	- Order Origin from
	- iOrderOriginTo	- Order Origin to
	- iItemFrom	- Item from
	- iItemTo	- Item to
	- iCurrency	- Currency
	- iUpdateType	- Update Type The method used to update the existing price of the sales document. Possible values: - Value: The price update is a fixed amount. - Percentage: The price update is a percentage.
	- iUpdateAmount	- Update Amount The value by which the existing price of the specified sales documents is updated.
	- iUpdatePercentage	- Update Percentage The percentage by which the existing price of the specified sales documents is updated.
	- iOnlyRetrobillingItems	-> Yes/No Only process Retrobilling items
Output:	- oAdviceNumber	- The ID of the Advice Header that is used. If the given Advice (iAdviceNumber) did not exist yet, then this

	argument contains the created Advice. - oNumberOfAdviceLines - Number of Advice Lines generated - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - Function was executed successfull. Price change advices may have been generated. - An error occurred during the execution of the function.

Sales.GetContractSettings

DLL:	tdextslsapi
Syntax:	<pre>long Sales.GetContractSettings(domain tcncmp iLogisticCompany, domain tccwoc iSalesOffice, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...)</pre>
Usage:	Expl: This function retrieves settings for sales contracts. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tdsls000), or office or site level (settings table tdsls083) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when sales contract series and step size for sales contract lines are needed, the function must be called as follows: <pre>if Sales.GetContractSettings(* Fixed arguments: company, --> input</pre>

	<pre> sales office, --> input site, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output * Variable arguments: "sesc", --> input series.for.sales.contracts, --> output "sssc", --> input step.size.for.sales.contract.lines) <> 0 then --> output * Error, do something Exception.Delete(exception.id) endif Alternatively, the function can also be called with "sesc.3" and "sssc.3", respectively, which refers to the mnemonics at parameter level. Pre: None Post: None Input: iLogisticCompany - Logistic Company: Mandatory iSite - Site: Not Mandatory iSalesOffice - Sales Office: Not Mandatory iForceRead - Option to force new query in stead of using cached information iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - An error occurred </pre>

Sales.GetGeneralSettings

DLL:	tdextslsapi
Syntax:	<pre> long Sales.GetGeneralSettings(domain tcncmp iLogisticCompany, domain tccwoc iSalesOffice, </pre>

		domain tcsite boolean boolean ref domain tcmcs.s999m ref long	iSite, iForceRead, iContextIsMasterData, oExceptionMessage mb, oExceptionID, ...)												
Usage:	Expl:	This function retrieves general settings for sales. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tdsls000), or office or site level (settings table tdsls080) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when sales list group and configutaion date for PCS are needed, the function must be called as follows: <pre>if Sales.GetGeneralSettings(* Fixed arguments: company, --> input sales office, --> input site, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output * Variable arguments: "clgr", --> input list.group.sales, --> output "codt", --> input configuration.date.pcs) <> 0 then --> output * Error, do something Exception.Delete(exception.id) endif</pre> Pre: None Post: None Input: <table><tr><td>iLogisticCompany</td><td>- Logistic Company: Mandatory</td></tr><tr><td>iSite</td><td>- Site: Not Mandatory</td></tr><tr><td>iSalesOffice</td><td>- Sales Office: Not Mandatory</td></tr><tr><td>iForceRead</td><td>- Option to force new query in stead of using cached information</td></tr><tr><td>iContextIsMasterData</td><td>- True/False: Indicates if call is done to retrieve data for master data or transactional data.</td></tr><tr><td>...</td><td>- The field mnemonic of the required field.</td></tr></table> Output:		iLogisticCompany	- Logistic Company: Mandatory	iSite	- Site: Not Mandatory	iSalesOffice	- Sales Office: Not Mandatory	iForceRead	- Option to force new query in stead of using cached information	iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.	...	- The field mnemonic of the required field.
iLogisticCompany	- Logistic Company: Mandatory														
iSite	- Site: Not Mandatory														
iSalesOffice	- Sales Office: Not Mandatory														
iForceRead	- Option to force new query in stead of using cached information														
iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.														
...	- The field mnemonic of the required field.														

	<pre> oExceptionMessage oExceptionID ... Return: 0 <> 0 </pre>	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - The value of the required field. - Data read - An error occurred

Sales.GetOrderSettings

DLL:	tdextslsapi
Syntax:	<pre> long Sales.GetOrderSettings(domain tcncmp iLogisticCompany, domain tccwoc iSalesOffice, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves settings for sales orders. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tdsls000), or office or site level (settings table tdsls084) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when sales order series and step size for sales order lines are needed, the function must be called as follows: <pre> if Sales.GetOrderSettings(* Fixed arguments: company, --> input sales office, --> input site, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output </pre>

	<pre> * Variable arguments: "seso", --> input series.for.sales.orders, --> output "ssso", --> input step.size.for.sales.order.lines) <> 0 then --> output * Error, do something Exception.Delete(exception.id) endif Alternatively, the function can also be called with "seso.4" and "ssso.4", respectively, which refers to the mnemonics at parameter level. Pre: None Post: None Input: iLogisticCompany - Logistic Company: Mandatory iSite - Site: Not Mandatory iSalesOffice - Sales Office: Not Mandatory iForceRead - Option to force new query in stead of using cached information iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - An error occurred </pre>

Sales.GetQuoteSettings

DLL:	tdextslsapi
Syntax:	<pre> long Sales.GetQuoteSettings (domain tcncmp iLogisticCompany, domain tccwoc iSalesOffice, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>

Usage:	Expl: This function retrieves settings for sales quotes. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tdsls000), or office or site level (settings table tdsls081) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when sales quote series and step size for sales quote lines are needed, the function must be called as follows: <pre> if Sales.GetQuoteSettings(/* Fixed arguments: company, --> input sales office, --> input site, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output /* Variable arguments: "sesq", --> input series.for.sales.quotes, --> output "sssq", --> input step.size.for.sales.quote.lines) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif </pre> Alternatively, the function can also be called with "sesq.1" and "sssq.1", respectively, which refers to the mnemonics at parameter level. Pre: None Post: None Input: <table border="0"> <tr> <td>iLogisticCompany</td><td>- Logistic Company: Mandatory</td></tr> <tr> <td>iSite</td><td>- Site: Not Mandatory</td></tr> <tr> <td>iSalesOffice</td><td>- Sales Office: Not Mandatory</td></tr> <tr> <td>iForceRead</td><td>- Option to force new query in stead of using cached information</td></tr> <tr> <td>iContextIsMasterData</td><td>- True/False: Indicates if call is done to retrieve data for master data or transactional data.</td></tr> <tr> <td>...</td><td>- The field mnemonic of the required field.</td></tr> </table> Output: <table border="0"> <tr> <td>oExceptionMessage</td><td>- The last message if the return value is not equal to 0.</td></tr> </table>	iLogisticCompany	- Logistic Company: Mandatory	iSite	- Site: Not Mandatory	iSalesOffice	- Sales Office: Not Mandatory	iForceRead	- Option to force new query in stead of using cached information	iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.	...	- The field mnemonic of the required field.	oExceptionMessage	- The last message if the return value is not equal to 0.
iLogisticCompany	- Logistic Company: Mandatory														
iSite	- Site: Not Mandatory														
iSalesOffice	- Sales Office: Not Mandatory														
iForceRead	- Option to force new query in stead of using cached information														
iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.														
...	- The field mnemonic of the required field.														
oExceptionMessage	- The last message if the return value is not equal to 0.														

	If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... Return: 0 <> 0 - The value of the required field. - Data read - An error occurred

Sales.GetScheduleSettings

DLL:	tdextslsapi
Syntax:	<pre> long Sales.GetScheduleSettings(domain tcncmp iLogisticCompany, domain tccwoc iSalesOffice, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves settings for sales schedules. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tdsls000), or office or site level (settings table tdsls085) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when sales schedule series and step size for sales schedule lines are needed, the function must be called as follows: <pre> if Sales.GetScheduleSettings(* Fixed arguments: company, --> input sales office, --> input site, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output </pre>

	<pre> * Variable arguments: "sess", --> input series.for.sales.schedules, --> output "ssss", --> input step.size.for.sales.schedule.lines) <> 0 then --> output * Error, do something Exception.Delete(exception.id) endif Alternatively, the function can also be called with "sess.3" and "ssss.3", respectively, which refers to the mnemonics at parameter level. Pre: None Post: None Input: iLogisticCompany - Logistic Company: Mandatory iSite - Site: Not Mandatory iSalesOffice - Sales Office: Not Mandatory iForceRead - Option to force new query in stead of using cached information iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - An error occurred </pre>

Sales.StartProcessProFormaInvoices

DLL:	tdextslsapi
	This function is available from KB2290270 .
Syntax:	<pre> long Sales.StartProcessProFormaInvoices(long iStartMode, boolean iIgnoreSelectionFields, domain tcyesno iProcessOrders, domain tcyesno iProcessInstallments, domain tcyesno iProcessSchedules, domain tcorno iFromSalesOrder, domain tcorno iFromSalesSchedule, </pre>

	domain tccotp iFromSalesOrderType, domain tccwoc iFromSalesOffice, domain tcccom.bpid iFromSoldToBusinessPartner, domain tcccom.bpid iFromShipToBusinessPartner, domain tcccom.bpid iFromInvoiceToBusinessPartner, domain tcrefs iFromShipmentReference mb, domain tcorno iToSalesOrder, domain tcorno iToSalesSchedule, domain tccotp iToSalesOrderType, domain tccwoc iToSalesOffice, domain tcccom.bpid iToSoldToBusinessPartner, domain tcccom.bpid iToShipToBusinessPartner, domain tcccom.bpid iToInvoiceToBusinessPartner, domain tcrefs iToShipmentReference mb, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl This function starts the session Process Pro Forma Invoices (tdsls4248m000). Input: iStartMode Not Used. iIgnoreSelectionFields If true, From/To selection fields will not be filled in the session. iProcessOrders Process orders, must be Yes or No. iProcessInstallments Process installments, must be Yes or No. iProcessSchedules Process schedules, must be Yes or No. iFromSalesOrder From Sales Order selection field is filled with this value. (when iIgnoreSelectionFields is false) iFromSalesSchedule From Sales Schedule selection field is filled with this value. (when iIgnoreSelectionFields is false) iFromSalesOrderType From Sales Order Type selection field is filled with this value. (when iIgnoreSelectionFields is false) iFromSalesOffice From Sales Office selection field is filled with this value. (when iIgnoreSelectionFields is false) iFromSoldToBusinessPartner From Sold-to Business Partner selection field is filled with this value. (when iIgnoreSelectionFields is false) iFromShipToBusinessPartner From Ship-to Business Partner selection field is filled with this value. (when iIgnoreSelectionFields is false) iFromInvoiceToBusinessPartner From Invoice-to Business Partner selection field is filled with this value. (when iIgnoreSelectionFields is false) iFromShipmentReference From Shipment Reference selection field is filled with this value. (when iIgnoreSelectionFields is false) iToSalesOrder To Sales Order selection field is filled with this value. (when iIgnoreSelectionFields is false) iToSalesSchedule To Sales Schedule selection field is filled with this

	value. (when iIgnoreSelectionFields is false) iToSalesOrderType To Sales Order Type selection field is filled with this value. (when iIgnoreSelectionFields is false) iToSalesOffice To Sales Office selection field is filled with this value. (when iIgnoreSelectionFields is false) iToSoldToBusinessPartner To Sold-to Business Partner selection field is filled with this value. (when iIgnoreSelectionFields is false) iToShipToBusinessPartner To Ship-to Business Partner selection field is filled with this value. (when iIgnoreSelectionFields is false) iToInvoiceToBusinessPartner To Invoice-to Business Partner selection field is filled with this value. (when iIgnoreSelectionFields is false) iToShipmentReference To Shipment Reference selection field is filled with this value. (when iIgnoreSelectionFields is false) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

Public Interfaces for SalesOffice

The following functions are available:

[SalesOffices.StartOverview](#)

SalesOffices.StartOverview

DLL:	tdextslsapi
	This function is available from KB2285765 .
Syntax:	long SalesOffices.StartOverview(long iStartMode,

	<pre> domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccwoc iSalesOffice, ref domain tccwoc oSalesOffice, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Sales Offices in overview mode (tdsls0512m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used iSessionIndex Specifies the table-index that is to be used. Optional. Supported values: 1: sort by Sales Office (default) iQueryExtend A specific query to be used when zooming to this session. (Optional) iSalesOffice Sales Office (Optional) Output: for iStartMode MODAL: oSalesOffice The selected Sales Office oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

Public Interfaces for SalesQuote

The following functions are available:

[SalesQuote.Approve](#)

[SalesQuote.Process](#)

[SalesQuote.StartMultiMain](#)

[SalesQuotes.StartOverview](#)

SalesQuote.Approve

DLL:	tdextslsapi
	This function is available from KB3515640 .
Syntax:	<pre> long SalesQuote.Approve(domain tcqono iSalesQuote, domain tcgen.ynds iRecalculatePricesAndDiscounts, domain tcgen.ynds iRedetermineMaterialPriceInformation, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function can be used to approve the input sales quote. Function is using the same logic as the LN session 'Approve Sales Quotations'(tdslsl102m000). Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iSalesQuote - Sales Quote (mandatory) iRecalculatePricesAndDiscounts - Recalculate Price and Discounts before Sales Quote is approved (mandatory) No - Do not recalculate Yes - Recalculate Use Default Settings - Consider the parameter and/or sales office setting 'Recalculate Prices and Discounts' iRedetermineMaterialPriceInformation - Redetermine Material Price Information before Sales Quote is approved (mandatory) No - Do not redetermine Yes - Redetermine Use Default Settings - Consider the parameter and/or sales office setting 'Redetermine Material Information in Sales' Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Approval was successful <> 0 - An error occurred

SalesQuote.Process

DLL:	tdextslsapi
	This function is available from KB2300933 .
Syntax:	<pre> long SalesQuote.Process (domain tcqono iFromQuote, domain tccom.bpid iFromSoldToBusinessPartner, domain tccwoc iFromSalesOffice, domain tcqono iToQuote, domain tccom.bpid iToSoldToBusinessPartner, domain tccwoc iToSalesOffice, domain tcseri iSalesOrderSeries, domain tcpsts iInitialProjectStatus, domain tcyesno iConsolidateBySoldToBusinessPartner, domain tcyesno iCopyAfterSalesServiceLines, domain tcyesno iEquateProjectWithSalesOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function processes a range of sales quotes to sales orders. It is using the same logic and transaction handling as the LN session 'Process Sales Quotations'. (tdslsl200m000). Pre: LN Application sets retry-point, so not by caller Post: LN Application sets commit/abort transaction, so not by caller Input: - iFromQuote - From Quote selection field is filled with this value. - iFromSoldToBusinessPartner - From Sold-to Business Partner selection field is filled with this value. - iFromSalesOffice - From Sales Office selection field is filled with this value. - iToQuote - To Quote selection field is filled with this value. - iToSoldToBusinessPartner - To Sold-to Business Partner selection field is filled with this value. - iToSalesOffice - To Sales Office selection field is filled with this value. - iSalesOrderSeries - Sales Order Series, optional. - iInitialProjectStatus - Mandatory in case Project Control (PCS) is implemented. - iConsolidateBySoldToBusinessPartner - Consolidate by Sold-to BusinessPartner, must be Yes or No. - iCopyAfterSalesServiceLines - Copy After Sales Service Lines, must be Yes or No. - iEquateProjectWithSalesOrder - Equate Project with Sales Order, must be Yes or No. Output: - oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to the exception information. Use the functions in

	Return: 0 <> 0 Exception to get all relevant information. - Session started - Error.

SalesQuote.StartMultiMain

DLL:	tdextslsapi
Syntax:	<pre>long SalesQuote.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcqono iSalesQuote, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the Multi-Main session Sales Quotation (tdsls1600m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iQueryExtend A specific query to be used when zooming to this session. iSalesQuote Sales Quote Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 Session started An error occurred

SalesQuotes.StartOverview

DLL:	tdextslsapi
	This function is available from KB2301528 .

Syntax:	<pre> long SalesQuotes.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcqono iSalesQuote, domain tcmno iInternalSalesRepresentative, domain tccom.bpid iSoldToBusinessPartner, domain tcdate iExpiryDate, domain tccwoc iSalesOffice, ref domain tcqono oSalesQuote, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session Sales Quotations Overview (tdslsl1500m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used iSessionIndex Specifies the table-index that is to be used. Supported values: 1: sort by Sales Quotation (default) 2: sort by Internal Sales Representative, Sold-to Business Partner 3: sort by Sold-to Business Partner 4: sort by Expiry Date, Sold-to Business Partner 5: sort by Sales Office, Sold-to Business Partner iQueryExtend A specific query to be used when zooming to this session. iSalesQuote Sales Quotation (Optional) iInternalSalesRepresentative Internal Sales Representative (Optional) iSoldToBusinessPartner Sold-to Business Partner Mandatory if iStartMode = MODELESS and session-index 3 or 5 is used. iExpiryDate Expiry Date (Optional) iSalesOffice Sales Office Mandatory if iStartMode = MODELESS and session-index 5 is used. Output: for iStartMode MODAL: oSalesQuote The selected Sales Quotation oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

Public Interfaces for SalesQuoteLine

The following functions are available:

[SalesQuoteLine.RecalculatePriceAndDiscounts](#)

[SalesQuoteLine.StartDetail](#)

SalesQuoteLine.RecalculatePriceAndDiscounts

DLL:	tdextslsapi This function is available from KB3510295 .
Syntax:	<pre> long SalesQuoteLine.RecalculatePriceAndDiscounts(domain tcqono iSalesQuote, domain tcpono iSalesQuoteLine, domain tdsls.altn iSalesQuoteLineAlternative, long long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function will handle the recalculation of prices and discounts and the redetermination of material price information for the given sales quote line. Pre: Caller must set the retry-point Post: Caller must commit/abort the transaction. Input: - iSalesQuote - Sales Quote (Mandatory) - iSalesQuoteLine - Sales Quote Line (Mandatory) - iSalesQuoteLineAlternative - Sales Quote Line Alternative (Optional) - iProcessingOptionSet - Processing Option Set (Optional). If 0, the default recalculate options are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcextstapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete() Processing Options have a direct relationship with the form fields on session "Price and Discount Recalculation Parameters" (tdpcg0240s000) and are not explained in further detail here. Please refer to the session help for additional information. Recalculate options which are not available as Processing Options will get defaulted in accordance with the session logic.

Processing Options which are set while a required Implemented Software Component is not available are ignored.		
NAME	TYPE	DEFAULT
RecalculatePrice	domain tcyesno	tcyesno.yes
RecalculateProductVariantPrice	domain tcyesno	tcyesno.no
RecalculateEffectivityUnitPrice	domain tcyesno	tcyesno.no
RecalculateDiscount	domain tcyesno	tcyesno.yes
RedetermineMaterialPriceInformation	domain tcyesno	tcyesno.yes
RedetermineMaterialPriceAgreement	domain tcyesno	tcyesno.yes
OverwriteManualMaterialPriceAgreement	domain tcyesno	tcyesno.no
RedetermineMaterialLines	domain tcyesno	tcyesno.yes
OverwriteManualMaterialLines	domain tcyesno	tcyesno.no
OverwriteIgnoredMaterialLines	domain tcyesno	tcyesno.no
Output: oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.	
oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.	
Return: 0	- Recalculation was successful, or no recalculation needed to be done.	
<> 0	- An error occurred	

SalesQuoteLine.StartDetail

DLL:	tdextslsapi
Syntax:	<pre>long SalesQuoteLine.StartDetail(long iStartMode, domain tcqono iSalesQuote, domain tcpono iSalesQuoteLine, domain tdsls.altn iAlternative, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the detail session Sales Quotation Lines (tdsls1501m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iSalesQuote Sales Quote (Mandatory) iSalesQuoteLine Sales Quote Line (Mandatory) iAlternative Alternative(must be >= 0) Note that the given sales quote line must exist. Output: oExceptionMessage The last message if the return

	<pre> oExceptionID Return: 0 <> 0 </pre>	value is not equal to 0. If more than one message is given, these are present in the oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Session started An error occurred

Public Interfaces for SalesContract

The following functions are available:

[SalesContract.GetTotalAmounts](#)

[SalesContract.StartMultiMain](#)

[SalesContract.StartWorkbench](#)

[SalesContracts.StartOverview](#)

SalesContract.GetTotalAmounts

DLL:	tdextslsapi	
Syntax:	<pre> long SalesContract.GetTotalAmounts (domain tccono iSalesContract, ref domain tcccur oContractCurrency, ref domain tcamnt oContractOpenAmount, ref domain tcamnt oContractTotalAmount, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>	
Usage:	Expl.: Function calculates total contract net amounts in contract currency. Pre: N.A. Post: N.A. Input: iSalesContract - Sales Contract; Mandatory. Output: oContractCurrency - Contract Currency. oContractOpenAmount - Contract Open Amount: Sum of amounts of all non-terminated contract lines. oContractTotalAmount - Contract Total Amount: Sum of amounts of all contract lines. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is	

	given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Amount could be determined. - Error occurred.
	Return: 0 <> 0

SalesContract.StartMultiMain

DLL:	tdextslsapi
	This function is available from KB3643771 .
Syntax:	<pre>long SalesContract.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tccono iSalesContract, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the Multi-Main session Sales Contract (tdsls3600m000). Input: iStartMode Specifies the start mode for the session (mandatory). Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iQueryExtend A specific query to be used when zooming to this session (optional). iSalesContract Sales contract (mandatory). Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

SalesContract.StartWorkbench

DLL:	tdextslsapi This function is available from KB2301023 .
Syntax:	<pre> long SalesContract.StartWorkbench(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccwoc iSalesOffice, domain tcemno iInternalSalesRepresentative, domain tccom.bpid iSoldToBusinessPartner, domain tcitem iItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the workbench session Sales Contracts (tdsls8330m000). Input: iStartMode - Specifies the start mode for the session (Mandatory) - Possible values are: - MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. - MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. - iStartFilter - Not Used - iSessionIndex - Not Used - iQueryExtend - Not Used Following input variables form the filtering fields, these fields are not mandatory: - iSalesOffice - Sales Office (Optional) - iInternalSalesRepresentative - Internal Sales Representative (Optional) - iSoldToBusinessPartner - Buy-from Business Partner (Optional) - iItem - Item (Optional) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - An error occurred

SalesContracts.StartOverview

DLL:	tdextslsapi
	This function is available from KB2285765 .
Syntax:	<pre> long SalesContracts.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccono iSalesContract, domain tccom.bpid iSoldToBusinessPartner, domain tcmcs.st30m iCustomerOrder mb, domain tccwoc iSalesOffice, domain tcemno iInternalSalesRepresentative, ref domain tccono oSalesContract, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Sales Contracts in overview mode (tdsls3500m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used iSessionIndex Specifies the table-index that is to be used. Optional. Supported values: 1: sort by Sales Contract (default) 2: sort by Sold-to Business Partner, Customer Order 3: sort by Description, Contract 4: sort by Sales Office, Sold-to Business Partner, Contract 6: sort by Internal Sales Rep, Sold-to Business Partner iQueryExtend A specific query to be used when zooming to this session. (Optional) iSalesContract Sales Contract (Optional) iSoldToBusinessPartner Sold-to Business Partner (Optional) iCustomerOrder Customer Order (Optional) iSalesOffice Sales Office (Optional) iInternalSalesRepresentative Internal Sales Representative (Optional) Output: for iStartMode MODAL: oSalesContract The selected Sales Contract oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error

	Return: 0 <> 0	information. Use the functions in Exception to get all relevant information. Session started An error occurred

Public Interfaces for SalesContractLine

The following functions are available:

[SalesContractLines.StartOverview](#)

SalesContractLines.StartOverview

DLL:	tdextslsapi This function is available from KB2285765 .	
Syntax:	<pre> long SalesContractLines.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccono iSalesContract, domain tcpono iSalesContractLine, domain tccwoc iSalesOffice, domain tccprg iSalesPriceGroup, domain tcitem iItem, domain tccom.bpid iSoldToBusinessPartner, domain tccom.bpid iShipToBusinessPartner, domain tcaitm iCustomerItem mb, ref domain tccono oSalesContract, ref domain tcpono oSalesContractLine, ref domain tccwoc oSalesOffice, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>	
Usage:	Expl: This function starts the session Sales Contract Lines in overview mode (tdsls3501m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously.	

	<pre> iStartFilter Not Used iSessionIndex Specifies the table-index that is to be used. Optional. Supported values: 1: sort by Contract, Contract Line, Sales Office (default) 2: sort by Price Group, Item, Contract, Contract Line, Sales Office 4: sort by Sold-to BP, Ship-to BP, Price Group, Item, Contract, Contract Line, Sales Office 5: sort by Sold-to BP, Ship-to BP, Customer Item iQueryExtend A specific query to be used when zooming to this session. (Optional) iSalesContract Sales Contract Mandatory if iStartMode is MODELESS and session-index is 1 iSalesContractLine Sales Contract Line (Optional) iSalesOffice Sales Office (Optional) iSalesPriceGroup Sales Price Group (Optional) iItem Item Mandatory if iStartMode is MODELESS and session-index is 2 iSoldToBusinessPartner Sold-to Business Partner (Optional) iShipToBusinessPartner Ship-to Business Partner (Optional) iCustomerItem Customer Item (Optional) Output: for iStartMode MODAL: oSalesContract The selected Sales Contract oSalesContractLine The selected Sales Contract Line oSalesOffice The selected Sales Office oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred </pre>

Public Interfaces for SalesSchedule

The following functions are available:

[SalesSchedule.GetForecastQuantity](#)

[SalesSchedule.StartMultiMain](#)

SalesSchedule.GetForecastQuantity

DLL:	tdextslsapi
Syntax:	<pre> long SalesSchedule.GetForecastQuantity(domain tcomm.clus iCluster, domain tcitem iItem, domain tcdate iRequestedStartDate, domain tcdate iRequestedEndDate, ref domain tcqsl2 oForecastQuantity, ref domain tccuni oInventoryUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function determines the forecast quantity with regards to sales schedules for the given item in the given date range. Forecast quantity comes from the planned requirements of a Material Release with Release Capability and from the recalculated planned requirements of a Material Release (MR) followed by a (Sequence) Shipping Schedule ((S)SHP). For all schedules found for the input item, the forecast is determined. If the Item - Sold-to Business Partner information is not present for a certain sales schedule, the schedule is skipped. Date handling: Input arguments iRequestedStartDate and iRequestedEndDate are used to select the proper schedule lines. A sales schedule line falls within the requested date-range if the period between its start date (tdsls307.sdat) and end date (tdsls307.edat) is partly or entirely within the requested date range. <pre> [307] [307] [307] </pre> legend: [] -> start/end date of tdsls307 -> requested start/ end date Handling of overlapping periods: For a Material Release followed by a Shipping Schedule or a Sequence Shipping Schedule, correction must be done in case (Sequence) Shipping periods overlap with Material Release periods. For a Material Release Only no correction is needed. Planning Cluster handling: If iCluster is passed as empty, all schedule lines linked to nettable warehouses are selected independent of the cluster linked to the warehouse. If iCluster is filled, all schedules linked to nettable or non-nettable warehouses are selected but only if these warehouses are linked to iCluster. Pre: NA

	Post: NA Input: iCluster - Cluster (Optional) iItem - Item (Mandatory) iRequestedStartDate - Requested Start Date iRequestedEndDate - Requested End Date (Mandatory) must be after iRequestedStartDate Output: oForecastQuantity - Total approved forecast quantity. expressed in inventory unit oInventoryUnit - The inventory unit of the item. Return: 0 - Function executed successfully <> 0 - An error occurred

SalesSchedule.StartMultiMain

DLL:	tdextslsapi This function is available from KB2318770 .
Syntax:	<pre> long SalesSchedule.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iSalesSchedule, domain tdsls.reltype iSalesScheduleType, domain tcpono iSalesScheduleRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the Multi-Main session Sales Schedule (tdsls361lm000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend A specific query to be used when zooming to this session; Optional iSalesSchedule Sales Schedule; Optional iSalesScheduleType Sales Schedule Type; Optional iSalesScheduleRevision Sales Schedule Revision; Optional Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started

	<> 0	An error occurred

Public Interfaces for SalesScheduleLine

The following functions are available:

[SalesScheduleLine.Reprice](#)

SalesScheduleLine.Reprice

DLL:	tdextslsapi
Syntax:	<pre> long SalesScheduleLine.Reprice(domain tcorno iSalesSchedule, domain tdsls.rectype iSalesScheduleType, domain tcpono iSalesScheduleRevision, domain tcpono iSalesScheduleLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: The purpose of this function is to reprice the given schedule line. An error is returned if repricing is not allowed anymore for the given schedule line. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iSalesSchedule - Sales Schedule (Mandatory) iSalesScheduleType - Sales Schedule Type (Mandatory) iSalesScheduleRevision - Sales Schedule Revision iSalesScheduleLine - Sales Schedule Line (Mandatory) Output: NA Return: 0 - Schedule repriced <> 0 - An error occurred or repricing not allowed.

Public Interfaces for SalesOrder

The following functions are available:

[SalesOrder.Approve](#)

[SalesOrder.Block](#)[SalesOrder.CalculateAmounts](#)[SalesOrder.CopyBOM](#)[SalesOrder.CreditCheck](#)[SalesOrder.DetermineAdditionalCostLines](#)[SalesOrder.DetermineDefaultPurchaseOrderData](#)[SalesOrder.FinalizeBlocking](#)[SalesOrder.GenerateStructure](#)[SalesOrder.InitiateChangeRequest](#)[SalesOrder.RecalculatePriceAndDiscounts](#)[SalesOrder.ReleaseBlocking](#)[SalesOrder.Rush](#)[SalesOrder.StartIntakeWorkbench](#)[SalesOrder.StartMultiMain](#)[SalesOrder.StartPrintAcknowledgement](#)[SalesOrders.StartOverview](#)

SalesOrder.Approve

DLL:	tdextslsapi
Syntax:	<pre> long SalesOrder.Approve(domain tcorno iSalesOrder, domain tcgen.ynds iRecalculatePricesAndDiscounts, domain tcgen.ynds iRedetermineMaterialPriceInformation, domain tcgen.ynds iRecalculateAdditionalCosts, domain tcyesno iHandleCommissionsAndRebates, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl. : This Public Interface approves the given sales order. After the approval, the automatic order processing is executed. Note that promotions are not applied when Approving via this Public Interface, because promotions require user-interaction. This function contain its own transaction logic. As this Public Interface starts Automatic Order Processing (which could take a relatively long time) it is advised to not call this Public Interface in a BOD or BDE context. Doing that would suppress the internal transaction logic and would create one large database transaction, with potential locking problems as a consequence. Pre: Not Applicable

	Post: Not Applicable Input: iSalesOrder - Sales order (Mandatory) iRecalculatePricesAndDiscounts - Recalculate prices and discounts: - No - Do not recalculate - Yes - Recalculate - Use Default Settings - Consider the parameter and/or sales office setting 'Recalculate Prices and Discounts' iRedetermineMaterialPriceInformation - Redetermine Material Price Information - No - Do not redetermine - Yes - Redetermine - Use Default Settings - Consider the parameter and/or sales office setting 'Redetermine Material Information in Sales' iRecalculateAdditionalCosts - Recalculate additional costs. - No - Do not recalculate - Yes - See 'Use Default Settings' - Use Default Settings - Consider the parameter and/or sales office setting 'Recalculation of Additional Costs' iHandleCommissionsAndRebates - Allow linking of relations and Recalculation of commissions and rebates (Yes/No) Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Approval was successfull <> 0 - An error occurred

SalesOrder.Block

DLL:	tdextslsapi
	This function is available from KB2257724 .
Syntax:	<pre>long SalesOrder.Block(domain tcorno iSalesOrder, domain tcmcs.hrea iHoldReason, domain tctrns.date iExpectedReleaseDate, ref domain tcmcs.s999m oExceptionMessage mb,</pre>

	ref	long	oExceptionID)
Usage:	Expl.: This function can be used to block a sales order. It behaves similar to the functionality provided through session 'Sales Order (Line) Blocking' (tdsls4120s000).		
	Note: This function should not be used for blockings which are related to Trade Compliance.		
	Pre:	Caller must set retry-point	
	Post:	Caller must commit/abort transaction	
	Input:	iSalesOrder	- Sales Order (mandatory)
		iHoldReason	- Hold Reason (Mandatory)
			* Hold Reasons with a Trade Compliance related category are not allowed.
			* A Hold Reason that is used as 'Credit Limit Hold Reason' in Sales Order Parameters/Sales Office Settings is not allowed.
		iExpectedReleaseDate	- Expected Release Date. Optional; if the value zero (0) is passed, then the system will determine the expected release date based on the maximum holding days as defined for the business partner type and hold reason.
	Output:	oExceptionMessage	- The last message if the return value is not equal to 0.
			If more than one message is given, these are present in the oExceptionID
		oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
	Return:	0	- The block was executed successfully
		<> 0	- An error occurred

SalesOrder.CalculateAmounts

DLL:	tdextslsapi		
	This function is available from KB2300531 .		
Syntax:	<pre> long SalesOrder.CalculateAmounts(domain tcorno iSalesOrder, boolean iIncludeCostLines, boolean iCalculateInvoiceAmounts, boolean iCalculateCostOfSales, boolean iCalculateTaxAmount, ref domain tcamnt oGrossAmount, ref domain tcamnt oNetAmount, ref domain tcamnt oDiscountAmount, ref domain tcamnt oInvoiceRequiredAmount, ref domain tcamnt oInvoicedAmount, ref domain tccoam oCompanyOwnedCostOfSales, ref domain tccoam oCustomerOwnedCostOfSales, ref domain tcamnt oTaxAmount, ref domain tcmcs.s999m oExceptionMessage mb, </pre>		

	ref	long	oExceptionID)
Usage:	Expl:	This function calculates and returns several sales order amounts. The returned order amounts are expressed in the order currency.	
	Pre:	Not applicable	
	Post:	Not applicable	
	Input:	iSalesOrder	- The Sales Order. Mandatory.
		iIncludeCostLines	- True/False Indicates whether additional cost lines will be included in the calculation.
		iCalculateInvoiceAmounts	- True/False Indicates whether invoice amounts will be calculated and returned.
		iCalculateCostOfSales	- True/False Indicates whether cost of sales will be calculated and returned.
		iCalculateTaxAmount	- True/False Indicates whether tax amounts will be calculated and returned.
	Output:	oGrossAmount	- Gross amount
		oNetAmount	- Amount minus all line and header discounts (promotions included in calculation)
		oDiscountAmount	- Header discount amount (promotions included in calculation)
		oInvoiceRequiredAmount	- Amount released to Invoicing
		oInvoicedAmount	- Invoiced amount
		oCompanyOwnedCostOfSales	- Company owned cost of sales
		oCustomerOwnedCostOfSales	- Customer owned cost of sales
		oTaxAmount	- Tax amount
		oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.
		oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.
	Return:	0	- Amounts were successfull calculated and returned.
		DALHOOKERROR	- An error occurred

SalesOrder.CopyBOM

DLL:	tdextslsapi This function is available from KB2324143 .
Syntax:	<pre> long SalesOrder.CopyBOM(domain tcorno iSalesOrder, domain tcitem iMainItem, domain tcuef.effn iEffectivityUnit, domain tcqsl1 iOrderQuantityInInventoryUnit, domain tdnbol iMaximumNumberOfPhantomLevelsToSkip, domain tccprj iProject, domain tccspa iElement, domain tccact iActivity, domain tcyesno iIgnoreWarehouseFromBOM, domain tcyesno iCombineIdenticalComponents, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function copies the Bill of Material (BOM) to the given sales order. Note that the transaction management must be performed by the caller. If the function returns a non-zero value, then the entire transaction must be aborted by the caller. Pre: The Sales Order must exist Pre: Caller must set retry-point Post: Caller must commit/abort the transaction Input: - iSalesOrder - Sales Order (Mandatory) - iMainItem - Main Item (Mandatory) The item-code of which the BOM components are copied to the sales order lines. - iEffectivityUnit - Effectivity Unit (Optional) - iOrderQuantityInInventoryUnit - Order Quantity in inventory unit (Mandatory). - iMaximumNumberOfPhantomLevelsToSkip - The number of BOM-levels that are summarized when you use phantoms. Must be greater or equal to 1. - iProject - Project (Optional) - iElement - Element (Optional) - iActivity - Activity (Optional) - iIgnoreWarehouseFromBOM - Ignore the warehouse from the BOM. Yes: LN uses the standard warehouse defaulting logic No: LN uses the warehouse from the BOM (if present). Note that if Job Shop by Site is active, LN will always ignore the warehouse from the BOM. - iCombineIdenticalComponents - Combine Identical components. Yes: If a particular component occurs in the BOM multiple times, then these are aggregated into a single sales order line. No: Combining of identical components is not done.

	Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Sales order line(s) have been inserted <> 0 An error occurred. Caller must abort the transaction.

SalesOrder.CreditCheck

DLL:	tdextslsapi
	This function is available from KB3565482 .
Syntax:	<pre>long SalesOrder.CreditCheck(domain tcorno iSalesOrder, long iPhaseNumber, domain tcyesno iBlock, ref boolean oBlocked, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function checks if a sales order (or an open sales order change request) must be blocked based on the credit check logic. If so, the sales order will be blocked in case input argument "iBlock" is 'Yes'. Credit control is done based on the phase number, input argument "iPhaseNumber", in which it has to take place. Phase 1 represents the 'Order Entry' moment, phase 2 represents the 'Release to Warehousing' moment and phase 3 represents the 'Confirm Shipment' moment; independent of how far a sales order line really is in the process. Signalling and/or blocking depends on the setup. When concept "Change Request" is implemented and for the given sales order an open sales order change request exists, that change request should be used as input argument for "iSalesOrder". Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iSalesOrder - Sales Order (mandatory) iPhaseNumber - Phase Number to check 1 = 1st phase credit check (order entry) 2 = 2nd phase credit check (release to warehouse) 3 = 3rd phase credit check (confirm shipment) iBlock - Indicates if the sales order should be blocked or only a message should be given (based on sales parameters and/or settings). Output: oBlocked - True: Sales order is (already) blocked - False: Sales order is not blocked oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the

	oExceptionID Return: 0 <> 0	oExceptionID. - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. - No error - Error occurred

SalesOrder.DetermineAdditionalCostLines

DLL:	tdextslsapi	
	This function is available from KB2320802 .	
Syntax:	<pre>long SalesOrder.DetermineAdditionalCostLines(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderLineSequence, ref long oNumberOfCostLines, ref domain tcitem oItemArray() fixed, ref domain tcamnt oAmountArray(), ref domain tcdate oPlannedDeliveryDateArray(), ref domain tccprj oProjectArray() fixed, ref domain tccspa oElementArray() fixed, ref domain tccact oActivityArray() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: This function determines and returns the Additional Cost lines for the input Sales Order or Sales Order Line. In case the input argument iSalesOrderLine is empty the additional cost lines for the Sales Order header are returned, otherwise for the input Order Line. The Amount is expressed in the Currency of the input Order. Pre: N.A. Post: During the process the output arrays will be allocated, so afterwards a free.mem() must be done by the caller in order to free the memory which is occupied by the arrays. Input: iSalesOrder - Sales Order (Mandatory) iSalesOrderLine - Sales Order Line (Optional) iSalesOrderLineSequence - Sales Order Line Sequence (Optional) Output: oNumberOfCostLines - Number of Cost Lines that are present in the arrays oItemArray - Array with Cost Items oAmountArray - Array with Amounts, expressed in the Currency of the Sales Order oPlannedDeliveryDateArray - Array with Planned Delivery Dates oProjectArray - Array with Projects oElementArray - Array with Elements oActivityArray - Array with Activities oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant	

	Return: 0 <> 0	information. - Determination of Additional Cost was successful - An error occurred

SalesOrder.DetermineDefaultPurchaseOrderData

DLL:	tdextslsapi																															
	This function is available from KB3541875 .																															
Syntax:	<pre>long SalesOrder.DetermineDefaultPurchaseOrderData (domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderLineSequence, domain tccwoc iSalesOffice, domain tccwoc iSalesDepartment, domain tccom.bpid iSoldToBusinessPartner, domain tccom.bpid iShipToBusinessPartner, domain tcdate iSalesOrderDate, domain tcitem iItem, domain tdsls.dltp iDeliveryType, domain tcsite iSite, domain tcyesno iIsReturnOrder, ref domain tccom.bpid oBuyFromBusinessPartner, ref domain tccom.bpid oShipFromBusinessPartner, ref domain tccom.cadr oShipFromAddress, ref domain tccwoc oPurchaseOffice, ref domain tccwoc oFinancialDepartment, ref domain tcccur oCurrency, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>																															
Usage:	Expl: This function determines the default purchase order data for a given sales order. It retrieves relevant information such as supplier details based on the sales order requirements in the input fields. The more specific the input fields are filled, the better the correct defaults can be found. Pre: Not applicable Post: Not applicable Input: <table><tr><td>iSalesOrder</td><td>- Sales Order</td></tr><tr><td>iSalesOrderLine</td><td>- Sales Order Line</td></tr><tr><td>iSalesOrderLineSequence</td><td>- Sales Order Line Sequence (should be >= 0)</td></tr><tr><td>iSalesOffice</td><td>- Sales Office (mandatory)</td></tr><tr><td>iSalesDepartment</td><td>- Sales Department</td></tr><tr><td>iSoldToBusinessPartner</td><td>- Sold-to Business Partner</td></tr><tr><td>iShipToBusinessPartner</td><td>- Ship-to Business Partner</td></tr><tr><td>iSalesOrderDate</td><td>- Sales Order Date</td></tr><tr><td>iItem</td><td>- Item (mandatory)</td></tr><tr><td>iDeliveryType</td><td>- Delivery Type</td></tr><tr><td>iSite</td><td>- Site</td></tr><tr><td>iIsReturnOrder</td><td>- Flag if the Sales Order is a Return Order (yes/no)</td></tr></table> Output: <table><tr><td>oBuyFromBusinessPartner</td><td>- Buy-from Business Partner</td></tr><tr><td>oShipFromBusinessPartner</td><td>- Ship-from Business Partner</td></tr><tr><td>oShipFromAddress</td><td>- Ship-from Address</td></tr></table>		iSalesOrder	- Sales Order	iSalesOrderLine	- Sales Order Line	iSalesOrderLineSequence	- Sales Order Line Sequence (should be >= 0)	iSalesOffice	- Sales Office (mandatory)	iSalesDepartment	- Sales Department	iSoldToBusinessPartner	- Sold-to Business Partner	iShipToBusinessPartner	- Ship-to Business Partner	iSalesOrderDate	- Sales Order Date	iItem	- Item (mandatory)	iDeliveryType	- Delivery Type	iSite	- Site	iIsReturnOrder	- Flag if the Sales Order is a Return Order (yes/no)	oBuyFromBusinessPartner	- Buy-from Business Partner	oShipFromBusinessPartner	- Ship-from Business Partner	oShipFromAddress	- Ship-from Address
iSalesOrder	- Sales Order																															
iSalesOrderLine	- Sales Order Line																															
iSalesOrderLineSequence	- Sales Order Line Sequence (should be >= 0)																															
iSalesOffice	- Sales Office (mandatory)																															
iSalesDepartment	- Sales Department																															
iSoldToBusinessPartner	- Sold-to Business Partner																															
iShipToBusinessPartner	- Ship-to Business Partner																															
iSalesOrderDate	- Sales Order Date																															
iItem	- Item (mandatory)																															
iDeliveryType	- Delivery Type																															
iSite	- Site																															
iIsReturnOrder	- Flag if the Sales Order is a Return Order (yes/no)																															
oBuyFromBusinessPartner	- Buy-from Business Partner																															
oShipFromBusinessPartner	- Ship-from Business Partner																															
oShipFromAddress	- Ship-from Address																															

	oPurchaseOffice oFinancialDepartment oCurrency oExceptionMessage oExceptionID Return: 0 -> No error <> 0 -> Error occurred	- Purchase Office - Financial Department - Currency - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - An ID that refers to the exception information. Use the functions in Exception to get all relevant information.

SalesOrder.FinalizeBlocking

DLL:	tdextslsapi	
	This function is available from KB2285765 .	
Syntax:	<pre>long SalesOrder.FinalizeBlocking(domain tcorno iSalesOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: This function handles the creation of a Blocking record if required. This is required for instance if release to warehousing is aborted because the credit check failed. Pre: - Caller must set the retry-point. Note that if this function is called after a failing sales order line activity (like Release to Warehousing, Confirm Shipment), then that transaction must have been aborted and a new transaction started prior to calling this function. - A relevant function (like SalesOrderLine.ReleaseToWarehousing) must have been called for the same order number, prior to calling this function. Post: Caller must commit/abort transaction Input: iSalesOrder - Sales Order (mandatory) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Finalize was successful <> 0 - An error occurred	

SalesOrder.GenerateStructure

DLL:	tdextslsapi This function is available from KB3538863 .
Syntax:	<pre> long SalesOrder.GenerateStructure(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, long iProcessingOptionSet, ref boolean oStructureGenerated, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl. : This function generates the project (PCS) structure or product structure for the items of all lines of a sales order or for the item of a specific order line. The execution of automatic order steps is not started. Function <code>SalesOrderLine.StartAutomaticProcessing</code> can be used for this. Pre: Caller must set a retry-point by default. When the Processing Option "ConsiderMaterialPricing" is set to Yes (tctyesno.yes), the transaction management is done by the function itself, setting a retry-point is not needed. See also Processing Option "ConsiderMaterialPricing" below for more information. Post: Caller must commit/abort the transaction by default. When the Processing Option "ConsiderMaterialPricing" is set to Yes (tctyesno.yes), the transaction management is done by the function itself, commit or abort is not needed. See also Processing Option "ConsiderMaterialPricing" below for more information. Input: <code>iSalesOrder</code> - Sales order (Mandatory) <code>iSalesOrderLine</code> - Sales order line (Optional) If 0, a structure is generated for all order lines if applicable. If not 0, a structure is generated only for the specified order line. <code>iProcessingOptionSet</code> - Processing Option Set (Optional). If 0, the default values for the settings and option fields of session 'Generate (Project PCS) Structure' (tdsls4244m000) are applied. A Processing Option Set can be created via a call to <code>ProcessingOptionSet.Create()</code> in DLL <code>tcextxtapi</code>. After the call the option set can be deleted by calling <code>ProcessingOptionSet.Delete()</code> Processing Options have a direct relationship with the form fields available in the Selection and Options groups of session 'Generate (Project PCS) Structure'. Please refer to the session help for additional information on these fields. Session option "Print to predefined Device" is not available, because printing is not supported using this function. Any options which are not available as Processing Options will get defaulted in accordance with the logic of session 'Generate (Project PCS) Structure'.

Supported Processing Options and their defaults:		
NAME	TYPE	DEFAULT
ConsiderMaterialPricing	domain tcyesno	tcyesno.no
GenerationMethod	domain tccpge	tccpge.eto
ProjectSeries	domain tcseri	""
InitialProjectStatus	domain tcpsts	tcpsts.active
CreateProjectPerSalesOrder	domain tcyesno	tcyesno.yes
EquateProjectWithSalesOrder	domain tcyesno	tcyesno.yes
CheckStandardItemInventory	domain tcyesno	tcyesno.yes
IgnoreWarehouseFromBOM	domain tcyesno	tcyesno.no
CreateOrderLinesForPhantom	domain tcyesno	tcyesno.no
GenerateProjectPartsForCostServiceItems	domain tcyesno	tcyesno.no
GenerateProjectPartsForAllOrders	domain tcyesno	tcyesno.no

Some notes on options:

- ConsiderMaterialPricing
This option is not available on the session. Normally session 'Generate (Project PCS) Structure' updates material pricing content for applicable items when the concept "Material Pricing" is used (concept "Material Pricing" reads Yes in session 'Implemented Software Components', tccom0500m000/tccom0100s000). The logic that updates material pricing content has its own transaction management. This forces the logic of generating a project or product structure to also use its own transaction management.
Function SalesOrder.GenerateStructure uses the available logic of session 'Generate (Project PCS) Structure' and so, transaction management controlled by the caller is normally not possible. This restricts the use of the function.
Therefore option "ConsiderMaterialPricing" has been added. The option can have 2 values:

No (tcyesno.no, the default value if it is not specified)
 - the logic for Material Pricing is skipped
 - the caller must use its own transaction managementYes (tcyesno.yes)
 - the logic for Material Pricing is executed when applicable
 - the caller cannot use its own transaction management
- GenerationMethod
Valid values:
 - tccpge.eto (Engineer-to-Order)
 - tccpge.sto (Standard-to-Order)
- InitialProjectStatus
Valid values:
 - tcpsts.free (Free)
 - tcpsts.simulation (Simulated)
 - tcpsts.active (Active)
- CreateProjectPerSalesOrder
This option can be overruled by the setting of Sales Order Parameter "Link Installments to Projects". If this parameter reads Yes, the option is ignored and the system will generate one project per Sales Order.

Not all combinations of options are possible:

- CreateProjectPerSalesOrder
If this option is set to No, the value of option EquateProjectWithSalesOrder must be set to No.
The system will correct the value of EquateProjectWithSalesOrder if needed.

- `EquateProjectWithSalesOrder`
If this option is set to No, it is mandatory to provide a valid series with option `ProjectSeries`. If the `ProjectSeries` is not provided or it does not exist, the function cannot continue and returns an error.
- `CreateOrderLinesForPhantom`
If this option is set to No, the value of option `IgnoreWarehouseFromBOM` must be set to No. The system will correct the value of option `IgnoreWarehouseFromBOM` if needed.

Examples:

- 1) Caller uses default Processing Options. So Processing Option "ConsiderMaterialPricing" is No (the default).
Update of material pricing content is skipped, and the caller must handle the transaction management.

```

domain   tcorno          sales.order
domain   tcpono          sales.order.line
         boolean         structure.generated
         boolean         process.stopped
domain   tcmcs.str132    exception.message
         long            exception.id
         long            processing.option.set
         long            ret.val

sales.order = "SLS000101"
sales.order.line = 0
processing.option.set = 0

db.retry.point()

ret.val = SalesOrder.GenerateStructure(
    sales.order,
    sales.order.line,
    processing.option.set,
    structure.generated,    /* ref
    exception.message,      /* ref
    exception.id)           /* ref

if ret.val = 0 then
    commit.transaction()
else
    abort.transaction()
endif

```

- 2) Caller wants Material Pricing to be updated. Processing Option "ConsiderMaterialPricing" must be set to Yes, meaning transaction management is done by function `ProcessingOptionSet.Create`.

```

domain   tcorno          sales.order
domain   tcpono          sales.order.line
         boolean         structure.generated
         boolean         process.stopped
domain   tcmcs.str132    exception.message
         long            exception.id
         long            processing.option.set
         long            ret.val

```

	<pre> sales.order = "MPR000101" sales.order.line = 0 ret.val = ProcessingOptionSet.Create(processing.option.set, * ref exception.message, * ref exception.id, * ref * Options to be set: "ConsiderMaterialPricing", tcyesno.yes) if ret.val = 0 then ret.val = SalesOrder.GenerateStructure(sales.order, sales.order.line, processing.option.set, structure.generated, * ref exception.message, * ref exception.id) * ref endif if processing.option.set <> 0 then ret.val = ProcessingOptionSet.Delete(processing.option.set) * ref endif Output: oStructureGenerated - True: A structure has been generated - False: No structure has been generated oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - No error occurred <> 0 - An error occurred </pre>

SalesOrder.InitiateChangeRequest

DLL:	tdextslsapi
	This function is available from KB2290338 .
Syntax:	<pre> long SalesOrder.InitiateChangeRequest(domain tcorno iSalesOrder, domain tdsls.corg iChangeRequestOrigin, ref domain tcorno oChangeRequest, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function initiates a new change request for the given sales order. The initiation consists of two parts: 1. Determine if initiate change request is allowed.

	2. Initiate the change request. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iSalesOrder - Sales Order (mandatory) iChangeRequestOrigin - Origin of the Change Request (mandatory) Output: oChangeRequest - New Change Request oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 -> No error <> 0 -> Error occurred

SalesOrder.RecalculatePriceAndDiscounts

DLL:	tdextslsapi															
	This function is available from KB2329362 .															
Syntax:	<pre>long SalesOrder.RecalculatePriceAndDiscounts(domain tcorno iSalesOrder, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>															
Usage:	Expl : This function will handle the recalculation of prices and discounts and the redetermination of material price information for the given sales order. Pre : Caller must set the retry-point Post : Caller must commit/abort the transaction. Input : iSalesOrder - Sales Order (Mandatory) iProcessingOptionSet - Optional. If 0, the default recalculate options are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcextxtapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete() Processing Options have a direct relationship with the form fields on session "Price and Discount Recalculation Parameters" (tdpcg0240s000) and are not explained in further detail here. Please refer to the session help for additional information. Recalculate options which are not available as Processing Options will get defaulted in accordance with the session logic. Processing Options which are set while a required Implemented Software Component is not available are ignored. <table><tr><th>NAME</th><th>TYPE</th><th>DEFAULT</th></tr><tr><td>RecalculatePrice</td><td>domain tcyesno</td><td>tcyesno.yes</td></tr><tr><td>ApplyCumulativePrice</td><td>domain tcyesno</td><td>tcyesno.yes</td></tr><tr><td>OverwriteManualPrice</td><td>domain tcyesno</td><td>tcyesno.no</td></tr><tr><td>RecalculateDiscount</td><td>domain tcyesno</td><td>tcyesno.yes</td></tr></table>	NAME	TYPE	DEFAULT	RecalculatePrice	domain tcyesno	tcyesno.yes	ApplyCumulativePrice	domain tcyesno	tcyesno.yes	OverwriteManualPrice	domain tcyesno	tcyesno.no	RecalculateDiscount	domain tcyesno	tcyesno.yes
NAME	TYPE	DEFAULT														
RecalculatePrice	domain tcyesno	tcyesno.yes														
ApplyCumulativePrice	domain tcyesno	tcyesno.yes														
OverwriteManualPrice	domain tcyesno	tcyesno.no														
RecalculateDiscount	domain tcyesno	tcyesno.yes														

	<pre> ApplyTotalDiscount domain tcyesno tcyesno.yes OverwriteManualDiscount domain tcyesno tcyesno.no ApplyCumulativeDiscount domain tcyesno tcyesno.yes RedetermineMaterialPriceInformation domain tcyesno tcyesno.yes RedetermineMaterialPriceAgreement domain tcyesno tcyesno.yes OverwriteManualMaterialPriceAgreement domain tcyesno tcyesno.no RedetermineMaterialLines domain tcyesno tcyesno.yes OverwriteManualMaterialLines domain tcyesno tcyesno.no OverwriteIgnoredMaterialLines domain tcyesno tcyesno.no RedetermineMaterialActualPrices domain tcyesno tcyesno.yes OverwriteManualMaterialActualPrices domain tcyesno tcyesno.no </pre>
	<pre> Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Recalculation was successful, or <> 0 - no recalculation needed to be done. <> 0 - An error occurred </pre>

SalesOrder.ReleaseBlocking

DLL:	tdextslsapi
Syntax:	<pre> long SalesOrder.ReleaseBlocking(domain tcorno iSalesOrder, domain tcmcs.hrea iHoldReason, domain tdsls.rlty iReleaseType, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: Function can be used to release (delete) a sales order blocking. Note that this function will only release blockings that are linked on header level. Blockings that are done on line level or component level must be released separately, using function SalesOrderLine.ReleaseBlocking and SalesOrderLineComponent.ReleaseBlocking respectively. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iSalesOrder - Sales Order (mandatory) iHoldReason - Hold Reason (Mandatory) iReleaseType - Release Type (Soft / Firm) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Release successful <> 0 - Error during release occurred

SalesOrder.Rush

DLL:	tdextslsapi
	This function is available from KB3644594 .
Syntax:	<pre>long SalesOrder.Rush(domain tcorno iSalesOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl.: This function can be used to rush a sales order. It behaves similar to the functionality provided through the 'Rush' form command in session 'Sales Orders' (tdsls4100m000). Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iSalesOrder - Sales Order (mandatory) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - The rush was executed successfully <> 0 - An error occurred

SalesOrder.StartIntakeWorkbench

DLL:	tdextslsapi
	This function is available from KB3578854 .
Syntax:	<pre>long SalesOrder.StartIntakeWorkbench(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccwoc iSalesOffice, domain tcemno iInternalSalesRepresentative, domain tccom.bpid iSoldToBusinessPartner, domain tcitem iItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the session Sales Order Intake Workbench (tdsls4601m200). Input: iStartMode - Specifies the start mode for the session (Mandatory)

	Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter - Not Used iSessionIndex - Not Used iQueryExtend - Not Used Following input variables form the filtering fields, these fields are not mandatory: iSalesOffice - Sales Office (Optional) iInternalSalesRepresentative - Internal Sales Representative (Optional) iSoldToBusinessPartner - Sold-to Business Partner (Optional) iItem - Item (Optional) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - An error occurred

SalesOrder.StartMultiMain

DLL:	tdextslsapi
Syntax:	<pre> long SalesOrder.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcorno iSalesOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the Multi-Main session Sales Order (tdsls4100m900). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used.

	iQueryExtend A specific query to be used when zooming to this session. iSalesOrder Sales Order Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

SalesOrder.StartPrintAcknowledgement

DLL:	tdextslsapi
	This function is available from KB2285765 .
Syntax:	<pre>long SalesOrder.StartPrintAcknowledgement (long iStartMode, boolean iIgnoreSelectionFields, domain tccom.bpid iFromBusinessPartner, domain tcorno iFromSalesOrder, domain tccwoc iFromSalesOffice, domain tccom.bpid iToBusinessPartner, domain tcorno iToSalesOrder, domain tccwoc iToSalesOffice, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl This function starts the session Print Sales Order Acknowledgement/RMAs (tdsls4401m000). Depending on the main table of the calling session, Non-Consecutive Record Selection (NCRS) is used. When the main table is: Sales Orders (tdsls400) or Sales Order Lines (tdsls401) or Sales Order Activities (tdsls413) then the range selections will be imported based on the selected records in the parent session and the selection related fields will be ignored. Input: iStartMode Not Used. iIgnoreSelectionFields If true, From/To selection fields will not be filled in the session. iFromBusinessPartner From Business Partner selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iFromSalesOrder From Sales Order selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS

	is not applicable) iFromSalesOffice From Sales Office selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToBusinessPartner To Business Partner selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToSalesOrder To Sales Order selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToSalesOffice To Sales Office selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

SalesOrders.StartOverview

DLL:	tdextslsapi
	This function is available from KB2268796 .
Syntax:	<pre>long SalesOrders.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iSalesOrder, domain tccom.bpid iSoldToBp, domain tcemno iInternalSalesRepresentative, domain tccwoc iSalesOffice, ref domain tcorno oSalesOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the session Sales Orders Overview (tdsls4100m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel

	sessions that can be manipulated simultaneously. iStartFilter Not Used iSessionIndex Specifies the table-index that is to be used. Optional. Supported values: 1: sort by Sales Order (default) 2: sort by Sold-to Business Partner 3: sort by Internal Sales Rep 4: sort by Customer Order 5: sort by Sales Office, Sold-to Business Partner 6: sort by Shipment iQueryExtend A specific query to be used when zooming to this session. iSalesOrder Sales Order (Optional) iSoldToBp Sold-to Business Partner Mandatory if iStartMode = MODELESS and session-index 2 or 5 is used. iInternalSalesRepresentative Internal Sales Representative (Optional) iSalesOffice Sales Office Mandatory if iStartMode = MODELESS and session-index 5 is used. Output: for iStartMode MODAL: oSalesOrder The selected Sales Order oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

Public Interfaces for SalesOrderLine

The following functions are available:

[SalesOrderLine.Block](#)

[SalesOrderLine.Cancel](#)

[SalesOrderLine.ConfirmBackorderLine](#)

[SalesOrderLine.ConfirmSalesDelivery](#)

[SalesOrderLine.CopyFromOriginalInvoice](#)
[SalesOrderLine.CopyFromOriginalOrder](#)
[SalesOrderLine.CopyFromOriginalScheduleShipment](#)
[SalesOrderLine.CopyFromOriginalShipment](#)
[SalesOrderLine.CreateSalesDelivery](#)
[SalesOrderLine.ExecuteTradeComplianceCheck](#)
[SalesOrderLine.GenerateProductionOrder](#)
[SalesOrderLine.GetStatusDescription](#)
[SalesOrderLine.LinkContract](#)
[SalesOrderLine.LinkOrUnlinkInstallationGroup](#)
[SalesOrderLine.RecalculatePriceAndDiscounts](#)
[SalesOrderLine.ReleaseBlocking](#)
[SalesOrderLine.ReleaseManualActivity](#)
[SalesOrderLine.ReleaseToWarehousing](#)
[SalesOrderLine.Split](#)
[SalesOrderLine.StartAutomaticProcessing](#)
[SalesOrderLine.StartDetail](#)
[SalesOrderLines.ReleaseManualActivities.StartOverview](#)

SalesOrderLine.Block

DLL:	tdextslsapi
	This function is available from KB2257724 .
Syntax:	<pre> long SalesOrderLine.Block(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderSequence, domain tcmcs.hrea iHoldReason, domain tctrns.date iExpectedReleaseDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function can be used to block a sales order line. It behaves similar to the functionality provided through session 'Sales Order (Line) Blocking' (tdsls4120s000). Note: This function should not be used for blockings which are related to Trade Compliance. Pre: Caller must set retry-point

	Post: Caller must commit/abort transaction Input: iSalesOrder - Sales Order (mandatory) iSalesOrderLine - Sales Order Line (mandatory) iSalesOrderSequence - Sales Sequence number (must be >= 0) iHoldReason - Hold Reason (Mandatory) * Hold Reasons with a Trade Compliance related category are not allowed. * A Hold Reason that is used as 'Credit Limit Hold Reason' in Sales Order Parameters/Sales Office Settings is not allowed. iExpectedReleaseDate - Expected Release Date. Optional; if the value zero (0) is passed, then the system will determine the expected release date based on the maximum holding days as defined for the business partner type and hold reason. Output: oExceptionMessage - The last message if the return value is not equal to 0. oExceptionID - If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - The block was executed successfully <> 0 - An error occurred

SalesOrderLine.Cancel

DLL:	tdextslsapi
Syntax:	<pre> long SalesOrderLine.Cancel(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderSequence, domain tcmcs.str2 iSalesOrderLineAcknowledgement, domain tcmcs.str8 iSalesOrderLineChangeOrderSequence, domain tccdis iSalesOrderLineChangeReason, domain tccdis iSalesOrderLineChangeType, domain tcmcs.str2 iPurchaseOrderAcknowledgement, domain tcmcs.str8 iPurchaseOrderChangeOrderSequence, domain tccdis iPurchaseOrderChangeReason, domain tccdis iPurchaseOrderChangeType, domain tcmcs.str2 iPurchaseOrderLineAcknowledgement, domain tcmcs.str8 iPurchaseOrderLineChangeOrderSequence, domain tccdis iPurchaseOrderLineChangeReason, domain tccdis iPurchaseOrderLineChangeType, domain tcyesno iCancelCrossDockOrderLine, domain boolean iCheckCrossDockOrderLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function cancels a sales order line. The input sales order line can be an order line of type:

	- Total (sequence number must be 0) - Detail - Backorder If the given sales order line is of type 'Total', then all linked order lines are canceled as well. If the given sales order line is of type 'Detail' or 'Backorder', then the related Total-line will be be synchronized. If no detail-lines are open anymore, then the Total-line will be canceled as well. Depending on the situation, a purchase order line that is linked to the given sales order line may be canceled as well. If it's Direct Delivery, then the purchase order line will be canceled. If it's Cross Docking, then it depends on the argument 'iCancelCrossDockOrderLine'. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: - iSalesOrder - Sales Order (Mandatory) - iSalesOrderLine - Sales Order Line (Mandatory) - iSalesOrderSequence - Sales Sequence number (must be >= 0) iSalesOrderLineAcknowledgement - Acknowledgement code iSalesOrderLineChangeOrderSequence - Change Order Sequence Number iSalesOrderLineChangeReason - Change Reason Code (Mandatory if 'Change Codes Mandatory' is set to Yes in the Sales Order Parameters or Sales Order Settings by Office) iSalesOrderLineChangeType - Change Type (Mandatory if 'Change Codes Mandatory' is set to Yes in the Sales Order Parameters or Sales Order Settings by Office) iPurchaseOrderAcknowledgement - Acknowledgement code purchase order header iPurchaseOrderChangeOrderSequence - Change Order Sequence Number purchase order header iPurchaseOrderChangeReason - Change Reason purchase order header iPurchaseOrderChangeType - Change Type purchase order header iPurchaseOrderLineAcknowledgement - Acknowledgement code purchase order line iPurchaseOrderLineChangeOrderSequence - Change Order Sequence purchase order line iPurchaseOrderLineChangeReason - Change Reason purchase order line (Mandatory if 'Change Codes Mandatory' is set to Yes in the Purchase Order Parameters or Purchase Order Settings by Office) iPurchaseOrderLineChangeType - Change Type purchase order line (Mandatory if 'Change Codes Mandatory' is set to Yes in the Purchase Order Parameters or Purchase Order Settings by Office)

	<code>iCancelCrossDockOrderLine</code> - Yes: If a purchase order line is linked to the given order line using cross-docking, then it will be canceled as well. No: The linked cross-dock purchase order line (if any) is not canceled. <code>iCheckCrossDockOrderLine</code> - True: If a purchase order is linked to the given order line using cross-docking, then it is not allowed to cancel the given sales order line. The process is aborted in that case. False: The sales order line is allowed to be canceled if it is linked to a cross-dock purchase order line. Output: <code>oExceptionMessage</code> The last message if the return value is not equal to 0. If more than one message is given, these are present in the <code>oExceptionID</code> <code>oExceptionID</code> An ID that refers to all error information. Use the functions in <code>Exception</code> to get all relevant information. Return: 0 Cancellation was successful <> 0 An error occurred

SalesOrderLine.ConfirmBackorderLine

DLL:	tdextslsapi
Syntax:	<pre> long SalesOrderLine.ConfirmBackorderLine(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderSequence, domain tcsite iSite, boolean iManualConfirm, ref boolean oBackorderLineConfirmed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This functions handles the confirmation of the backorder line. When <code>iManualConfirm</code> is set to false the parameter <code>Confirm Back Orders Automatically</code> is checked if the backorder line should be processed automatically. If <code>Confirm Back Orders Automatically</code> is not set to automatically the process stops without error. In case <code>iManualConfirm</code> is set to true, the parameter <code>Confirm Back Orders Automatically</code> is not checked. Pre: Caller must set retry-point Post: Caller must commit/abort transaction

	Input: iSalesOrder - Sales Order (Mandatory) iSalesOrderLine - Sales Order Line (Mandatory) iSalesOrderSequence - Sales Sequence number (must be >= 0) iSite - Site of the Sales Order Line iManualConfirm - Backorder line is manually confirmed (True/False) or via a process. Output: oBackorderLineConfirmed - Backorder Line is confirmed oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Confirm succesfull or confirm not required. <> 0 Error during confirm occurred.

SalesOrderLine.ConfirmSalesDelivery

DLL:	tdextslsapi
Syntax:	<pre>long SalesOrderLine.ConfirmSalesDelivery(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderSequence, domain tcpono iSalesDeliverySequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This functions handles the confirmation of the Sales Delivery Line. iSalesDeliverySequence is the sequence from that comes from the Sales Delivery (tdsls426) record. This record must be present and allowed to be confirmed. Pre: Caller must set retry-point. Post: Caller must commit/abort transaction Input: iSalesOrder - Sales Order (Mandatory) iSalesOrderLine - Sales Order Line (Mandatory) iSalesOrderSequence - Sales Sequence number (must be >= 0) iSalesDeliverySequence - Sales Delivery Sequence number (must be >= 1) Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Confirm succesfull <> 0 Error during confirm occurred.

SalesOrderLine.CopyFromOriginalInvoice

DLL:	tdextslsapi
Syntax:	<pre> long SalesOrderLine.CopyFromOriginalInvoice(domain tcorno iSalesReturnOrder, domain tcncmp iInvoiceCompany, domain tccttyp iInvoiceTransactionType, domain tcinvn iOriginalInvoiceNumber, domain tcyesno iCopyAllLines, long iNrOfLinesToAdd, const domain tcorno iOriginalSalesOrderArray() fixed, const domain tcpono iOriginalLineArray(), const domain tcpono iOriginalSequenceArray(), const domain tcpono iOriginalActualDeliverySequenceArray(), const domain tcpono iOriginalInvoiceLineArray(), domain tcyesno iCopyFromHistory, ref long oNrOfLinesAdded, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function will copy the given sales order invoice lines to the sales return order (iSalesReturnOrder). Pre: The given return order must exist. Post: This function sets a retry-point and will commit and/or abort the transaction. Input: - iSalesReturnOrder - The Sales Return Order to which the order lines will be copied. Mandatory. - iInvoiceCompany - The company number of the invoice - iInvoiceTransactionType - The transaction type of the invoice - iOriginalInvoiceNumber - The invoice number from which part or all of the lines are to be returned. Mandatory. - iCopyAllLines - Yes: This will copy all the lines of the original invoice. No: Only a selection of lines will be copied. These lines are identified by the arguments below. - iNrOfLinesToAdd - The number of lines that are to be copied to the return order. This argument is mandatory if iCopyAllLines = No. - iOriginalSalesOrderArray - The sales order numbers from which the lines will be copied. The number of entries in this array must be equal to iNrOfLinesToAdd. This array is allowed to be empty if iCopyAllLines = Yes. - iOriginalLineArray - The line numbers of the original invoice lines that will be copied. The number of entries in this array must be equal to iNrOfLinesToAdd. This array is allowed to be empty if iCopyAllLines = Yes. - iOriginalSequenceArray - The sequence numbers of the original

	invoice lines that will be copied. The number of entries in this array must be equal to iNrOfLinesToAdd. This array is allowed to be empty if iCopyAllLines = Yes. iOriginalActualDeliverySequenceArray - The actual delivery sequence numbers of the original invoice lines that will be copied. The number of entries in this array must be equal to iNrOfLinesToAdd. This array is allowed to be empty if iCopyAllLines = Yes. iOriginalInvoiceLineArray - The invoice line numbers of the original invoice lines that will be copied. The number of entries in this array must be equal to iNrOfLinesToAdd. This array is allowed to be empty if iCopyAllLines = Yes. iCopyFromHistory - Not used Output: oNrOfLinesAdded - The number of lines added to the order. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 Copy was successfull An error occurred

SalesOrderLine.CopyFromOriginalOrder

DLL:	tdextslsapi
Syntax:	<pre> long SalesOrderLine.CopyFromOriginalOrder(domain tcorno iSalesReturnOrder, domain tcorno iOriginalSalesOrder, domain tcyesno iCopyAllLines, long iNrOfLinesToAdd, const domain tcpono iOriginalLineArray(), const domain tcpono iOriginalSequenceArray(), domain tcyesno iCopyFromHistory, ref long oNrOfLinesAdded, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function will copy the given sales order lines to the sales return order (iSalesReturnOrder). Pre: The given return order must exist. Post: This function sets a retry-point and will commit and/or abort the transaction. Input: iSalesReturnOrder - The Sales Return Order to which the order lines will be copied. Mandatory.

	<code>iOriginalSalesOrder</code> - The sales order from which the lines will be copied. Mandatory. <code>iCopyAllLines</code> - Yes: This will copy all the lines of the original sales order. (Note that Total lines and backorder lines are excluded from this.) No: Only a selection of lines will be copied. These lines are identified by the arguments below. <code>iNrOfLinesToAdd</code> - The number of lines that are to be copied to the return order. This argument is mandatory if <code>iCopyAllLines = No</code>. <code>iOriginalLineArray</code> - The line numbers of the original order lines that will be copied. The number of entries in this array must be equal to <code>iNrOfLinesToAdd</code>. This array is allowed to be empty if <code>iCopyAllLines = Yes</code>. <code>iOriginalSequenceArray</code> - The sequence numbers of the original order lines that will be copied. The number of entries in this array must be equal to <code>iNrOfLinesToAdd</code>. This array is allowed to be empty if <code>iCopyAllLines = Yes</code>. <code>iCopyFromHistory</code> - Yes: Copies the lines from the sales order line history table. No: Copies the lines from the actual sales order line table. Output: <code>oNrOfLinesAdded</code> - The number of lines added to the order. <code>oExceptionMessage</code> The last message if the return value is not equal to 0. If more than one message is given, these are present in the <code>oExceptionID</code> <code>oExceptionID</code> An ID that refers to all error information. Use the functions in <code>Exception</code> to get all relevant information. Return: 0 Copy was successfull <> 0 An error occurred

SalesOrderLine.CopyFromOriginalScheduleShipment

DLL:	tdextslsapi
	This function is available from KB2278226 .
Syntax:	<pre>long SalesOrderLine.CopyFromOriginalScheduleShipment(domain tcorno iSalesReturnOrder, domain tdshpm iOriginalShipment, domain tcyesno iCopyAllLines, long iNrOfLinesToAdd, const domain tcorno iOriginalScheduleArray() fixed, const domain tcpono iOriginalInvoiceLineArray(),</pre>

	ref	long	oNrOfLinesAdded,
	ref	domain tcmcs.s999m	oExceptionMessage mb,
	ref	long	oExceptionID)
Usage:	Expl: This function will copy the given sales schedule actual delivery lines (shipment lines) to the sales return order (iSalesReturnOrder). Pre: The given return order must exist. Post: LN Application sets retry-point, so not by caller Input: iSalesReturnOrder - The Sales Return Order to which the sales schedule actual lines delivery lines (shipment lines) will be copied. Mandatory. iOriginalShipment - The shipment from which part or all of the lines are to be returned. Mandatory. iCopyAllLines - Yes: This will copy all the lines of the original shipment. No: Only a selection of lines will be copied. These lines are identified by the arguments below. iNrOfLinesToAdd - The number of lines that are to be copied to the return order. This argument is mandatory if iCopyAllLines = No. iOriginalScheduleArray - The sales schedule numbers from which the lines will be copied. The number of entries in this array must be equal to iNrOfLinesToAdd. This array is allowed to be empty if iCopyAllLines = Yes. iOriginalInvoiceLineArray - The invoice line numbers of the original actual delivery lines that will be copied. The number of entries in this array must be equal to iNrOfLinesToAdd. This array is allowed to be empty if iCopyAllLines = Yes. Output: oNrOfLinesAdded - The number of lines added to the order. The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Copy was successful <> 0 An error occurred		

SalesOrderLine.CopyFromOriginalShipment

DLL:	tdextslsapi
Syntax:	long SalesOrderLine.CopyFromOriginalShipment (

	domain tcorno domain tdshpm domain tcyesno long const domain tcorno const domain tcpono const domain tcpono const domain tcpono const domain tcpono ref long ref domain tcmcs.s999m ref long	iSalesReturnOrder, iOriginalShipment, iCopyAllLines, iNrOfLinesToAdd, iOriginalSalesOrderArray() fixed, iOriginalLineArray(), iOriginalSequenceArray(), iOriginalActualDeliverySequenceArray(), iOriginalInvoiceLineArray(), oNrOfLinesAdded, oExceptionMessage mb, oExceptionID)
Usage:	Expl: This function will copy the given sales order actual delivery lines to the sales return order (iSalesReturnOrder). Note that this function does not support the returning of component lines. Pre: The given return order must exist. Post: This function sets a retry-point and will commit and/or abort the transaction. Input: - iSalesReturnOrder - The Sales Return Order to which the order lines will be copied. Mandatory. - iOriginalShipment - The shipment from which part or all of the lines are to be returned. Mandatory. - iCopyAllLines - Yes: This will copy all the lines of the original shipment. No: Only a selection of lines will be copied. These lines are identified by the arguments below. - iNrOfLinesToAdd - The number of lines that are to be copied to the return order. This argument is mandatory if iCopyAllLines = No. - iOriginalSalesOrderArray - The sales order numbers from which the lines will be copied. The number of entries in this array must be equal to iNrOfLinesToAdd. This array is allowed to be empty if iCopyAllLines = Yes. - iOriginalLineArray - The line numbers of the original actual delivery lines that will be copied. The number of entries in this array must be equal to iNrOfLinesToAdd. This array is allowed to be empty if iCopyAllLines = Yes. - iOriginalSequenceArray - The sequence numbers of the original actual delivery lines that will be copied. The number of entries in this array must be equal to iNrOfLinesToAdd. This array is allowed to be empty if iCopyAllLines = Yes. - iOriginalActualDeliverySequenceArray - The actual delivery sequence numbers of the original actual delivery lines that will be copied. The number of entries in this array must be equal to iNrOfLinesToAdd. This array is allowed to be empty if iCopyAllLines = Yes. - iOriginalInvoiceLineArray - The invoice line numbers of the original actual delivery lines that will be	

	Return: 0 <> 0 Exception to get all relevant information. Creation was successful Error during creation occurred.

SalesOrderLine.ExecuteTradeComplianceCheck

DLL:	tdextslsapi
	This function is available from KB2301023 .
Syntax:	<pre>long SalesOrderLine.ExecuteTradeComplianceCheck(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderLineSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Function can be used to execute the sales trade compliance check and log the related consumption for the given sales order line. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iSalesOrder - Sales Order (mandatory) iSalesOrderLine - Sales Order Line (mandatory) iSalesOrderSequence - Sales Sequence number Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - The trade compliance check was executed successfully <> 0 - An error occurred

SalesOrderLine.GenerateProductionOrder

DLL:	tdextslsapi
	This function is available from KB3569018 .
Syntax:	<pre>long SalesOrderLine.GenerateProductionOrder(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderLineSequence, long iProcessingOptionSet, ref domain tcorno oProductionOrder, ref domain tcmcs.s999m oExceptionMessage mb,</pre>

	ref	long	oExceptionID)
Usage:	Expl. :	This function generates a production order for the item of a specific sales order line.	
		The execution of automatic order steps is not started. Function SalesOrderLine.StartAutomaticProcessing can be used for this.	
	Pre:	Caller must set retry-point	
	Post:	Caller must commit/abort transaction	
	Input:	iSalesOrder	- Sales order (Mandatory)
		iSalesOrderLine	- Sales order line (Mandatory)
		iSalesOrderLineSequence	- Sales order line sequence
		iProcessingOptionSet	- Processing Option Set (Optional).
		If 0, the default options of session "Generate Production Orders" are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcextxtapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete()	
		Processing Options have a direct relationship with the form fields on session "Generate Production Orders" (tdsls4243m000) and are not explained in further detail here. Please refer to the session help for additional information. The session's print options are not applicable, because printing is not supported using this function to generate a production order for a single sales order line.	
	Any options which are not available as Processing Option will get defaulted in accordance with the session logic.		
	Processing Options which are set while a required Implemented Software Component is not available are ignored.		
	Supported Processing Options and their defaults:		
	NAME	TYPE	DEFAULT
	ProductionOrderSeries	domain tcseri	""
	ProcessDeliveryTypeProduction	domain tcyesno	tcyesno.yes
	ProcessDeliveryTypeWarehouse	domain tcyesno	tcyesno.yes
	ConsiderAllocatedToInventory	domain tcyesno	tcyesno.yes
	ConsiderDemandPeggedToSupplyOrders	domain tcyesno	tcyesno.yes
	Output:	oProductionOrder	- The generated production order
		oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID
		oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information
	Return:	0	- No error
		<> 0	- Error occurred

SalesOrderLine.GetStatusDescription

DLL:	tdextslsapi

	This function is available from KB2285765 .
Syntax:	<pre> long SalesOrderLine.GetStatusDescription(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderLineSequence, domain tcpono iSalesActualDeliverySequence, domain tcpono iInvoiceLine, ref domain tcdsca oDescription mb, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function retrieves the description of the status in Order Management for the given sales order line/sales order actual delivery sequence/invoice line. Note: The returned value in oDescription is intended for feedback to a user, not for decision making within the software itself. Pre: None Post: None Input: - iSalesOrder - Sales Order (Mandatory) - iSalesOrderLine - Sales Order Line (Mandatory) - iSalesOrderLineSequence - Sales Sequence number (must be >= 0) - iSalesActualDeliverySequence - Sales Actual Delivery Sequence number (Optional) - If iSalesActualDeliverySequence and iInvoiceLine are both 0, then the output will be determined on *order-line* level. - iInvoiceLine - Invoice Line (Optional) - If iSalesActualDeliverySequence and iInvoiceLine are both 0, then the output will be determined on *order-line* level. Output: - oDescription - The description of the status. - oExceptionMessage - The last message if the return value is not equal to 0. - If more than one message is given, these are present in the oExceptionID - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Function was executed successful <> 0 - An error occurred during execution of the function.

SalesOrderLine.LinkContract

DLL:	tdextslsapi
	This function is available from KB3566686 .
Syntax:	<pre> long SalesOrderLine.LinkContract(domain tcorno iSalesOrder, </pre>

	<table><tr><td></td><td>domain</td><td>tcpono</td><td>iSalesOrderLine,</td></tr><tr><td></td><td>domain</td><td>tcpono</td><td>iSalesOrderLineSequence,</td></tr><tr><td></td><td>domain</td><td>tcyesno</td><td>iSimulate,</td></tr><tr><td></td><td></td><td>long</td><td>iProcessingOptionSet,</td></tr><tr><td>ref</td><td>domain</td><td>tccono</td><td>oContract,</td></tr><tr><td>ref</td><td>domain</td><td>tcpono</td><td>oContractLine,</td></tr><tr><td>ref</td><td>domain</td><td>tccwoc</td><td>oContractSalesOffice,</td></tr><tr><td>ref</td><td>domain</td><td>tcmcs.s999m</td><td>oExceptionMessage mb,</td></tr><tr><td>ref</td><td></td><td>long</td><td>oExceptionID)</td></tr></table>		domain	tcpono	iSalesOrderLine,		domain	tcpono	iSalesOrderLineSequence,		domain	tcyesno	iSimulate,			long	iProcessingOptionSet,	ref	domain	tccono	oContract,	ref	domain	tcpono	oContractLine,	ref	domain	tccwoc	oContractSalesOffice,	ref	domain	tcmcs.s999m	oExceptionMessage mb,	ref		long	oExceptionID)
	domain	tcpono	iSalesOrderLine,																																		
	domain	tcpono	iSalesOrderLineSequence,																																		
	domain	tcyesno	iSimulate,																																		
		long	iProcessingOptionSet,																																		
ref	domain	tccono	oContract,																																		
ref	domain	tcpono	oContractLine,																																		
ref	domain	tccwoc	oContractSalesOffice,																																		
ref	domain	tcmcs.s999m	oExceptionMessage mb,																																		
ref		long	oExceptionID)																																		
Usage:	Expl.: This function tries to find an active contract that matches the order line data of the given sales order line. If so, the contract is linked to the sales order line. If there are multiple valid (special) contracts, the first contract found is always taken. When the "Change Request" concept is applied, a change request can be initiated for the supplied SalesOrder or an already open change request is used for contract linking. Contract linking can also be done by providing the change request as input argument for the "SalesOrder" field. If a change request is initiated by the Public Interface and no contract is found or the found contract is the same as on the sales order line, the initiated change request is canceled, because nothing is changed. Linking of contract to sales order line depends on the setup. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: <table><tr><td>iSalesOrder</td><td>- Sales order (mandatory)</td></tr><tr><td>iSalesOrderLine</td><td>- Sales order line (mandatory)</td></tr><tr><td>iSalesOrderLineSequence</td><td>- Sales order line sequence</td></tr><tr><td>iSimulate</td><td>- Indicates if contract linking should be simulated or really done if possible.</td></tr><tr><td>iProcessingOptionSet</td><td>- Processing Option Set (Optional). If 0, the default options are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcextxtapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete()</td></tr></table> Any options which are not available as Processing Option will get defaulted. Processing Options which are set while a required Implemented Software Component is not available are ignored. Supported Processing Options and their defaults: <table><tr><td>NAME</td><td>TYPE</td><td>DEFAULT</td></tr><tr><td>StillUseContractIfMaximumQuantityExceeds</td><td>domain tcyesno</td><td>tcyesno.yes</td></tr><tr><td colspan="3">Yes: If the quantity exceeds the maximum for a contract, the given input value, depending on the parameters of the sales contract, will cause the found contract to be accepted.</td></tr><tr><td colspan="3">No: If the quantity exceeds the maximum for a contract, the given input value, depending on the parameters of the sales contract, will cause the found contract to be skipped.</td></tr><tr><td>StillUseContractIfExpiryDateExceeds</td><td>domain tcyesno</td><td>tcyesno.yes</td></tr><tr><td colspan="3">Yes: If the expiry date exceeds for a contract, the given input value, depending on the parameters of the sales contract, will cause the found</td></tr></table>	iSalesOrder	- Sales order (mandatory)	iSalesOrderLine	- Sales order line (mandatory)	iSalesOrderLineSequence	- Sales order line sequence	iSimulate	- Indicates if contract linking should be simulated or really done if possible.	iProcessingOptionSet	- Processing Option Set (Optional). If 0, the default options are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcextxtapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete()	NAME	TYPE	DEFAULT	StillUseContractIfMaximumQuantityExceeds	domain tcyesno	tcyesno.yes	Yes: If the quantity exceeds the maximum for a contract, the given input value, depending on the parameters of the sales contract, will cause the found contract to be accepted.			No: If the quantity exceeds the maximum for a contract, the given input value, depending on the parameters of the sales contract, will cause the found contract to be skipped.			StillUseContractIfExpiryDateExceeds	domain tcyesno	tcyesno.yes	Yes: If the expiry date exceeds for a contract, the given input value, depending on the parameters of the sales contract, will cause the found										
iSalesOrder	- Sales order (mandatory)																																				
iSalesOrderLine	- Sales order line (mandatory)																																				
iSalesOrderLineSequence	- Sales order line sequence																																				
iSimulate	- Indicates if contract linking should be simulated or really done if possible.																																				
iProcessingOptionSet	- Processing Option Set (Optional). If 0, the default options are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcextxtapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete()																																				
NAME	TYPE	DEFAULT																																			
StillUseContractIfMaximumQuantityExceeds	domain tcyesno	tcyesno.yes																																			
Yes: If the quantity exceeds the maximum for a contract, the given input value, depending on the parameters of the sales contract, will cause the found contract to be accepted.																																					
No: If the quantity exceeds the maximum for a contract, the given input value, depending on the parameters of the sales contract, will cause the found contract to be skipped.																																					
StillUseContractIfExpiryDateExceeds	domain tcyesno	tcyesno.yes																																			
Yes: If the expiry date exceeds for a contract, the given input value, depending on the parameters of the sales contract, will cause the found																																					

	contract to be accepted. No: If the expiry date exceeds for a contract, the given input value, depending on the parameters of the sales contract, will cause the found contract to be skipped. ApproveAndProcessChangeRequestAutomatically domain tcyesno tcyesno.yes Yes: If Change Requests are applicable the created change request will be approved and processed automatically. No: Approval and processing of the change request (if any) is not done automatically. Output: oContract - Linked contract, if applicable oContractLine - Linked contract line, if applicable oContractSalesOffice - Linked contract office, if applicable oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - No error <> 0 - Error occurred

SalesOrderLine.LinkOrUnlinkInstallationGroup

DLL:	tdextslsapi
	This function is available from KB3622010 .
Syntax:	<pre> long SalesOrderLine.LinkOrUnlinkInstallationGroup(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderLineSequence, domain tcyesno iLinkInstallationGroup, domain tcclst iInstallationGroupToBeLinked, ref domain tcclst oUnlinkedInstallationGroup, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function links or unlinks an installation group to a sales order line and does the same as linking or unlinking an installation group to a sales order line in the session "Sales Order Lines" (tdsls4101m000). Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iSalesOrder - Sales order (mandatory) iSalesOrderLine - Sales order line (mandatory) iSalesOrderLineSequence - Sales order line sequence (should always be '0') iLinkInstallationGroup - Yes: If installation group should be linked to sales order line. No: If installation group should be unlinked from sales order line.

installation	<pre> iInstallationGroupToBeLinked - Installation group to be linked * if input argument "iLinkInstallationGroup" is 'Yes' then - if installation group to be linked is filled this one is used; - if installation group to be linked is empty an group from Service is determined if possible. * if input argument "iLinkInstallationGroup" is 'No' then installation group to be linked should be empty (not applicable). Output: oUnlinkedInstallationGroup - The unlinked installation group (if input argument "iLinkInstallationGroup" is 'No' and successfully unlinked) oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information Return: 0 - No error <> 0 - Error occurred </pre>

SalesOrderLine.RecalculatePriceAndDiscounts

DLL:	tdextslsapi
	This function is available from KB3500234 .
Syntax:	<pre> long SalesOrderLine.RecalculatePriceAndDiscounts(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderSequence, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function will handle the recalculation of prices and discounts and the redetermination of material price information for the given sales order line. Pre: Caller must set the retry-point Post: Caller must commit/abort the transaction. Input: - iSalesOrder - Sales Order (Mandatory) - iSalesOrderLine - Sales Order Line (Mandatory) - iSalesOrderSequence - Sales Sequence number (must be >= 0) - iProcessingOptionSet - Processing Option Set (Optional). If 0, the default recalculate options are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcextxtapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete() Processing Options have a direct relationship with the form fields

on session "Price and Discount Recalculation Parameters" (tdpcg0240s000) and are not explained in further detail here. Please refer to the session help for additional information.

Recalculate options which are not available as Processing Options will get defaulted in accordance with the session logic.

Processing Options which are set while a required Implemented Software Component is not available are ignored.

NAME	TYPE	DEFAULT
RecalculatePrice	domain tcyesno	tcyesno.yes
RecalculateProductVariantPrice	domain tcyesno	tcyesno.no
RecalculateEffectivityUnitPrice	domain tcyesno	tcyesno.no
RecalculateDiscount	domain tcyesno	tcyesno.yes
RedetermineMaterialPriceInformation	domain tcyesno	tcyesno.yes
RedetermineMaterialPriceAgreement	domain tcyesno	tcyesno.yes
OverwriteManualMaterialPriceAgreement	domain tcyesno	tcyesno.no
RedetermineMaterialLines	domain tcyesno	tcyesno.yes
OverwriteManualMaterialLines	domain tcyesno	tcyesno.no
OverwriteIgnoredMaterialLines	domain tcyesno	tcyesno.no
RedetermineMaterialActualPrices	domain tcyesno	tcyesno.yes
OverwriteManualMaterialActualPrices	domain tcyesno	tcyesno.no

Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID.

oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information.

Return: 0 - Recalculation was successful, or no recalculation needed to be done.

<> 0 - An error occurred

SalesOrderLine.ReleaseBlocking

DLL:	tdextslsapi
Syntax:	<pre> long SalesOrderLine.ReleaseBlocking(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderSequence, domain tcmcs.hrea iHoldReason, domain tdsls.rlty iReleaseType, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: Function can be used to release (delete) a sales order line blocking. Note: This function should NOT be used for blockings which are caused by Trade Compliance related checks. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iSalesOrder - Sales Order (mandatory) iSalesOrderLine - Sales Order Line (mandatory) iSalesOrderSequence - Sales Sequence number

	(must be >= 0) iHoldReason - Hold Reason (Mandatory) iReleaseType - Release Type (Soft / Firm) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Release successful <> 0 - Error during release occurred

SalesOrderLine.ReleaseManualActivity

DLL:	tdextslsapi
	This function is available from KB3638043 .
Syntax:	<pre>long SalesOrderLine.ReleaseManualActivity(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderLineSequence, ref domain tcmcs.st20m oActivity mb, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl.: This function releases a manual activity for a sales order line. Note: This function does not trigger any automatic processing afterwards, if applicable. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iSalesOrder - Sales order (mandatory) iSalesOrderLine - Sales order line (mandatory) iSalesOrderLineSequence - Sales order line sequence Output: oActivity - The released manual activity, if applicable oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - The manual activity is released <> 0 - An error occurred

SalesOrderLine.ReleaseToWarehousing

DLL:	tdextslsapi
	This function is available from KB2285765 .
Syntax:	<pre> long SalesOrderLine.ReleaseToWarehousing(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderLineSequence, ref domain tcyesno oReleased, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function handles the release of a sales order line to Warehousing. Pre: Caller must set retry-point Post: * Caller must commit/abort transaction * Function SalesOrder.FinalizeBlocking must be called * If desired, function SalesOrderLine.StartAutomaticProcessing can be called to start any automatic order steps for the given order (line). Input: iSalesOrder - Sales Order (Mandatory) iSalesOrderLine - Sales Order Line (Mandatory) iSalesOrderLineSequence - Sales Sequence number (must be >= 0) Output: oReleased Yes: The order line has been released to warehousing No: The order line has not been released to warehousing. Error or information messages may be present in oExceptionID. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Release successful <> 0 - Error during release occurred

SalesOrderLine.Split

DLL:	tdextslsapi
	This function is available from KB3567247 .
Syntax:	<pre> long SalesOrderLine.Split(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderLineSequence, domain tcqsl1 iQuantityForNewDetail, domain tcdate iPlannedDeliveryDateForNewDetail, domain tcsite iSiteForNewDetail, domain tccwar iWarehouseForNewDetail, </pre>

	ref ref long long domain long tcms.s999m long iProcessingOptionSet, oExceptionMessage mb, oExceptionID)															
Usage:	Expl.: This function performs similar functionality as splitting sales order lines/detail lines in the session "Sales Order Planned Delivery Lines" (tdsls4101m100). It handles the splitting of a sales order line (option "Generate Line" in session) and detail line (option "Split Line" in session). Note: 1. In case change requests are applicable and required, a CR is initiated to perform the split action. An option is provided to automatically approve/process the change request to the order (it performs a straightforward approve; no recalculations etc.). The change request will have origin 'Manual'. 2. This function does not trigger any automatic processing afterwards, if applicable. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iSalesOrder - Sales order (mandatory) iSalesOrderLine - Sales order line (mandatory) iSalesOrderLineSequence - Sales order line sequence iQuantityForNewDetail - The quantity that will be used for the new detail line (mandatory) iPlannedDeliveryDateForNewDetail- The planned delivery date that will be used for the new detail line (mandatory) iSiteForNewDetail - The site that will be used for the new detail line (mandatory if Sites is implemented) iWarehouseForNewDetail - The warehouse that will be used for the new detail line (mandatory) iProcessingOptionSet - Processing Option Set (Optional). If 0, the default options are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcextxtapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete() Any options which are not available as Processing Option will get defaulted. Processing Options which are set while a required Implemented Software Component is not available are ignored. Supported Processing Options and their defaults: <table><tr><th>NAME</th><th>TYPE</th><th>DEFAULT</th></tr><tr><td>DeleteFreightOrder</td><td>domain tcyesno</td><td>tcyesno.yes</td></tr><tr><td colspan="3"> Yes: If Freight Management is implemented an existing freight order linked to the sales order line will be deleted. No: If Freight Management is implemented an existing freight order linked to the sales order line will not be deleted; by that the sales order line will not be split. </td></tr><tr><td>ApproveAndProcessChangeRequestAutomatically</td><td>domain tcyesno</td><td>tcyesno.yes</td></tr><tr><td colspan="3"> Yes: If Change Requests are applicable, the created change request will be approved and processed automatically. No: Approval and processing of the change request (if any) is not done automatically. </td></tr></table>	NAME	TYPE	DEFAULT	DeleteFreightOrder	domain tcyesno	tcyesno.yes	Yes: If Freight Management is implemented an existing freight order linked to the sales order line will be deleted. No: If Freight Management is implemented an existing freight order linked to the sales order line will not be deleted; by that the sales order line will not be split.			ApproveAndProcessChangeRequestAutomatically	domain tcyesno	tcyesno.yes	Yes: If Change Requests are applicable, the created change request will be approved and processed automatically. No: Approval and processing of the change request (if any) is not done automatically.		
NAME	TYPE	DEFAULT														
DeleteFreightOrder	domain tcyesno	tcyesno.yes														
Yes: If Freight Management is implemented an existing freight order linked to the sales order line will be deleted. No: If Freight Management is implemented an existing freight order linked to the sales order line will not be deleted; by that the sales order line will not be split.																
ApproveAndProcessChangeRequestAutomatically	domain tcyesno	tcyesno.yes														
Yes: If Change Requests are applicable, the created change request will be approved and processed automatically. No: Approval and processing of the change request (if any) is not done automatically.																

	Output: oExceptionMessage oExceptionID Return: 0 <> 0	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. - The sales order line has been split. - An error occurred

SalesOrderLine.StartAutomaticProcessing

DLL:	tdextslsapi
Syntax:	<pre>long SalesOrderLine.StartAutomaticProcessing(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts processing the next sales order line activity that is set to execute automatically. This public interface will stop the automatic process when the next activity is not set to automatic. This function must not be called within a logical transaction as this function will have its own transaction handling. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iSalesOrder - Sales Order (mandatory) iSalesOrderLine - Sales Order Line (mandatory) iSalesOrderSequence - Sales Sequence number (must be >= 0) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Automatic Processing finished or next activity is not automatic. <> 0 - Not all mandatory input arguments are filled.

SalesOrderLine.StartDetail

DLL:	tdextslsapi
Syntax:	<pre> long SalesOrderLine.StartDetail(long iStartMode, domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderLineSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the detail session Sales Order Lines (tdsls4101m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iSalesOrder Sales Order (Mandatory) iSalesOrderLine Sales Order Line (Mandatory) iSalesOrderLineSequence Sales Order Line Sequence number (must be >= 0) Note that the given sales order line must exist. Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

SalesOrderLines.ReleaseManualActivities.StartOverview

DLL:	tdextslsapi
	This function is available from KB3557009 .
Syntax:	<pre> long SalesOrderLines.ReleaseManualActivities.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderLineSequence, </pre>

	<pre> domain tcmcs.st20m iActivity mb, ref domain tcorno oSalesOrder, ref domain tcpono oSalesOrderLine, ref domain tcpono oSalesOrderLineSequence, ref domain tcmcs.st20m oActivity mb, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session Release Manual Activities Sales Order Lines (tdsls4501m150). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used iSessionIndex Specifies the session index that is to be used. Supported values: 1: sort by "Order Lines by Activity" (default) 2: sort by "Execution Sequence of Activity by Order Line" 3: sort by "Activities by Order Line" iQueryExtend A specific query to be used when zooming to this session. iSalesOrder Sales Order Mandatory if iStartMode = MODELESS and session index 2 is used. iSalesOrderLine Sales Order Line iSalesOrderLineSequence Sales Order Line Sequence iActivity Activity Mandatory if iStartMode = MODELESS and session index 1 is used. Output: for iStartMode MODAL: oSalesOrder The selected Sales Order oSalesOrderLine The selected Sales Order Line oSalesOrderLineSequence The selected Sales Order Line Sequence oActivity The selected Activity oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

Public Interfaces for SalesOrderLineComponent

The following functions are available:

[SalesOrderLineComponent.Block](#)

[SalesOrderLineComponent.ReleaseBlocking](#)

SalesOrderLineComponent.Block

DLL:	tdextslsapi
	This function is available from KB2257724 .
Syntax:	<pre> long SalesOrderLineComponent.Block(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderSequence, domain tcpono iComponentSequence, domain tcmcs.hrea iHoldReason, domain tctrns.date iExpectedReleaseDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function can be used to block a sales order line component. It behaves similar to the functionality provided through session 'Sales Order (Line) Blocking' (tdsls4120s000). Note: This function should not be used for blockings which are related to Trade Compliance. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: - iSalesOrder - Sales Order (mandatory) - iSalesOrderLine - Sales Order Line (mandatory) - iSalesOrderSequence - Sales Sequence number (must be >= 0) - iComponentSequence - Component Sequence (Mandatory) - iHoldReason - Hold Reason (Mandatory) * Hold Reasons with a Trade Compliance related category are not allowed. * A Hold Reason that is used as 'Credit Limit Hold Reason' in Sales Order Parameters/Sales Office Settings is not allowed. - iExpectedReleaseDate - Expected Release Date. Optional; if the value zero (0) is passed, then the system will determine the expected release date based on the maximum holding days as defined for the business partner type and hold reason. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID

	oExceptionID Return: 0 <> 0	- An ID that refers to all error information. Use the functions in Exception to get all relevant information. - The block was executed successfully - An error occurred

SalesOrderLineComponent.ReleaseBlocking

DLL:	tdextslsapi	
Syntax:	<pre>long SalesOrderLineComponent.ReleaseBlocking(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderSequence, domain tcpono iComponentSequence, domain tcmcs.hrea iHoldReason, domain tdsls.rlty iReleaseType, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl.: Function can be used to release (delete) a sales order line component blocking. Note: This function should NOT be used for blockings which are caused by Trade Compliance related checks. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: - iSalesOrder - Sales Order (mandatory) - iSalesOrderLine - Sales Order Line (mandatory) - iSalesOrderSequence - Sales Sequence number - iComponentSequence - Component Sequence (Mandatory) - iHoldReason - Hold Reason (Mandatory) - iReleaseType - Release Type (Soft / Firm) Output: oExceptionMessage - The last message if the return value is not equal to 0. - If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Release successful <> 0 - Error during release occurred	

Public Interfaces for SalesOrderActualDeliveryLine

The following functions are available:

[SalesOrderActualDeliveryLine.ReadWarehousingStatusDescription](#)

SalesOrderActualDeliveryLine.ReadWarehousingStatusDescription

DLL:	tdextslsapi
	This function is available from KB2285765 .
Syntax:	<pre> long SalesOrderActualDeliveryLine.ReadWarehousingStatusDescription (domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderLineSequence, domain tcpono iSalesActualDeliverySequence, domain tcpono iInvoiceLine, ref domain tcdsca oDescription mb, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function retrieves the description of the status in Warehousing for the given sales order line/sales order actual delivery sequence. Note: The returned value in oDescription is intended for feedback to a user, not for decision making within the software itself. Pre: None Post: None calling this function. Input: iSalesOrder - Sales Order (Mandatory) iSalesOrderLine - Sales Order Line (Mandatory) iSalesOrderLineSequence - Sales Sequence number (must be >= 0) iSalesActualDeliverySequence - Sales Actual Delivery Sequence number (Optional) If iSalesActualDeliverySequence and iInvoiceLine are both 0, then the output will be determined on *order-line* level. iInvoiceLine - Invoice Line (Optional) If iSalesActualDeliverySequence and iInvoiceLine are both 0, then the output will be determined on *order-line* level. Output: oDescription - The description of the status in Warehousing oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information.

	Return: 0 <> 0	- Function was executed successful - An error occurred during execution of the function.

Public Interfaces for SalesOrderInvoiceLine

The following functions are available:

[SalesOrderInvoiceLine.ChangePriceAndDiscounts](#)

[SalesOrderInvoiceLine.ReadInvoicingStatusDescription](#)

[SalesOrderInvoiceLine.ReleaseToInvoicing](#)

SalesOrderInvoiceLine.ChangePriceAndDiscounts

DLL:	tdextslsapi
	This function is available from KB3518728 .
Syntax:	<pre> long SalesOrderInvoiceLine.ChangePriceAndDiscounts(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderLineSequence, domain tcpono iActualDeliveryLineSequence, domain tcpono iInvoiceLine, domain tcyesno iApplyPriceAndDiscountsToOtherInvoiceLines, domain tcyesno iApplyPriceStageToOtherInvoiceLinesForInstallments, domain tcyesno iUnlinkContractIfAlwaysUseContractPriceAndDiscount, domain tcyesno iUpdatePriceInItemSales, domain tcyesno iUpdatePriceInItemSalesByOffice, domain tcyesno iApproveAndProcessChangeRequestAutomatically, domain tcpric iPrice, domain tcprsg iPriceStage, const domain tcdisc iDiscountPercentage(), const domain tddiam iDiscountAmount(), const domain tddmth iDiscountMethod(), const domain tccdsc iDiscountCode() fixed, domain tcamnt iInvoiceAmount, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function can be used to change the Price and Discounts after Delivery has taken place. The change is done for the Sales Order Invoice Lines.

	If ION Workflow Document Approval is used and the Order is waiting for Approval in ION, updating is not allowed. Invoice lines are only updated if they have not yet been processed too far in Invoicing. Price changes must be possible (meaning that e.g. price is not read-only (yet)). Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iSalesOrder - Sales Order (mandatory) iSalesOrderLine - Sales Order Line (mandatory) iSalesOrderLineSequence - Sales Order Line Sequence Number iActualDeliveryLineSequence - Actual Delivery Line Sequence Number (mandatory) iInvoiceLine - Invoice Line iApplyPriceAndDiscountsToOtherInvoiceLines - Yes: Other Invoice Lines of the same Sales Order Line will be updated with Price and Discounts. No: Other Invoice Lines of the same Sales Order Line will not be updated. iApplyPriceStageToOtherInvoiceLinesForInstallments - Yes: Other Invoice Lines of the same Sales Order will be updated with Price Stage if 'Invoicing by Installments' applies to the Sales Order. No: No Invoice Lines of the Sales Order will be updated with the Price Stage. iUnlinkContractIfAlwaysUseContractPriceAndDiscount - Yes: When a Contract is linked and Price and/or Discounts are changed the Contract will be unlinked if 'Always Use Contract Price and Discount' applies to the Sales Order Invoice Line. No: Contract for Invoice Line is not unlinked if 'Always Use Contract Price and Discount' applies to the Sales Order Invoice Line. The Price and Discounts will not be changed. iUpdatePriceInItemSales - Yes: Price in Item Sales will be updated. No: Price is not updated iUpdatePriceInItemSalesByOffice - Yes: Price in Item Sales by Office will be updated. No: Price is not updated iApproveAndProcessChangeRequestAutomatically - Yes: If Change Requests are applicable the created change request will be approved and processed automatically. No: Approval and processing of the change request (if any) is not done automatically. iPrice - Price (In order currency) iPriceStage - Price Stage iDiscountPercentage - Discount Percentage; Array of 11 elements iDiscountAmount - Discount Amount; Array of 11 elements (In order currency) iDiscountMethod - Discount Method; Array of 11 elements iDiscountCode - Discount Code; Array of 11 elements iInvoiceAmount - Invoice Amount (In order currency) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the

	oExceptionID Return: 0 <> 0	oExceptionID. - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. - Price and Discounts are updated. - An error occurred

SalesOrderInvoiceLine.ReadInvoicingStatusDescription

DLL:	tdextslsapi This function is available from KB2285765.	
Syntax:	<pre>long SalesOrderInvoiceLine.ReadInvoicingStatusDescription(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderLineSequence, domain tcpono iSalesActualDeliverySequence, domain tcpono iInvoiceLine, ref domain tcdsca oDescription mb, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: This function retrieves the description of the status in Invoicing for the given sales order line/sales order invoice line. Note: The returned value in oDescription is intended for feedback to a user, not for decision making within the software itself. Pre: None Post: None Input: - iSalesOrder - Sales Order (Mandatory) - iSalesOrderLine - Sales Order Line (Mandatory) - iSalesOrderLineSequence - Sales Sequence number (must be >= 0) - iSalesActualDeliverySequence - Sales Actual Delivery Sequence number (Optional) - If iSalesActualDeliverySequence and iInvoiceLine are both 0, then the output will be based on all actual delivery lines/invoice lines that have been released to invoicing. - iInvoiceLine - Invoice Line (Optional) - If iSalesActualDeliverySequence and iInvoiceLine are both 0, then the output will be based on all actual delivery lines/invoice lines that have been released to invoicing. Output: - oDescription - The description of the status in Invoicing. - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - oExceptionID - An ID that refers to all error	

	information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - Function was executed successful - An error occurred during execution of the function.

SalesOrderInvoiceLine.ReleaseToInvoicing

DLL:	tdextslsapi
Syntax:	<pre>long SalesOrderInvoiceLine.ReleaseToInvoicing(domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderLineSequence, domain tcpono iInvoiceLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function handles the release of a sales order invoice line to Invoicing. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iSalesOrder - Sales Order (Mandatory) iSalesOrderLine - Sales Order Line (Mandatory) iSalesOrderLineSequence - Sales Sequence number (must be >= 0) iInvoiceLine - Invoice Line (must be >= 0) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - Release successful - Error during release occurred

Public Interfaces for SalesOrderChangeRequest

The following functions are available:

[SalesOrderChangeRequest.Approve](#)

[SalesOrderChangeRequest.Process](#)

SalesOrderChangeRequest.Approve

DLL:	tdextslsapi
	This function is available from KB2290338 .
Syntax:	<pre> long SalesOrderChangeRequest.Approve(domain tcorno iChangeRequest, domain tcgen.ynds iRecalculatePriceAndDiscounts, domain tcgen.ynds iRedetermineMaterialPrice, domain tcgen.ynds iRecalculateAdditionalCosts, ref boolean oChangeRequestApproved, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function approves a modified sales order change request. Pre: LN Application sets retry-point, so not by caller Post: LN Application sets commit/abort transaction, so not by caller Input: iChangeRequest - Sales Order Change Request (mandatory) iRecalculatePriceAndDiscounts - Recalculate Price and Discounts (mandatory) iRedetermineMaterialPrice - Redetermine Material Price (mandatory) iRecalculateAdditionalCosts - Recalculate Additional Costs (mandatory) Output: oChangeRequestApproved - Approved Sales Order Change Request (true/false) oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 -> No error <> 0 -> Error occurred

SalesOrderChangeRequest.Process

DLL:	tdextslsapi
	This function is available from KB2290338 .
Syntax:	<pre> long SalesOrderChangeRequest.Process(domain tcorno iChangeRequest, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function processes an approved sales order change request to the original sales order. Pre: Caller must set retry-point Post: Caller must set commit/abort transaction Input: iChangeRequest - Sales Order Change Request (mandatory) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error

	information. Use the functions in Exception to get all relevant information. Return: 0 -> No error <> 0 -> Error occurred

Public Interfaces for SalesOrderInstallment

The following functions are available:

[SalesOrderInstallments.Close](#)

[SalesOrderInstallments.StartMultiMain](#)

SalesOrderInstallments.Close

DLL:	tdextslsapi
	This function is available from KB3515640 .
Syntax:	<pre>long SalesOrderInstallments.Close(domain tcorno iSalesOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl.: This function closes sales order installments. Invoicing by installments must be 'Direct Settlement' or 'Indirect Settlement'. Correction installment lines will be added if required. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iSalesOrder - Sales Order (mandatory) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 -> No error <> 0 -> Error occurred

SalesOrderInstallments.StartMultiMain

DLL:	tdextslsapi
	This function is available from KB3634823 .
Syntax:	<pre> long SalesOrderInstallments.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcmcs.s999m iSalesOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the Multi-Main session Sales Order - Installments (tdsls4600m100). Input: iStartMode Specifies the start mode for the session (mandatory). Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used iQueryExtend A specific query to be used when zooming to this session (optional). iSalesOrder Sales order (mandatory). Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

Chapter 10 Public Interfaces for Commissions and Rebates

Public Interfaces for CommissionsAndRebates

The following functions are available:

[CommissionsAndRebates.CalculateCumulativeSales](#)

[CommissionsAndRebates.DetermineSalesOrderData](#)

[CommissionsAndRebates.EvaluateAgreementGrades](#)

[CommissionsAndRebates.InsertBasedOnAgreementGrades](#)

[CommissionsAndRebates.InsertCumulativeSales](#)

[CommissionsAndRebates.InsertCumulativeSalesV2](#)

CommissionsAndRebates.CalculateCumulativeSales

DLL:	tdextcmsapi
Syntax:	<pre> long CommissionsAndRebates.CalculateCumulativeSales(domain tdcms.type iCommissionRebateType, domain tdcms.prty iAgreementSearchPriority, domain tccom.bpid iAgreementSoldToBusinessPartner, domain tccom.bpid iAgreementRelation, domain tdcms.agrp iAgreementGroup, domain tccprj iAgreementProject, domain tcitem iAgreementItem, domain tdcms.cmgrp iAgreementCommissionRebateGroup, domain tcdate iAgreementEffectiveDate, domain tcccp.cpdtdt iAgreementPeriodTable, domain tcyrno iAgreementYear, domain tcpern iAgreementPeriod, domain tcorno iSalesOrderFrom, domain tcorno iSalesOrderTo, domain tcpono iSalesOrderLineFrom, domain tcpono iSalesOrderLineTo, domain tccwoc iSalesOfficeFrom, </pre>

	domain tccwoc iSalesOfficeTo, domain tcyesno iFullUpdate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl.: This function calculates commissions and rebates based on cumulative sales for all those agreements which have cumulative set to 'Yes'. When the commission and rebate has been calculated for a cumulative sales order line, the field 'Cumulative Processed (Y/N)' is set to 'Yes'. Pre: Retry point must be set Post: Transaction must be committed or aborted. Input: iCommissionRebateType - Type Commission or Rebate iAgreementSearchPriority - Agreement search priority as defined in Commissions and Rebates parameters iAgreementSoldToBusinessPartner - Agreement Sold-to Business Partner iAgreementRelation - Agreement Relation iAgreementGroup - Agreement Group iAgreementProject - Agreement Project iAgreementItem - Agreement Item iAgreementCommissionRebateGroup - Agreement Commission/Rebate Group iAgreementEffectiveDate - Agreement Effective Date iAgreementPeriodTable - Commission Period Table iAgreementYear - Agreement Year iAgreementPeriod - Agreement Period iSalesOrderFrom - Sales Order range From iSalesOrderTo - Sales Order range To iSalesOrderLineFrom - Sales Order Line range From iSalesOrderLineTo - Sales Order Line range To iSalesOfficeFrom - Sales Office range From iSalesOfficeTo - Sales Office range To iFullUpdate - Yes/No. Indicate if previous calculated Cumulative Commissions and Rebates need to be removed and calculated again Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Calculation of commission/rebate amount was successfull <> 0 An error occurred

CommissionsAndRebates.DetermineSalesOrderData

DLL:	tdextcmsapi
Syntax:	<pre> long CommissionsAndRebates.DetermineSalesOrderData(domain tdcms.type iCommissionRebateType, domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderSequence, domain tcpono iSalesOrderDeliverySequence, domain tcpono iSalesOrderInvoiceLine, domain tccom.bpid iRelation, domain tdcms.cmcmm iCalculationMethod, domain tdcms.cmac iCalculationMethodForDeliveries, ref domain tcitem oSalesOrderItem, ref domain tccuni oSalesOrderUnit, ref domain tcconv oSalesOrderUnitConversionFactor, ref domain tccprj oSalesOrderProject, ref domain tcdate oSalesOrderOrderDate, ref domain tcdate oSalesOrderRateDate, ref domain tcrtyp oSalesOrderRateType, ref domain tcratc oSalesOrderRate(), ref domain tcratf oSalesOrderRateFactor(), ref domain tcdate oSalesOrderDeliveryDate, ref domain tcttyp oSalesOrderTransactionType, ref domain tcinvn oSalesOrderInvoiceNumber, ref domain tcdate oSalesOrderInvoiceDate, ref domain tcncmp oSalesOrderInvoiceCompany, ref domain tcqivl oSalesOrderTotalQuantity, ref domain tcamnt oSalesOrderTotalNetAmount, ref domain tcqivl oSalesOrderAccumulatedOrderQuantity, ref domain tcamnt oSalesOrderAccumulatedOrderAmount, ref domain tdcms.cmpr oSalesOrderGrossProfit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: Function determines sales order (line) data based on history, including accumulated quantity and amount. Sales order gross profit is also calculated which is used while searching commission and rebate agreements. Depending on the Calculation Method (tdcms010.calm) the intake and cancellation records (tdsls451) are taken or the turnover records (tdsls456). Basically, total and accumulated quantity and amount are calculated as follows: Linking of Relations On: Sales Order Calculation Method: Sales Order - Total amount = sum of all sales order history lines (tdsls451) - Accumulated amount = sum of all sales order history lines (tdsls451) Calculation Method: Sales Invoice / Paid Sales Invoice - Total amount = sum of all sales order delivery history lines (tdsls456) - Accumulated amount = sum of all sales order delivery history lines (tdsls456) Linking of Relations On: Sales Order Line Calculation Method: Sales Order - Total amount = sum of all sales order history lines

	<pre> (tdsls451), intake and cancellation based on sales order/line/sequence Calculation method for deliveries: Separately - Accumulated amount = sum of all sales order history lines (tdsls451), intake and cancellation based on sales order/line/sequence Calculation method for deliveries: Accumulate, Ordered - Accumulated amount = sum of all sales order history lines (tdsls451), intake and cancellation based on sales order/line Calculation Method: Sales Invoice / Paid Sales Invoice - Total amount = sum of all sales order delivery history lines (tdsls456), based on sales order/ line/sequence/delivery sequence/invoice line Calculation method for deliveries: Separately - Accumulated amount = sum of all sales order delivery history lines (tdsls456), based on sales order/line/ sequence/delivery sequence/ invoice line Calculation method for deliveries: Accumulate/Fully Invoiced Line Accumulate/Partial Invoiced Line - Accumulated amount = sum of all sales order delivery history lines (tdsls456), based on sales order/line/ sequence </pre>
Pre:	None
Post:	None
Input:	<pre> iCommissionRebateType - Type Commission or Rebate ***** * Sales order (line) related arguments based on Sales Order * (Line) History (tdsls451/tdsls456) ***** iSalesOrder - Sales Order (Mandatory) iSalesOrderLine - Sales Order Line iSalesOrderSequence - Sales Sequence Number, it is 0 in case the Commission/Rebate Parameter 'Linking of Relations On' is set to Sales Order iSalesOrderDeliverySequence - Sales Order Delivery Sequence Line iSalesOrderInvoiceLine - Sales Order Invoice Line iRelation - Relation by sales order line iCalculationMethod - Calculation Method defined in Relations, used to control when the commissions and rebates are calculated iCalculationMethodForDeliveries - Calculation Method for Deliveries defined in Relations, used to control how the commissions and rebates are calculated after sales order deliveries are executed. </pre>
Output:	<pre> oSalesOrderItem - Sales order line item oSalesOrderUnit - Sales order unit oSalesOrderUnitConversionFactor </pre>

		- Conversion Factor Sales to Inventory Unit
	oSalesOrderProject	- Sales order line project
	oSalesOrderDate	- Sales order date
	oSalesOrderRateDate	- Sales order rate date
	oSalesOrderRateType	- Sales order rate type
	oSalesOrderRate	- Sales order rates
	oSalesOrderRateFactor	- Sales order rate factor
	oSalesOrderDeliveryDate	- Sales order delivery date
	oSalesOrderTransactionType	- Sales order transaction type
	oSalesOrderInvoiceNumber	- Sales order invoice number
	oSalesOrderInvoiceDate	- Sales order invoice date
	oSalesOrderInvoiceCompany	- Sales order invoice company
	oSalesOrderTotalQuantity	- Sum of all quantities in sales order unit.
	oSalesOrderTotalNetAmount	- Sum of all amounts in sales order currency
	oSalesOrderAccumulatedOrderQuantity	- Accumulated quantity in sales order unit. See explanation above.
	oSalesOrderAccumulatedOrderAmount	- Accumulated net amount in sales order currency. See explanation above.
	oSalesOrderGrossProfit	- Sales order gross profit
	oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
	oExceptionID	An ID that refers to all error information. Use the functions in Exception to get all relevant information.
	Return: 0	Sales order history records exist and quantities/amounts could be calculated
	<> 0	Sales order history records do not exist or quantity/amounts could not be calculated.

CommissionsAndRebates.EvaluateAgreementGrades

DLL:	tdextcmsapi
Syntax:	<pre>long CommissionsAndRebates.EvaluateAgreementGrades(domain tdcms.type iCommissionRebateType, domain tdcms.prtty iAgreementSearchPriority, domain tcitem iAgreementItem, domain tckogr iAgreementGradebyQuantityAmount, domain tcccur iAgreementCurrency, domain tccuni iAgreementGradeUnit, domain tcconv iGradeUnitConversionFactor, domain tcgnpr iAgreementGradeCalculation, domain tdcms.fagu iAgreementFixedAmountOn, domain tdcms.grtl iAgreementCommissionRebateOver, domain tdcms.grtl iAgreementGrowCommissionRebateOver,</pre>

		domain	tdcms.pyrs	iAgreementInvoiceDirectReserve,
		domain	tccdsc	iAgreementDiscountCode,
		domain	tcqanp	iAgreementGradeUpToQuantityAmount,
		domain	tdcms.cmpr	iAgreementGrossProfitPercentage,
		domain	tdcms.cmpr	iAgreementCommissionRebatePercentage,
		domain	tdcms.cmpr	iAgreementGrowPercentage,
		domain	tcamnt	iAgreementFixedAmount,
		domain	tcamnt	iAgreementMaximumLimit,
		domain	tcorno	iSalesOrder,
		domain	tcpono	iSalesOrderLine,
		domain	tcpono	iSalesOrderSequence,
		domain	tcpono	iSalesOrderDeliverySequence,
		domain	tcpono	iSalesOrderInvoiceLine,
		domain	tcncmp	iSalesOrderInvoiceCompany,
		domain	tcctyp	iSalesOrderTransactionType,
		domain	tcinvn	iSalesOrderInvoiceNumber,
		domain	tcdate	iSalesOrderInvoiceDate,
		domain	tccwoc	iSalesOrderOffice,
		domain	tccwoc	iSalesOrderFinancialDepartment,
		domain	tcamnt	iSalesOrderAmount,
		domain	tcccur	iSalesOrderCurrency,
		domain	tcqivl	iSalesOrderQuantity,
		domain	tccuni	iSalesOrderUnit,
		domain	tcconv	iSalesOrderUnitConversionFactor,
		domain	tcdate	iSalesOrderRateDate,
		domain	tcrtyp	iSalesOrderRateType,
	ref	domain	tcratc	iSalesOrderRate(),
	ref	domain	tcratf	iSalesOrderRateFactor(),
		domain	tdcms.cmpr	iSalesOrderGrossProfitPercentage,
		domain	tcqivl	iSalesOrderAccumulatedOrderQuantity,
		domain	tcamnt	iSalesOrderAccumulatedOrderAmount,
		domain	tccom.bpid	iRelation,
		domain	tdcms.rtyp	iRelationType,
		domain	tcpono	iRelationSequence,
		domain	tdcms.cupf	iRelationCurrencyForInvoicing,
		domain	tccom.bpid	iRelationBuyFromBusinessPartner,
		domain	tccom.bpid	iRelationSoldToBusinessPartner,
	ref	domain	tcccur	oCommissionRebateInvoiceCurrency,
	ref	domain	tdcms.cmpr	oCommissionRebatePercentage,
	ref	domain	tdcms.cmpr	oCommissionRebateGrowPercentage,
	ref	domain	tcamnt	oCommissionRebateGrowAmount,
	ref	domain	tcamnt	oCommissionRebateFixedAmountInOrderCurrency,
	ref	domain	tcamnt	oCommissionRebateSalesOrderAmountInOrderCurrency,
	ref	domain	tcamnt	oCommissionRebateAmountInOrderCurrency,
	ref		boolean	oSkipCommissionRebateAgreement,
	ref		boolean	oStopSearchingAgreement,
	ref		boolean	ioInitializeStatics,
	ref	domain	tcmcs.s999m	oExceptionMessage mb,
	ref		long	oExceptionID)
Usage:	Expl.: Function searches and applies agreements based on the search priority which is used to calculate the commissions and rebates. Pre: None Post: None Input: iCommissionRebateType - Type Commission or Rebate iAgreementSearchPriority - Agreement search priority as defined in Commissions and Rebates parameters ***** * Input arguments as defined in Commissions/Rebates Agreements ***** iAgreementItem - Agreement Item iAgreementGradeByQuantityAmount			

		- Agreement Grade by Quantity/ Amount
iAgreementCurrency		- Agreement Currency
iAgreementGradeUnit		- Agreement Grade Unit
iGradeUnitConversionFactor		- Conversion factor between inventory unit of item and Commission/Rebate Agreement Grade Unit Only applicable in case Agreement Grade by Quantity/Amount is set to Quantity
iAgreementGradeCalculation		- Agreement Grade Calculation Gross/Net
iAgreementFixedAmountOn		- Agreement Fixed Amount on Grade/Total
iAgreementCommissionRebateOver		- Agreement Commission/Rebate Over Grade/Total
iAgreementGrowCommissionRebateOver		- Agreement Grow Commission/ Rebate Over Grade/Total
iAgreementInvoiceDirectReserve		- Agreement Invoice/Reserve
iAgreementDiscountCode		- Agreement Discount Code
iAgreementGradeUpToQuantityAmount		- Agreement Grade Upto Quantity/ Amount. In case Agreement Grade by Quantity/Amount is set to 'Amount' then Agreement Grade is Upto Amount otherwise it is Upto Quantity
iAgreementGrossProfitPercentage		- Agreement Grade Gross Profit Percentage
iAgreementCommissionRebatePercentage		- Agreement Grade Commission/Rebate Percentage
iAgreementGrowPercentage		- Agreement Grade Growing Percentage
iAgreementFixedAmount		- Agreement Grade Fixed Amount
iAgreementMaximumLimit		- Agreement Grade Maximum Limit

* Sales order (line) related arguments based on Sales Order		
* (Line) History (tdsls451/tdsls456)		

iSalesOrder	- Sales Order	
iSalesOrderLine	- Sales Order Line	
iSalesOrderSequence	- Sales Sequence Number, it is 0 in case the Commission/Rebate Parameter 'Linking of Relations On' is set to Sales Order	
iSalesOrderDeliverySequence	- Sales Order Delivery Sequence Line	
iSalesOrderInvoiceLine	- Sales Order Invoice Line	
iSalesOrderInvoiceCompany	- Sales Order Line Invoice Company	
iSalesOrderTransactionType	- Sales Order Line Transaction Type	
iSalesOrderInvoiceNumber	- Sales Order Line Invoice Number	
iSalesOrderInvoiceDate	- Sales Order Line Invoice Date	

iSalesOrderOffice	- Sales Order Line Sales Office
iSalesOrderFinancialDepartment	- Sales Order Line Financial Department
iSalesOrderAmount	- Sales Order Line amount in order currency
iSalesOrderCurrency	- Sales Order Currency
iSalesOrderQuantity	- Sales Order Line quantity in sales Order Line unit
iSalesOrderUnit	- Sales Order Line Unit
iSalesOrderUnitConversionFactor	- Conversion Factor Sales to Inventory Unit
iSalesOrderRateDate	- Sales Order Line Rate Date
iSalesOrderRateType	- Sales Order Line Rate Type
iSalesOrderRate	- Sales Order Line Rate
iSalesOrderRateFactor	- Sales Order Line Rate Factor
iSalesOrderGrossProfitPercentage	- Sales Order Line Gross Profit percentage
iSalesOrderAccumulatedOrderQuantity	- Accumulated Sales Order (lines) quantity. Sum of all quantities in order unit.
iSalesOrderAccumulatedOrderAmount	- Accumulated Sales Order (lines) amount in order currency
iRelation	- Relation by sales order line
iRelationType	- Relation Type: Employee/Supplier/Customer
iRelationSequence	- Relation sequence by order line
iRelationCurrencyForInvoicing	- Relation Currency for Invoicing Relation/Company/N.A./Sales Order Agreement
iRelationBuyFromBusinessPartner	- Relation Buy-from BP, only applicable in case the Relation type is 'Supplier'
iRelationSoldToBusinessPartner	- Relation Buy-from BP, only applicable in case the relation type is set to 'Customer'
Output: oCommissionRebateInvoiceCurrency	- Commissions/Rebates Invoice Currency based on invoice currency for relation
oCommissionRebatePercentage	- Commissions/Rebates according Commissions/Rebates Agreement Grades
oCommissionRebateGrowPercentage	- Commissions/Rebates Grow Percentage according Commissions/Rebates Agreement Grades
oCommissionRebateGrowAmount	- Commissions/Rebates Grow Amount based on Commissions/Rebates Agreement Grades Grow Percentage
oCommissionRebateFixedAmountInOrderCurrency	- Commissions/Rebates Fixed Amount according Commissions/Rebates Agreement Grades
oCommissionRebateSalesOrderAmountInOrderCurrency	- Sales Order (Line) Amount in sales

	order line currency oCommissionRebateAmountInOrderCurrency - Commissions/Rebates Amount in order currency based on Commissions/Rebates Agreement Grades Commission/Rebate Percentage oSkipCommissionRebateAgreement - True/False. Indicates that the Commission/Rebate agreement grade is not applicable to sales order (line) oStopSearchIngAgreement - True/False. Indicates that all agreement grade quantity/amounts have been fit ioInitializeStatics - True/False. Indicates that variables used for determining agreement grades need to be initialized. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Evaluation of commission/rebate amount was successfull <> 0 An error occurred

CommissionsAndRebates.InsertBasedOnAgreementGrades

DLL:	tdextcmsapi
Syntax:	<pre> long CommissionsAndRebates.InsertBasedOnAgreementGrades(domain tdcms.type iCommissionRebateType, domain tdcms.prtty iAgreementSearchPriority, domain tdcms.pyrs iAgreementInvoiceDirectReserve, domain tccdsc iAgreementDiscountCode, domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderSequence, domain tcpono iSalesOrderDeliverySequence, domain tcpono iSalesOrderInvoiceLine, domain tccttyp iSalesOrderTransactionType, domain tcinvn iSalesOrderInvoiceNumber, domain tcdate iSalesOrderInvoiceDate, domain tccwoc iSalesOrderOffice, domain tccwoc iSalesOrderFinancialDepartment, domain tcccur iSalesOrderCurrency, domain tcdate iSalesOrderRateDate, domain tcrtyp iSalesOrderRateType, ref domain tcratc iSalesOrderRate(), ref domain tcratf iSalesOrderRateFactor(), </pre>

	domain tccom.bpid iRelation, domain tdcms.rtyp iRelationType, domain tcpono iRelationSequence, domain tdcms.cmpr iCommissionRebatePercentage, domain tdcms.cmpr iCommissionRebateGrowPercentage, domain tcamnt iCommissionRebateGrowAmount, domain tcamnt iCommissionRebateFixedAmountInOrderCurrency, domain tcamnt iCommissionRebateSalesOrderAmountInOrderCurrency, domain tcamnt iCommissionRebateAmountInOrderCurrency, domain tccur iCommissionRebateInvoiceCurrency, ref boolean oCommissionRebateAdded, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl.: This function generates a commission/rebate line for commissions/rebates which are calculated for a sales order (line) based on Agreement Grades Before adding new commission/rebate lines, a counter posting is created for existing commission/rebate commissions/rebates that are already processed. Pre: Retry point must be set Post: Transaction must be committed or aborted. Input: iCommissionRebateType - Type Commission or Rebate iAgreementSearchPriority - Agreement search priority as defined in Commissions and Rebates parameters iAgreementInvoiceDirectReserve - Agreement Invoice/Reserve iAgreementDiscountCode - Agreement Discount Code iSalesOrder - Sales Order iSalesOrderLine - Sales Order Line iSalesOrderSequence - Sales Sequence Number, it is 0 in case the Commission/Rebate Parameter 'Linking of Relations On' is set to Sales Order iSalesOrderDeliverySequence - Sales Order Delivery Sequence Line iSalesOrderInvoiceLine - Sales Order Invoice Line iSalesOrderTransactionType - Sales Order Line Transaction Type iSalesOrderInvoiceNumber - Sales Order Line Invoice Number iSalesOrderInvoiceDate - Sales Order Line Invoice Date iSalesOrderOffice - Sales Order Line Sales Office iSalesOrderFinancialDepartment - Sales Order Line Financial Department iSalesOrderCurrency - Sales Order Currency iSalesOrderRatedate - Sales Order Line Rate Date iSalesOrderRatetype - Sales Order Line Rate Type iSalesOrderRate - Sales Order Line Rate iSalesOrderRateFactor - Sales Order Line Rate Factor iRelation - Relation by sales order line iRelationType - Relation Type: Employee/Supplier/Customer iRelationSequence - Relation sequence by order line iCommissionRebatePercentage - Commissions/Rebates according Commissions/Rebates Agreement Grades iCommissionRebateGrowPercentage - Commissions/Rebates Grow Percentage according Commissions/Rebates Agreement Grades iCommissionRebateGrowAmount

	- Commissions/Rebates Grow Amount based on Commissions/Rebates Agreement Grades Grow Percentage
iCommissionRebateFixedAmountInOrderCurrency	- Commissions/Rebates Fixed Amount according Commissions/Rebates Agreement Grades
iCommissionRebateSalesOrderAmountInOrderCurrency	- Sales Order (Line) Amount in sales order line currency
iCommissionRebateAmountInOrderCurrency	- Commissions/Rebates Amount in order currency based on Commissions/Rebates Agreement Grades Commission/Rebate Percentage
iCommissionRebateInvoiceCurrency	- Commissions/Rebates Invoice Currency based on invoice currency for relation
Output: oCommissionRebateAdded	- True/False. Indicates that Commission/Rebate line has been inserted
oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
oExceptionID	An ID that refers to all error information. Use the functions in Exception to get all relevant information.
Return: 0	Insert of commission/rebate amount was successfull
<> 0	An error occurred.

CommissionsAndRebates.InsertCumulativeSales

DLL:	tdextcmsapi
Syntax:	<pre> long CommissionsAndRebates.InsertCumulativeSales(domain tdcms.type iCommissionRebateType, domain tdcms.prty iAgreementSearchPriority, domain tccom.bpid iAgreementSoldToBusinessPartner, domain tccom.bpid iAgreementRelation, domain tdcms.agrp iAgreementGroup, domain tccprj iAgreementProject, domain tcitem iAgreementItem, domain tdcms.cmgrp iAgreementCommissionRebateGroup, domain tcccur iAgreementCurrency, domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesSequence, domain tcpono iSalesDeliverySequence, domain tcpono iSalesInvoiceLine, domain tccwoc iSalesOrderFinancialDepartment, domain tcccur iSalesOrderCurrency, domain tdcms.stat iRelationByStatus, domain tcpono iRelationSequence, </pre>

	<pre> domain tdcms.cupf iRelationCurrencyForInvoicing, domain tdcms.rtyp iRelationType, domain tccom.bpid iRelationBuyFromBusinessPartner, domain tccom.bpid iRelationSoldToBusinessPartner, domain tcccp.cpdtdt iRelationPeriodTable, domain tcdate iSearchDate, domain tcyesno iFullUpdate, ref boolean oCumulativeSalesAdded, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl.: *** Warning *** This public interface is deprecated and will be obsolete from LN Cloud 2021.04 onwards; use: CommissionsAndRebates.InsertCumulativeSalesV2 Function evaluates whether commissions rebates cumulative sales should be added either by order or order line and inserts cumulative sales lines. This logic is executed after calculating commissions and rebates for those cumulative agreements. Pre: Retry point must be set Post: Transaction must be committed or aborted. Input: iCommissionRebateType - Type Commission or Rebate iAgreementSearchPriority - Agreement search priority as defined in Commissions and Rebates parameters iAgreementSoldToBusinessPartner - Agreement Sold-to Business Partner iAgreementRelation - Agreement Relation iAgreementGroup - Agreement Group iAgreementProject - Agreement Project iAgreementItem - Agreement Item iAgreementCommissionRebateGroup - Agreement Commission/Rebate Group iAgreementCurrency - Agreement Currency iSalesOrder - Sales Order iSalesOrderLine - Sales Order Line iSalesOrderSequence - Sales Sequence Number, it is 0 in case the Commission/Rebate Parameter 'Linking of Relations On' is set to Sales Order iSalesOrderDeliverySequence - Sales Order Delivery Sequence Line iSalesOrderInvoiceLine - Sales Order Invoice Line iSalesOrderFinancialDepartment - Sales Order Line Financial Department iSalesOrderCurrency - Sales Order Currency iRelationByStatus - The status of the relation linked to sales order (line) iRelationSequence - Relation sequence by order line iRelationCurrencyForInvoicing - Relation Currency for Invoicing iRelationType - Relation Type: Employee/Supplier/ Customer iRelationBuyFromBusinessPartner - Relation Buy-from BP, only applicable in case the relation type is 'Supplier' iRelationSoldToBusinessPartner - Relation Buy-from BP, only applicable in case the relation type is </pre>

		'Customer'
	iRelationPeriodTable	- Relation Period Table
	iSearchDate	- Search date based on Commissions/ Rebates parameter 'Search for Commission Based on Date'
	iFullUpdate	- Yes/No. Indicate if previous calculated Cumulative Commissions and Rebates need to be removed and calculated again
	Output: oCumulativeSalesAdded	- True/False. Indicates that Cumulative Sales Commission/Rebate line has been inserted.
	oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
	oExceptionID	An ID that refers to all error information. Use the functions in Exception to get all relevant information.
	Return: 0	Insert of commission/rebate amount was successfull
	<> 0	An error occurred

CommissionsAndRebates.InsertCumulativeSalesV2

DLL:	tdextcmsapi
Syntax:	<pre> long CommissionsAndRebates.InsertCumulativeSalesV2(domain tdcms.type iCommissionRebateType, domain tdcms.prt iAgreementSearchPriority, domain tccom.bpid iAgreementSoldToBusinessPartner, domain tccom.bpid iAgreementRelation, domain tdcms.agrp iAgreementGroup, domain tccprj iAgreementProject, domain tcitem iAgreementItem, domain tdcms.cmgrp iAgreementCommissionRebateGroup, domain tcccur iAgreementCurrency, domain tcdate iAgreementEffectiveDate, domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesSequence, domain tcpono iSalesDeliverySequence, domain tcpono iSalesInvoiceLine, domain tccwoc iSalesOffice, domain tccwoc iSalesOrderFinancialDepartment, domain tcccur iSalesOrderCurrency, ref domain tcratc iSalesOrderCurrencyRate(), ref domain tcratf iSalesOrderCurrencyRateFactor(), domain tcdate iSalesOrderCurrencyRateDate, domain tcrtyp iSalesOrderCurrencyRateType, domain tdcms.stat iRelationByStatus, domain tcpono iRelationSequence, domain tdcms.cmcm iRelationCalculationMethod, domain tdcms.ruci iRelationRateUsedForCalculating, </pre>

	<pre> domain tdcms.exrt iRelationRateType, domain tcrtyp iRelationRateTypeUsed, domain tdcms.cupf iRelationCurrencyForInvoicing, domain tdcms.rtyp iRelationType, domain tccom.bpid iRelationBuyFromBusinessPartner, domain tccom.bpid iRelationSoldToBusinessPartner, domain tcccp.cpdtdt iRelationPeriodTable, domain tcdate iSearchDate, domain tcyesno iFullUpdate, ref boolean oCumulativeSalesAdded, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl.: Function evaluates whether commissions rebates cumulative sales should be added either by order or order line and inserts cumulative sales lines. This logic is executed after calculating commissions and rebates for those cumulative agreements. Pre: Retry point must be set Post: Transaction must be committed or aborted. Input: iCommissionRebateType - Type Commission or Rebate (Mandatory) iAgreementsearchPriority - Agreement search priority as defined in Commissions and Rebates parameters (Mandatory) iAgreementSoldToBusinessPartner - Agreement Sold-to Business Partner iAgreementRelation - Agreement Relation iAgreementGroup - Agreement Group iAgreementProject - Agreement Project iAgreementItem - Agreement Item iAgreementCommissionRebateGroup - Agreement Commission/Rebate Group iAgreementCurrency - Agreement Currency iAgreementEffectiveDate - Agreement Effective Date iSalesOrder - Sales Order (Mandatory) iSalesOrderLine - Sales Order Line iSalesOrderSequence - Sales Sequence Number, it is 0 in case the Commission/Rebate Parameter 'Linking of Relations On' is set to Sales Order iSalesOrderDeliverySequence - Sales Order Delivery Sequence Line iSalesOrderInvoiceLine - Sales Order Invoice Line iSalesOffice - Sales Order Sales Office iSalesOrderFinancialDepartment - Sales Order Line Financial Department iSalesOrderCurrency - Sales Order Currency iSalesOrderCurrencyRate - Sales Order Currency Rate iSalesOrderCurrencyRateFactor - Sales Order Currency Rate Factor iSalesOrderCurrencyRateDate - Sales Order Currency Rate Date iSalesOrderCurrencyRateType - Sales Order Currency Rate Type iRelationByStatus - The status of the relation linked to sales order (line) iRelationSequence - Relation sequence by order line iRelationCalculationMethod - Relation Calculation Method - Order - Invoice - Paid Sales Invoice </pre>

	iRelationRateUsedForCalculating - Relation Rate used for Calculating/Invoicing - Sales Order Date - Sales Invoice Rate - Sales Invoice Date - Calculation Date iRelationRateType - Relation Rate Type - Sales Order - Commissions/Rebates Parameter - User Defined - Not Applicable iRelationRateTypeUsed - Relation Rate Type Used iRelationCurrencyForInvoicing - Relation Currency for Invoicing iRelationType - Relation Type: Employee/Supplier/ Customer iRelationBuyFromBusinessPartner - Relation Buy-from BP, only applicable in case the relation type is 'Supplier' iRelationSoldToBusinessPartner - Relation Buy-from BP, only applicable in case the relation type is 'Customer' iRelationPeriodTable - Relation Period Table (Mandatory) iSearchDate - Search date based on Commissions/ Rebates parameter 'Search for Commission Based on Date' (Mandatory) iFullUpdate - Yes/No. Yes: Indicates if previous calculated Cumulative Commissions and Rebates needs to be removed and calculated again (Mandatory) Output: oCumulativeSalesAdded - True/False. Indicates that Cumulative Sales Commission/Rebate line has been inserted. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Insert of commission/rebate amount was successfull <> 0 An error occurred

Public Interfaces for Rebate

The following functions are available:

[Rebates.StartOverview](#)

Rebates.StartOverview

DLL:	tdextcmsapi This function is available from KB2262331 .
Syntax:	<pre> long Rebates.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderSequence, domain tcpono iActualDeliverySequence, domain tcpono iInvoiceLine, domain tccom.bpid iRelation, domain tcpono iSerialNumber, ref domain tcorno oSalesOrder, ref domain tcpono oSalesOrderLine, ref domain tcpono oSalesOrderSequence, ref domain tcpono oActualDeliverySequence, ref domain tcpono oInvoiceLine, ref domain tccom.bpid oRelation, ref domain tcpono oSerialNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Rebates Overview (tdcms2550m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Specifies the table-index that will be used. iQueryExtend A specific query to be used when zooming to this session. iSalesOrder Sales Order. Mandatory if iStartMode is MODELESS and session index 1 is used. iSalesOrderLine Sales Order Line Not mandatory iSalesOrderSequence Sales Order Sequence Not mandatory iActualDeliverySequence Actual Delivery Sequence Not mandatory iInvoiceLine Invoice Line Not mandatory

iRelation	Relation
iSerialNumber	Mandatory if iStartMode is MODELESS and session index 2 is used.
Output: for iStartMode MODAL:	Serial Number
oSalesOrder	Not mandatory
oSalesOrderLine	The sales order of the selected record
oSalesOrderSequence	The sales order line of the selected record
oActualDeliverySequence	The sales order sequence of the selected record
oInvoiceLine	The actual delivery sequence of the selected record
oRelation	The invoice line of the selected record
oSerialNumber	The relation of the selected record
oExceptionMessage	The serial number of the selected record
oExceptionID	The last message if the return value is not equal to 0.
Return: 0	If more than one message is given, these are present in the oExceptionID
<> 0	An ID that refers to all error information. Use the functions in Exception to get all relevant information.
	Session started
	An error occurred

Public Interfaces for Commission

The following functions are available:

[Commissions.StartOverview](#)

Commissions.StartOverview

DLL:	tdextcmsapi
	This function is available from KB2262331 .
Syntax:	<pre>long Commissions.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(),</pre>

	domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, domain tcpono iSalesOrderSequence, domain tcpono iActualDeliverySequence, domain tcpono iInvoiceLine, domain tccom.bpid iRelation, domain tcpono iSerialNumber, ref domain tcorno oSalesOrder, ref domain tcpono oSalesOrderLine, ref domain tcpono oSalesOrderSequence, ref domain tcpono oActualDeliverySequence, ref domain tcpono oInvoiceLine, ref domain tccom.bpid oRelation, ref domain tcpono oSerialNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function starts the session Commissions Overview (tdcms1550m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Specifies the table-index that will be used. iQueryExtend A specific query to be used when zooming to this session. iSalesOrder Sales Order. Mandatory if iStartMode is MODELESS and session index 1 is used. iSalesOrderLine Sales Order Line Not mandatory iSalesOrderSequence Sales Order Sequence Not mandatory iActualDeliverySequence Actual Delivery Sequence Not mandatory iInvoiceLine Invoice Line Not mandatory iRelation Relation Mandatory if iStartMode is MODELESS and session index 2 is used. iSerialNumber Serial Number Not mandatory Output: for iStartMode MODAL: oSalesOrder The sales order of the selected record oSalesOrderLine The sales order line of the selected record oSalesOrderSequence The sales order sequence of the selected record oActualDeliverySequence The actual delivery sequence of the selected record oInvoiceLine The invoice line of the selected record oRelation The relation of the selected record oSerialNumber The serial number of the selected record oExceptionMessage The last message if the return

	oExceptionID Return: 0 <> 0	value is not equal to 0. If more than one message is given, these are present in the oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Session started An error occurred

Chapter 11 Public Interfaces for Product Catalogs

Public Interfaces for Catalog

The following functions are available:

[Catalog.StartOverview](#)

Catalog.StartOverview

DLL:	tdextpctapi
Syntax:	<pre> long Catalog.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcmcs.str6 iCategory, domain tcseek iSearchKey mb, domain tcyesno iCatalog, ref domain tcmcs.str6 oCategory, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Catalogs in overview mode (tdpct0510m000). Input: <code>iStartMode</code> Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. <code>iStartFilter</code> Not Used. <code>iSessionIndex</code> The index that will be used. Supported values: 1: sort by Category 2: sort by Search Key/Category 3: sort by Catalog/Category <code>iQueryExtend</code> A specific query to be used when zooming to this session.

	<pre>iCategory iSearchKey iCatalog Output: for iStartMode MODAL: oCategory oExceptionMessage oExceptionID Return: 0 <> 0</pre>	<pre>Category Search Key Catalog Category of the selected line The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Session started An error occurred</pre>

Chapter 12 Public Interfaces for Purchase

Public Interfaces for Purchase

The following functions are available:

[Purchase.GenerateRetrobilledPriceChangeAdvices](#)

[Purchase.GenerateRetrobilledPriceChangeAdvicesForContracts](#)

[Purchase.GetContractSettings](#)

[Purchase.GetGeneralSettings](#)

[Purchase.GetOrderSettings](#)

[Purchase.GetRequisitionSettings](#)

[Purchase.GetRFQSettings](#)

[Purchase.GetScheduleSettings](#)

Purchase.GenerateRetrobilledPriceChangeAdvices

DLL:	tdextpurapi
Syntax:	<pre> long Purchase.GenerateRetrobilledPriceChangeAdvices (domain tcorno iAdviceNumber, boolean iGenerateForOrders, boolean iGenerateForSchedules, boolean iGenerateForReceipts, domain tcorno iPurchaseOrderFrom, domain tcorno iPurchaseOrderTo, domain tcpono iOrderLineFrom, domain tcpono iOrderLineTo, domain tcorno iPurchaseScheduleFrom, domain tcorno iPurchaseScheduleTo, domain tcorno iReceiptFrom, domain tcorno iReceiptTo, domain tdpur.corg iOrderOriginFrom, domain tdpur.corg iOrderOriginTo, domain tccom.bpid iBuyFromBusinessPartnerFrom, </pre>

276 | Infor LN Public Interfaces & Process Extensions Reference Guide (On-premises)

	iShipFromBusinessPartnerTo - Ship-from Business Partner to iInvoiceFromBusinessPartnerFrom - Invoice-from Business Partner from iInvoiceFromBusinessPartnerTo - Invoice-from Business Partner to iContractFrom - Contract from iContractTo - Contract to iContractLineFrom - Contract Line from iContractLineTo - Contract Line to iDateFrom - Date from iDateTo - Date to iItemFrom - Item from iItemTo - Item to iUpdateType - Update Type The method used to update the existing price of the purchase document. Possible values: - Value: - The price update is a fixed amount. - Percentage: - The price update is a percentage. iUpdatePercentage - Update Percentage The percentage by which the existing price of the specified purchase documents is updated. iUpdateAmount - Update Amount The value by which the existing price of the specified purchase documents is updated. iCurrency - Currency iShowProgressIndicator - Yes/No Indicates if showing the progress indicator is allowed. Yes: If applicable, the progress indicator is shown. No: The progress indicator will not be shown. Output: oAdviceNumber - Advice Header that is used. If the given Advice (iAdviceNumber) did not exist yet, then this argument contains the created Advice. oAdviceLinesGenerated - True/False Indicates if advice lines have been generated. oProcessStopped - True/False Indicates if the process was aborted by the user. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Function was executed successful. Price change advices may have been generated. <> 0 - An error occurred during the execution of the function.

Purchase.GenerateRetrobilledPriceChangeAdvicesForContracts

DLL:	tdextpurapi																
Syntax:	<pre> long Purchase.GenerateRetrobilledPriceChangeAdvicesForContracts(domain tccono iPurchaseContract, domain tccono iChangeRequest, domain tcorno iAdviceNumber, domain tcyesno iGenerateForOrders, domain tcyesno iGenerateForSchedules, domain tcyesno iShowProgressIndicator, ref domain tcorno oAdviceNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>																
Usage:	Expl: This function will generate Retro-Billed Advices* and Advice Lines based on the given input parameters. The price change is calculated from the given Contract (that has the newest prices and discounts) against the Orders and/or Schedules that have a link to that Contract. * If the given Advice Number (iAdviceNumber) is a series, then a new Advice header will be created based upon that series. If it is an existing Advice, then the given Advice will be used. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: <table> <tr> <td>iPurchaseContract</td><td>- Purchase Contract: Mandatory</td></tr> <tr> <td>iChangeRequest</td><td>- Contains the change request number that initiated the price change. Must be empty when the price change was not initiated by a change request.</td></tr> <tr> <td>iAdviceNumber</td><td>- Advice Number Can be a series or an Advice ID.</td></tr> <tr> <td>iGenerateForOrders</td><td>- Yes/No Indicates if Retro Billed advices must be generated for orders.</td></tr> <tr> <td>iGenerateForSchedules</td><td>- Yes/No Indicates if Retro Billed advices must be generated for schedules</td></tr> <tr> <td>iShowProgressIndicator</td><td>- Yes/No Indicates if showing the progress indicator is allowed. Yes: If applicable, the progress indicator is shown. No: The progress indicator will not be shown.</td></tr> </table> Output <table> <tr> <td>oAdviceNumber</td><td>- Generated Advice Number</td></tr> <tr> <td>oExceptionMessage</td><td>- The last message if the return value is not equal to 0. If more than one message is</td></tr> </table>	iPurchaseContract	- Purchase Contract: Mandatory	iChangeRequest	- Contains the change request number that initiated the price change. Must be empty when the price change was not initiated by a change request.	iAdviceNumber	- Advice Number Can be a series or an Advice ID.	iGenerateForOrders	- Yes/No Indicates if Retro Billed advices must be generated for orders.	iGenerateForSchedules	- Yes/No Indicates if Retro Billed advices must be generated for schedules	iShowProgressIndicator	- Yes/No Indicates if showing the progress indicator is allowed. Yes: If applicable, the progress indicator is shown. No: The progress indicator will not be shown.	oAdviceNumber	- Generated Advice Number	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is
iPurchaseContract	- Purchase Contract: Mandatory																
iChangeRequest	- Contains the change request number that initiated the price change. Must be empty when the price change was not initiated by a change request.																
iAdviceNumber	- Advice Number Can be a series or an Advice ID.																
iGenerateForOrders	- Yes/No Indicates if Retro Billed advices must be generated for orders.																
iGenerateForSchedules	- Yes/No Indicates if Retro Billed advices must be generated for schedules																
iShowProgressIndicator	- Yes/No Indicates if showing the progress indicator is allowed. Yes: If applicable, the progress indicator is shown. No: The progress indicator will not be shown.																
oAdviceNumber	- Generated Advice Number																
oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is																

	given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Function was executed successfull. Price change advices may have been generated. <> 0 - An error occurred during the execution of the function.

Purchase.GetContractSettings

DLL:	tdextpurapi
Syntax:	<pre>long Purchase.GetContractSettings(domain tcncmp iLogisticCompany, domain tcsite iSite, domain tccwoc iPurchaseOffice, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...)</pre>
Usage:	Expl: This function retrieves settings for Purchase Contracts Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tdpur000), or site or office level (settings table tdpur083) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when purchase list group and configutaion date for PCS are needed, the function must be called as follows: <pre>if Purchase.GetPurchaseContractSettings(* Fixed arguments: company, --> input site, --> input purchase office, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output</pre>

	<pre> /* Variable arguments: "ngpc.3", --> input number.group.for.contracts, --> output "sepc", --> input contract.series, --> output "rpcs.3", --> input link.with.pcs.module) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif </pre>
Pre:	None
Post:	None
Input:	<pre> iLogisticCompany - Logistic Company: Mandatory iSite - Site: Not Mandatory iPurchaseOffice - Purchase Office: Not Mandatory iForceRead - Option to force new query in stead of using cached information iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. </pre>
Output:	<pre> oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. </pre>
Return:	<pre> 0 - Data read <> 0 - An error occurred </pre>

Purchase.GetGeneralSettings

DLL:	tdextpurapi
Syntax:	<pre> long Purchase.GetGeneralSettings(domain tcncmp iLogisticCompany, domain tcsite iSite, domain tccwoc iPurchaseOffice, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	<pre> Expl: This function retrieves general settings for purchase. Individual fields are retrieved, multiple fields can be retrieved in one call. </pre>

This function is Multi Site aware, and data is read from general level (parameter table tdpur000), or site or office level (settings table tdpur080) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site.

Notes:

1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved.
2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value.

Example: when purchase list group and configuration date for PCS are needed, the function must be called as follows:

```
if Purchase.GetGeneralSettings(
    /* Fixed arguments:

        company,          --> input
        site,             --> input
        purchase office,  --> input
        force.read,       --> input
        context is master data --> input
        exception.message, --> output
        exception.id,     --> output

    /* Variable arguments:

        "clgr",          --> input
        list.group.purchase, --> output
        "codt",          --> input
        configuration.date.pcs) <> 0 then --> output
    /* Error, do something
        Exception.Delete(exception.id)
endif
```

Pre: None

Post: None

Input:

iLogisticCompany	- Logistic Company: Mandatory
iSite	- Site: Not Mandatory
iPurchaseOffice	- Purchase Office: Not Mandatory
iForceRead	- Option to force new query in stead of using cached information
iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.
...	- The field mnemonic of the required field.

Output:

oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
...	- The value of the required field.

	Return: 0 <> 0	- Data read - An error occurred

Purchase.GetOrderSettings

DLL:	tdextpurapi
Syntax:	<pre> long Purchase.GetOrderSettings(domain tcncmp iLogisticCompany, domain tcsite iSite, domain tccwoc iPurchaseOffice, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves settings for Purchase Orders. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tdpur000), or site or office level (settings table tdpur084) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. 3. Several table-fields are stored as arrays. Individual elements of these table-fields can be retrieved by adding the element number to the field-mnemonic. Example: "comm(7)", "ctrl(8)". Example: when number group for orders, series for purchase invoice and confirm receipt as final are needed, the function must be called as follows: <pre> if Purchase.GetPurchaseOrderSettings(* Fixed arguments: company, --> input site, --> input purchase office, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output * Variable arguments: "ngpo.4", --> input number.group.for.orders, --> output "sepi", --> input series.for.purchase.invoice, --> output </pre>

	<pre> "crsf", --> input confirm.receipt.as.final) <> 0 then --> output * Error, do something Exception.Delete(exception.id) endif Pre: None Post: None Input: iLogisticCompany - Logistic Company: Mandatory iSite - Site: Not Mandatory iPurchaseOffice - Purchase Office: Not Mandatory iForceRead - Option to force new query in stead of using cached information iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - An error occurred </pre>

Purchase.GetRequisitionSettings

DLL:	tdextpurapi
Syntax:	<pre> long Purchase.GetRequisitionSettings(domain tcncmp iLogisticCompany, domain tcscite iSite, domain tccwoc iPurchaseOffice, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves settings for Purchase Requisitions. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tdpur000), or site or office level (settings table tdpur082) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site.

	Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when purchase list group and configuration date for PCS are needed, the function must be called as follows: <pre> if Purchase.GetPurchaseRequisitionSettings(/* Fixed arguments: company, --> input site, --> input purchase office, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output /* Variable arguments: ngpr.2", --> input number.group.for.requisitions, --> output "sepr", --> input series.for.requisitions) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif </pre> Pre: None Post: None Input: <table> <tr> <td>iLogisticCompany</td><td>- Logistic Company: Mandatory</td></tr> <tr> <td>iSite</td><td>- Site: Not Mandatory</td></tr> <tr> <td>iPurchaseOffice</td><td>- Purchase Office: Not Mandatory</td></tr> <tr> <td>iForceRead</td><td>- Option to force new query in stead of using cached information</td></tr> <tr> <td>iContextIsMasterData</td><td>- True/False: Indicates if call is done to retrieve data for master data or transactional data.</td></tr> <tr> <td>...</td><td>- The field mnemonic of the required field.</td></tr> </table> Output: <table> <tr> <td>oExceptionMessage</td><td>- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID</td></tr> <tr> <td>oExceptionID</td><td>- An ID that refers to all error information. Use the functions in Exception to get all relevant information.</td></tr> <tr> <td>...</td><td>- The value of the required field.</td></tr> </table> Return: 0 <table> <tr> <td><> 0</td><td>- Data read</td></tr> <tr> <td></td><td>- An error occurred</td></tr> </table>	iLogisticCompany	- Logistic Company: Mandatory	iSite	- Site: Not Mandatory	iPurchaseOffice	- Purchase Office: Not Mandatory	iForceRead	- Option to force new query in stead of using cached information	iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.	...	- The field mnemonic of the required field.	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.	...	- The value of the required field.	<> 0	- Data read		- An error occurred
iLogisticCompany	- Logistic Company: Mandatory																						
iSite	- Site: Not Mandatory																						
iPurchaseOffice	- Purchase Office: Not Mandatory																						
iForceRead	- Option to force new query in stead of using cached information																						
iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.																						
...	- The field mnemonic of the required field.																						
oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID																						
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.																						
...	- The value of the required field.																						
<> 0	- Data read																						
	- An error occurred																						

Purchase.GetRFQSettings

DLL:	tdextpurapi								
Syntax:	<pre> long Purchase.GetRFQSettings(domain tcncmp iLogisticCompany, domain tcsite iSite, domain tccwoc iPurchaseOffice, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>								
Usage:	Expl: This function retrieves settings for purchase RFQs. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tdpur000), or site or office level (settings table tdpur081) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when purchase list group and configutaion date for PCS are needed, the function must be called as follows: <pre> if Purchase.GetRFQSettings(/* Fixed arguments: company, --> input site, --> input purchase office, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output /* Variable arguments: "enlh.1", --> input enhanced.line.handling, --> output "bor1", --> input buyer.on.rfq.line) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif </pre> Pre: None Post: None Input: <table> <tr> <td>iLogisticCompany</td><td>- Logistic Company: Mandatory</td></tr> <tr> <td>iSite</td><td>- Site: Not Mandatory</td></tr> <tr> <td>iPurchaseOffice</td><td>- Purchase Office: Not Mandatory</td></tr> <tr> <td>iForceRead</td><td>- Option to force new query in stead of</td></tr> </table>	iLogisticCompany	- Logistic Company: Mandatory	iSite	- Site: Not Mandatory	iPurchaseOffice	- Purchase Office: Not Mandatory	iForceRead	- Option to force new query in stead of
iLogisticCompany	- Logistic Company: Mandatory								
iSite	- Site: Not Mandatory								
iPurchaseOffice	- Purchase Office: Not Mandatory								
iForceRead	- Option to force new query in stead of								

	using cached information iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - An error occurred

Purchase.GetScheduleSettings

DLL:	tdextpurapi
Syntax:	<pre>long Purchase.GetScheduleSettings(domain tcncmp iLogisticCompany, domain tcsite iSite, domain tccwoc iPurchaseOffice, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...)</pre>
Usage:	Expl: This function retrieves settings for Purchase Schedules. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tdpur000), or site or office level (settings table tdpur085) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. 3. Several table-fields are stored as arrays. Individual elements of these table-fields can be retrieved by adding the element number to the field-mnemonic. Example: "crr1(3)", "crr3(2)". Example: when number group for Call-offs and cum-model used are needed, the function must be called as follows:

	<pre> if Purchase.GetPurchaseScheduleSettings(/* Fixed arguments: company, --> input site, --> input purchase office, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output /* Variable arguments: "ngco.3", --> input number.group.for.calloffs, --> output "cumm", --> input cum.model.used) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif </pre>
Pre:	None
Post:	None
Input:	<pre> iLogisticCompany - Logistic Company: Mandatory iSite - Site: Not Mandatory iPurchaseOffice - Purchase Office: Not Mandatory iForceRead - Option to force new query in stead of using cached information iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. </pre>
Output:	<pre> oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. </pre>
Return:	<pre> 0 - Data read <> 0 - An error occurred </pre>

Public Interfaces for RequestForQuoteLine

The following functions are available:

[RequestForQuoteLines.StartOverview](#)

RequestForQuoteLines.StartOverview

DLL:	tdextpurapi
	This function is available from KB3501645 .
Syntax:	<pre> long RequestForQuoteLines.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcqono iRequestForQuote, domain tcpono iRequestForQuoteLine, domain tcpono iRequestForQuoteSequence, ref domain tcqono oRequestForQuote, ref domain tcpono oRequestForQuoteLine, ref domain tcpono oRequestForQuoteSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the Overview session Request for Quotation Lines (tdpurl502ml00). Input: iStartMode Specifies the start mode for the session (Mandatory). Possible values are: MODAL The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Start filter (Not used). iSessionIndex Session index (Not used). iQueryExtend A specific query to be used when zooming to this session (Optional). iRequestForQuote Request for Quote (Optional). iRequestForQuoteLine Request for Quote Line (Optional). iRequestForQuoteSequence Request for Quote Sequence (Optional). Output: For iStartMode MODAL: oRequestForQuote The selected Request for Quote oRequestForQuoteLine The selected Request for Quote Line oRequestForQuoteSequence The selected Request for Quote Sequence oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information.

	Return: 0 <> 0	Session started An error occurred

Public Interfaces for RequestForQuoteResponse

The following functions are available:

[RequestForQuoteResponse.Accept](#)

[RequestForQuoteResponse.Reject](#)

RequestForQuoteResponse.Accept

DLL:	tdextpurapi
	This function is available from KB3634332 .
Syntax:	<pre> long RequestForQuoteResponse.Accept (domain tcqono iRequestForQuote, domain tcpono iRequestForQuoteLine, domain tcpono iRequestForQuoteResponseSequence, domain tccom.bpid iRequestForQuoteBidder, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function accepts a request for quotation response. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: - iRequestForQuote - Request for Quote (mandatory) - iRequestForQuoteLine - Request for Quote Line (mandatory) - iRequestForQuoteResponseSequence - Request for Quote Response Sequence (optional). The database field "tdpur106.srnbn" is used here - iRequestForQuoteBidder - Request for Quote Bidder (mandatory) - iProcessingOptionSet - Processing Option Set (optional). If 0, the default options are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcexttextapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete() Any options which are not available as Processing Option will get defaulted. Processing Options which are set while a required Implemented Software Component is not available are ignored. Supported Processing Option and the default:

NAME	TYPE	DEFAULT
AcknowledgeUnacknowledgedNegotiationLine	domain tcyesno	tcyesno.yes
Yes: An unacknowledged negotiation line is automatically acknowledged		
No: An unacknowledged negotiation line is not automatically acknowledged		
Output: oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID	
oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information	
Return: 0	- The Request for Quote is accepted	
<> 0	- An error occurred	

RequestForQuoteResponse.Reject

DLL:	tdextpurapi						
	This function is available from KB3634332 .						
Syntax:	<pre>long RequestForQuoteResponse.Reject (domain tcqono iRequestForQuote, domain tcpono iRequestForQuoteLine, domain tcpono iRequestForQuoteResponseSequence, domain tccom.bpid iRequestForQuoteBidder, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>						
Usage:	Expl.: This function rejects a request for quotation response. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iRequestForQuote - Request for Quote (mandatory) iRequestForQuoteLine - Request for Quote Line (mandatory) iRequestForQuoteResponseSequence - Request for Quote Response Sequence (optional). The database field "tdpur106.srnbn" is used here iRequestForQuoteBidder - Request for Quote Bidder (mandatory) iProcessingOptionSet - Processing Option Set (optional). If 0, the default options are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcexttextapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete() Any options which are not available as Processing Option will get defaulted. Processing Options which are set while a required Implemented Software Component is not available are ignored. Supported Processing Option and the default: <table><tr><td>NAME</td><td>TYPE</td><td>DEFAULT</td></tr><tr><td>AcknowledgeUnacknowledgedNegotiationLine</td><td>domain tcyesno</td><td>tcyesno.yes</td></tr></table>	NAME	TYPE	DEFAULT	AcknowledgeUnacknowledgedNegotiationLine	domain tcyesno	tcyesno.yes
NAME	TYPE	DEFAULT					
AcknowledgeUnacknowledgedNegotiationLine	domain tcyesno	tcyesno.yes					

	Yes: An unacknowledged negotiation line is automatically acknowledged No: An unacknowledged negotiation line is not automatically acknowledged Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information Return: 0 - The Request for Quote is rejected <> 0 - An error occurred

Public Interfaces for PurchaseRequisition

The following functions are available:

[PurchaseRequisition.ApproveApprovalRecord](#)

[PurchaseRequisition.RejectApprovalRecord](#)

[PurchaseRequisition.Submit](#)

PurchaseRequisition.ApproveApprovalRecord

DLL:	tdextpurapi
Syntax:	<pre>long PurchaseRequisition.ApproveApprovalRecord(domain tcrqno iPurchaseRequisition, domain tcsern iSequenceNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl.: This function can be used to approve the given Requisition Approval Progress record. Approval is only allowed if all previous records have already been approved. This function checks whether the current user is allowed to perform the approval. The purchase requisition parameter/setting 'Approval Authorizations' (tdpur000.apau.2/tdpur082.apau) is taken into account for this. This function cannot be used if approval of requisitions is done through Workflow. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iPurchaseRequisition - Purchase Requisition; Mandatory. iSequenceNumber - Sequence number of the Approval Progress record; Mandatory Output: oExceptionMessage - The last message if the return

	value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - The Approval Progress record is approved. - An error occurred

PurchaseRequisition.RejectApprovalRecord

DLL:	tdextpurapi
	This function is available from KB3632205 .
Syntax:	<pre>long PurchaseRequisition.RejectApprovalRecord(domain tcrqno iPurchaseRequisition, domain tcsern iSerialNumber, domain tcddis iReasonForRejection, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl.: This function can be used to reject the given Requisition Approval Progress record. Rejection of a record is only allowed if the previous records have been approved. This function checks whether the current user is allowed to perform the rejection. The purchase requisition parameter/setting 'Approval Authorizations' (tdpur000.apau.2/tdpur082.apau) is taken into account for this. This function cannot be used if rejection of requisitions is done through Workflow. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iPurchaseRequisition - Purchase requisition (mandatory) iSerialNumber - Serial number of the Approval Progress record (mandatory) iReasonForRejection - Reason for rejection of the approval record; this input argument overwrites any existing reason for rejection. Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - The Approval Progress record is rejected. - An error occurred

PurchaseRequisition.Submit

DLL:	tdextpurapi
	This function is available from KB3632205 .
Syntax:	<pre> long PurchaseRequisition.Submit(domain tcrqno iPurchaseRequisition, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function can be used to submit the given Requisition. This function checks whether the current user is allowed to perform the submitting. The purchase requisition parameter/setting 'Approval Authorizations' (tdpur000.apau.2/tdpur082.apau) is taken into account for this. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iPurchaseRequisition - Purchase requisition (mandatory) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - The Purchase requisition is submitted. <> 0 - An error occurred

Public Interfaces for PurchaseRequisitionLine

The following functions are available:

[PurchaseRequisitionLine.ConvertToPurchaseOrder](#)

[PurchaseRequisitionLine.ConvertToPurchaseRFQ](#)

[PurchaseRequisitionLine.StartDetail](#)

PurchaseRequisitionLine.ConvertToPurchaseOrder

DLL:	tdextpurapi
Syntax:	<pre> long PurchaseRequisitionLine.ConvertToPurchaseOrder(domain tcrqno iPurchaseRequisition, domain tcpono iRequisitionLine, </pre>

	domain tcseri iOrderSeries, domain tccotp iOrderType, domain tcseri iSubcontractingOrderSeries, domain tccotp iSubcontractingOrderType, domain tcseri iServiceSubcontractingOrderSeries, domain tccotp iServiceSubcontractingOrderType, domain tcyesno iCalculateNewPriceAndDiscounts, domain tcyesno iCurrencyFromBusinessPartner, domain tcorno iPreviousPurchaseOrder, domain tcpono iPreviousPurchaseOrderLine, ref domain tcorno oGeneratedOrder, ref domain tcpono oGeneratedOrderLine, ref domain tcpono oGeneratedOrderSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function converts the given requisition line to a purchase order. Conversion is only allowed if the requisition header has the proper status (i.e. Approved or In Process), and the requisition line is not rejected. The Conversion Type of the requisition line must be 'Purchase Order'. When input argument iPreviousPurchaseOrder is filled, this function tries to add a line to that order. If the requisition line does not match with the given order, then a new purchase order will be generated. If maximal commingling must be obtained, then the order in which the requisition lines are converted is important. When input argument iPreviousPurchaseOrderLine is filled, the function tries to add another sequence to that order line. Pre: Caller must set a retry-point Post: Caller must commit or abort the transaction. Input: - iPurchaseRequisition - Requisition; Mandatory - iRequisitionLine - Requisition Line; Mandatory - iOrderSeries - Order Series that will be used to generate the purchase order. Depending on the setup, this can be empty. - iOrderType - Order Type that will be used to generate the purchase order. Depending on the setup, this can be empty. - iSubcontractingOrderSeries - Order Series for subcontracting purchase orders. - iSubcontractingOrderType - Order Type for subcontracting purchase orders. - iServiceSubcontractingOrderSeries - Order Series for service subcontracting purchase orders. - iServiceSubcontractingOrderType - Order Type for service subcontracting purchase orders. - iCalculateNewPriceAndDiscounts - Yes: LN calculates new prices and discounts when converting requisitions to purchase orders. - No: The prices defined in the requisition are transferred to the purchase order. - iCurrencyFromBusinessPartner - Yes: LN uses the business partner's currency on the purchase order. The requisition's price and amount are converted to this currency.

	No: LN uses the currency of the requisition on the purchase order. iPreviousPurchaseOrder - If filled, the function tries to add a line to that order. Mandatory if iPreviousPurchaseOrderLine is filled. iPreviousPurchaseOrderLine - If filled, the function tries to add another sequence to that order line. Output: oGeneratedOrder - The generated purchase order oGeneratedOrderLine - The generated purchase order line oGeneratedOrderSequence - The generated purchase order line sequence Return: 0 - The requisition line is converted <> 0 - An error occurred

PurchaseRequisitionLine.ConvertToPurchaseRFQ

DLL:	tdextpurapi
Syntax:	<pre>long PurchaseRequisitionLine.ConvertToPurchaseRFQ(domain tcrqno iPurchaseRequisition, domain tcpono iRequisitionLine, domain tcyesno iAddToExistingRFQ, domain tcqono iExistingRFQ, domain tcseri iRFQSeries, domain tcrfq.type iRFQType, domain tcqono iPreviousRFQ, ref domain tcqono oGeneratedRFQ, ref domain tcpono oGeneratedRFQLine, ref domain tcpono oGeneratedRFQSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function converts the given requisition line to a purchase RFQ. Conversion is only allowed if the requisition header has the proper status (i.e. Approved or In Process), and the requisition line is not rejected. The Conversion Type of the requisition line must be 'RFQ'. Pre: Caller must set a retry-point Post: Caller must commit or abort the transaction. Input: - iPurchaseRequisition - Requisition; Mandatory - iRequisitionLine - Requisition Line; Mandatory - iAddToExistingRFQ - Yes: the requisition line is converted to an existing RFQ. Use iExistingRFQ to indicate to which RFQ the requisition line must be added. Notes: * this will override the value that is passed in iPreviousRFQ. * this option is only allowed if the concept of 'Enhanced Line Handling' is used for the given RFQ. No: this option allows a more regular form of commingling. Several requisition lines can be converted

	to one (or more) RFQ's, but only if the passed RFQ matches the data on the requisition line. See explanation at 'iPreviousRFQ'. iExistingRFQ - Indicates the RFQ to which the given requisition line must be converted. This field is mandatory if iAddToExistingRFQ is Yes. The field must be empty if iAddToExistingRFQ is No. iRFQSeries - The RFQ series that will be used when a new RFQ is generated. iRFQType - The RFQ Type that will be used when a new RFQ is generated. iPreviousRFQ - If filled, the function tries to add a requisition line to that RFQ. If they do not match, then a new RFQ is generated. Filling this field can be done when calling this function multiple times in order to convert multiple requisition lines to one (or: as little as possible) RFQs. If maximal commingling must be obtained, then the order in which the requisition lines are converted is important. Output: oGeneratedRFQ - The generated RFQ oGeneratedRFQLine - The generated RFQ line oGeneratedRFQSequence - The generated RFQ sequence Return: 0 - The requisition line is converted <> 0 - An error occurred

PurchaseRequisitionLine.StartDetail

DLL:	tdextpurapi
	This function is available from KB2285765 .
Syntax:	<pre>long PurchaseRequisitionLine.StartDetail(long iStartMode, domain tcrqno iPurchaseRequisition, domain tcpono iRequisitionLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the detail session Purchase Requisition Lines (tdpur2502m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iPurchaseRequisition Requisition; Mandatory iRequisitionLine Requisition Line; Mandatory

		Note that the given purchase requisition line must exist.
Output:		
	oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
	oExceptionID	An ID that refers to all error information. Use the functions in Exception to get all relevant information.
Return: 0		Session started
<> 0		An error occurred

Public Interfaces for PurchaseContract

The following functions are available:

[PurchaseContract.Activate](#)

[PurchaseContract.Deactivate](#)

[PurchaseContract.GetTotalAmounts](#)

[PurchaseContract.StartWorkbench](#)

[PurchaseContract.Terminate](#)

PurchaseContract.Activate

DLL:	tdextpurapi
Syntax:	<pre> long PurchaseContract.Activate(domain tccono iPurchaseContract, boolean iUpdateLineStatus, boolean iUpdateLogisticDataStatus, boolean iUpdatePriceStatus, boolean iUpdateTerminatedLines, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: Function can be used to activate a purchase contract. If ION Workflow Document Approval is used, then the following applies: * if the contract is waiting for Approval in ION, then updating the status is not allowed. * After executing this Public Interface, the status

	update must be approved in ION Workflow Document Approval. * Activation/Deactivation cannot be initiated for individual lines, but should be triggered on contract header level. If Change Requests are applicable for the given purchase contract, then the following applies: * Activation of the purchase contract can only be done once. No change request is required for that. After that, the change request procedure is required to perform changes Note: * This function does not trigger the generation of retro-billed price change advices. A separate Public Interface can be used to trigger that if necessary: 'Purchase.GenerateRetrobilledPriceChangeAdvicesForContracts' Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iPurchaseContract - Purchase Contract; Mandatory iUpdateLineStatus - True: will update the status of the contract lines as well. False: Contract lines will not be updated. iUpdateLogisticDataStatus - True: will update the status of the logistic data records as well. False: Logistic Data is not updated. iUpdatePriceStatus - True: will update the status of the contract price records as well. False: Price records are not updated. iUpdateTerminatedLines - True: will also update the status of terminated contract lines. False: Terminated contract lines are not updated. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Contract is activated <> 0 - An error occurred

PurchaseContract.Deactivate

DLL:	tdextpurapi
Syntax:	<pre>long PurchaseContract.Deactivate(domain tccono iPurchaseContract, boolean iUpdateLineStatus,</pre>

		boolean	iUpdateLogisticDataStatus,
		boolean	iUpdatePriceStatus,
		boolean	iUpdateTerminatedLines,
ref	domain	tcmcs.s999m	oExceptionMessage mb,
ref		long	oExceptionID)
Usage:	Expl.: Function can be used to deactivate a purchase contract. If ION Workflow Document Approval is used, then the following applies: * if the contract is waiting for Approval in ION, then updating the status is not allowed. * Activation/Deactivation cannot be initiated for individual lines, but should be triggered on contract header level. If Change Requests are applicable for the given purchase contract, then the following applies: * Deactivating a contract is not applicable. The contract will remain active. The change request procedure is required to perform changes. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iPurchaseContract - Purchase Contract; Mandatory iUpdateLineStatus - True: will update the status of the contract lines as well. False: Contract lines will not be updated. iUpdateLogisticDataStatus - True: will update the status of the logistic data records as well. False: Logistic Data is not updated. iUpdatePriceStatus - True: will update the status of the contract price records as well. False: Price records are not updated. iUpdateTerminatedLines - True: will also update the status of terminated contract lines. False: Terminated contract lines are not updated. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - The contract is deactivated <> 0 - An error occurred		

PurchaseContract.GetTotalAmounts

DLL:	tdextpurapi
Syntax:	<pre> long PurchaseContract.GetTotalAmounts (domain tccono iPurchaseContract, ref domain tcccur oContractCurrency, ref domain tcamnt oContractOpenAmount, ref domain tcamnt oContractTotalAmount, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: Function calculates total contract net amounts in contract currency. Contract is selected in the current company. Pre: N.A. Post: N.A. Input: iPurchaseContract - Purchase Contract; Mandatory. Output: oContractCurrency - Contract Currency. oContractOpenAmount - Contract Open Amount: Sum of amounts of all non terminated contract lines. oContractTotalAmount - Contract Total Amount: Sum of amounts of all contract lines. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Amount could be determined. <> 0 - Error occurred.

PurchaseContract.StartWorkbench

DLL:	tdextpurapi
Syntax:	<pre> long PurchaseContract.StartWorkbench (long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccwoc iPurchaseOffice, domain tcemno iBuyer, domain tccom.bpid iBuyFromBusinessPartner, domain tcsite iSite, domain tcitem iItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the workbench session Purchase Contracts (tdpur8330m000). Input: iStartMode Specifies the start mode for the session. Possible values are:

	MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend Not Used. Following input variables form the filtering fields, these fields are not mandatory. iPurchaseOffice Purchase Office iBuyer Buyer iBuyFromBusinessPartner Buy-from Business Partner iSite Site iItem Item Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - An error occurred

PurchaseContract.Terminate

DLL:	tdextpurapi
Syntax:	<pre>long PurchaseContract.Terminate(domain tccono iPurchaseContract, domain tcyesno iApproveAndProcessChangeRequestAutomatically, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl.: Function can be used to terminate a purchase contract. If ION Workflow Document Approval is used, then the following applies: * if the contract is waiting for Approval in ION, then updating the status is not allowed. * Termination does not require Workflow Document Approval If Change Requests are applicable for the given purchase contract, then the following applies: * A change request will be created for the status change. * Depending on the input argument 'iApproveAndProcessChangeRequestAutomatically', Approving and Processing of the Change Request will be done automatically or not. Pre: Caller must set retry-point Post: Caller must commit/abort transaction

Input: iPurchaseContract	- Purchase Contract or Contract Change Request; Mandatory.
iApproveAndProcessChangeRequestAutomatically	- Yes: The created change request will be approved and processed automatically. No: Approval and processing of the change request (if any) are not done automatically.
Output: oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
Return: 0	- The contract is terminated
<> 0	- An error occurred

Public Interfaces for PurchaseContractLine

The following functions are available:

[PurchaseContractLine.Activate](#)

[PurchaseContractLine.Deactivate](#)

[PurchaseContractLine.GetNetAmountOnPriceDate](#)

[PurchaseContractLine.Terminate](#)

PurchaseContractLine.Activate

DLL:	tdextpurapi
Syntax:	<pre>long PurchaseContractLine.Activate(domain tccono iPurchaseContract, domain tcpono iContractLine, domain tccwoc iContractPurchaseOffice, domain tcpono iContractSequence, boolean iUpdateLogisticDataStatus, boolean iUpdatePriceStatus, boolean iUpdateTerminatedLines, ref domain tcmcs.s999m oExceptionMessage mb,</pre>

	ref	long	oExceptionID)
Usage:	Expl.:	Function can be used to activate a purchase contract line. This function cannot be used if: * ION Workflow Document Approval is used. Status changes of the contract must be triggered on contract header level in that case. * Change Requests are applicable for the given purchase contract. Activation of a contract line is not applicable in that case. Note: * This function does not trigger the generation of retro-billed price change advices. A separate Public Interface can be used to trigger that if necessary: 'Purchase.GenerateRetrobilledPriceChangeAdvicesForContracts'	
	Pre:	Caller must set retry-point	
	Post:	Caller must commit/abort transaction	
	Input:	iPurchaseContract	- Purchase Contract; Mandatory
		iContractLine	- Purchase Contract Line; Mandatory
		iContractPurchaseOffice	- Purchase Office
		iContractSequence	- Contract Sequence (Must be 0. The status change will be applied to its contract detail lines as well.)
		iUpdateLogisticDataStatus	- True: will update the status of the logistic data records as well. False: Logistic Data is not updated.
		iUpdatePriceStatus	- True: will update the status of the contract price records as well. False: Price records are not updated.
		iUpdateTerminatedLines	- True: will also update the status of terminated contract lines. False: Terminated contract lines are not updated.
	Output:	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
		oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
	Return:	0	- Contract line is activated
		<> 0	- An error occurred

PurchaseContractLine.Deactivate

DLL:	tdextpurapi
Syntax:	<pre>long PurchaseContractLine.Deactivate(domain tccono iPurchaseContract, domain tcpono iContractLine,</pre>

	<pre> domain tccwoc iContractPurchaseOffice, domain tcpono iContractSequence, boolean iUpdateLogisticDataStatus, boolean iUpdatePriceStatus, boolean iUpdateTerminatedLines, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: Function can be used to deactivate a purchase contract line. This function cannot be used if: * ION Workflow Document Approval is used. Status changes of the contract must be triggered on contract header level in that case. * Change Requests are applicable for the given purchase contract. Deactivation of a contract line is not applicable in that case. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: - iPurchaseContract - Purchase Contract; Mandatory - iContractLine - Purchase Contract Line; Mandatory - iContractPurchaseOffice - Purchase Office - iContractSequence - Contract Sequence (Must be 0. The status change will be applied to its contract detail lines as well.) - iUpdateLogisticDataStatus - True: will update the status of the logistic data records as well. False: Logistic Data is not updated. - iUpdatePriceStatus - True: will update the status of the contract price records as well. False: Price records are not updated. - iUpdateTerminatedLines - True: will also update the status of terminated contract lines. False: Terminated contract lines are not updated. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Contract line is deactivated <> 0 - An error occurred

PurchaseContractLine.GetNetAmountOnPriceDate

DLL:	tdextpurapi
Syntax:	long PurchaseContractLine.GetNetAmountOnPriceDate(

	domain tccono iPurchaseContract, domain tcpono iContractLine, domain tccwoc iContractPurchaseOffice, domain tcpono iContractSequence, domain tcdate iPriceSearchDate, ref domain tcccur oContractCurrency, ref domain tcamnt oContractLineNetAmount, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl.: Function can be used to retrieve the contract line net amount, based on the price on the given date. Note that inactive price revisions are considered as well. Pre: NA Post: NA Input: iPurchaseContract - Purchase Contract (Mandatory) iContractLine - Contract Line (Mandatory) iContractPurchaseOffice - Purchase Office iContractSequence - Contract Sequence iPriceSearchDate - The date that is used to search the price revision. (Mandatory) Note that inactive price revisions are considered as well. Output: oContractCurrency oContractLineNetAmount - Net Amount, expressed in contract currency oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Function was executed successfull. <> 0 - An error occurred during the execution of the function.

PurchaseContractLine.Terminate

DLL:	tdextpurapi
Syntax:	long PurchaseContractLine.Terminate(domain tccono iPurchaseContract, domain tcpono iContractLine, domain tccwoc iContractPurchaseOffice, domain tcpono iContractSequence, domain tcyesno iApproveAndProcessChangeRequestAutomatically, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl.: Function can be used to terminate a purchase contract line. If ION Workflow Document Approval is used, then the following applies: * if the contract is waiting for Approval in ION, then updating the status is not allowed. * Termination does not require Workflow Document Approval If Change Requests are applicable for the given purchase

	contract, then the following applies: * A change request will be created for the status change. * Depending on the input argument 'iApproveAndProcessChangeRequestAutomatically', Approving and Processing of the Change Request will be done automatically or not. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iPurchaseContract - Purchase Contract or Contract Change Request; Mandatory. iContractLine - Purchase Contract Line; Mandatory iContractPurchaseOffice - Purchase Office iContractSequence - Contract Sequence (Must be 0. The status change will be applied to its contract detail lines as well.) iApproveAndProcessChangeRequestAutomatically - Yes: The created change request will be approved and processed automatically. - No: Approval and processing of the change request (if any) are not done automatically. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Contract line is terminated <> 0 - An error occurred

Public Interfaces for PurchaseContractLineLogisticData

The following functions are available:

[PurchaseContractLineLogisticData.Activate](#)

[PurchaseContractLineLogisticData.Deactivate](#)

PurchaseContractLineLogisticData.Activate

DLL:	tdextpurapi
Syntax:	<pre> long PurchaseContractLineLogisticData.Activate(domain tccono iPurchaseContract, domain tcpono iContractLine, domain tccwoc iContractPurchaseOffice, domain tcpono iContractSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: Function can be used to activate a purchase contract logistic data record. This function cannot be used if Change Requests are applicable for the given purchase contract. Activation of a contract logistic data record is not applicable in that case. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iPurchaseContract - Purchase Contract; Mandatory iContractLine - Purchase Contract Line; Mandatory iContractPurchaseOffice - Purchase Office iContractSequence - Contract Sequence Note: If the given sequence refers to a total-line, then also the related detail-lines will be updated with the new status. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Contract Logistic Data is activated <> 0 - An error occurred

PurchaseContractLineLogisticData.Deactivate

DLL:	tdextpurapi
Syntax:	<pre> long PurchaseContractLineLogisticData.Deactivate(domain tccono iPurchaseContract, domain tcpono iContractLine, domain tccwoc iContractPurchaseOffice, domain tcpono iContractSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: Function can be used to deactivate a purchase contract logistic data record. This function cannot be used if Change Requests are applicable for the given purchase contract. Deactivation of a contract logistic data record is not applicable in that case. Pre: Caller must set retry-point

	Post: Caller must commit/abort transaction Input: iPurchaseContract - Purchase Contract; Mandatory iContractLine - Purchase Contract Line; Mandatory iContractPurchaseOffice - Purchase Office iContractSequence - Contract Sequence Note: If the given sequence refers to a total-line, then also the related detail-lines will be updated with the new status. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Contract Logistic Data is deactivated <> 0 - An error occurred

Public Interfaces for PurchaseContractPrice

The following functions are available:

[PurchaseContractPrice.Activate](#)

[PurchaseContractPrice.Deactivate](#)

PurchaseContractPrice.Activate

DLL:	tdextpurapi
Syntax:	<pre>long PurchaseContractPrice.Activate(domain tccono iPurchaseContract, domain tcpono iContractLine, domain tccwoc iContractPurchaseOffice, domain tcpono iContractSequence, domain tcdate iEffectiveDate, domain tcyesno iApproveAndProcessChangeRequestAutomatically, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl.: Function can be used to activate a purchase contract price line. If ION Workflow Document Approval is used, then the following applies: * if the contract is waiting for Approval in ION, then updating the status is not allowed.

	* Activation/Deactivation of a price does not require Workflow Document Approval If Change Requests are applicable for the given purchase contract, then the following applies: * A change request will be created for the status change. * Depending on the input argument 'iApproveAndProcessChangeRequestAutomatically', Approving and Processing of the Change Request will be done automatically or not. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iPurchaseContract - Purchase Contract or Contract Change Request; Mandatory. iContractLine - Purchase Contract Line; Mandatory iContractPurchaseOffice - Purchase Office iContractSequence - Contract Sequence Note: If the given sequence refers to a total-line, then also the related detail-lines will be updated with the new status. iEffectiveDate - Effective Date; Mandatory iApproveAndProcessChangeRequestAutomatically - Yes: The created change request will be approved and processed automatically. No: Approval and processing of the change request (if any) are not done automatically. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Contract price line is activated <> 0 - An error occurred

PurchaseContractPrice.Deactivate

DLL:	tdextpurapi
Syntax:	<pre> long PurchaseContractPrice.Deactivate(domain tccono iPurchaseContract, domain tcpono iContractLine, domain tccwoc iContractPurchaseOffice, domain tcpono iContractSequence, domain tcdate iEffectiveDate, domain tcyesno iApproveAndProcessChangeRequestAutomatically, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: Function can be used to deactivate a purchase contract price line. If ION Workflow Document Approval is used, then the following

	<pre> applies: * if the contract is waiting for Approval in ION, then updating the status is not allowed. * Activation/Deactivation of a price does not require Workflow Document Approval If Change Requests are applicable for the given purchase contract, then the following applies: * A change request will be created for the status change. * Depending on the input argument 'iApproveAndProcessChangeRequestAutomatically', Approving and Processing of the Change Request will be done automatically or not. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iPurchaseContract - Purchase Contract or Contract Change Request; Mandatory. iContractLine - Purchase Contract Line; Mandatory iContractPurchaseOffice - Purchase Office iContractSequence - Contract Sequence Note: If the given sequence refers to a total-line, then also the related detail-lines will be updated with the new status. iEffectiveDate - Effective Date; Mandatory iApproveAndProcessChangeRequestAutomatically - Yes: The created change request will be approved and processed automatically. No: Approval and processing of the change request (if any) are not done automatically. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Contract price line is deactivated <> 0 - An error occurred </pre>

Public Interfaces for PurchaseSchedule

The following functions are available:

[PurchaseSchedule.GenerateReleaseForPushAndPullForecast](#)

[PurchaseSchedule.Regenerate](#)

[PurchaseSchedule.StartMultiMain](#)

[PurchaseSchedule.Terminate](#)

PurchaseSchedule.GenerateReleaseForPushAndPullForecast

DLL:	tdextpurapi
Syntax:	<pre> long PurchaseSchedule.GenerateReleaseForPushAndPullForecast (domain tcorno iPurchaseSchedule, domain tdstyp iPurchaseScheduleType, domain tcyesno iGenerateMaterialRelease, domain tcyesno iGenerateShippingSchedule, boolean iRebuildRelease, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function will generate the purchase releases for Push and Pull-Forecast schedules. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iPurchaseSchedule Purchase Schedule (Mandatory) iPurchaseScheduleType Schedule Type (Mandatory) Allowed values: Push Pull Forecast iGenerateMaterialRelease No: Release of type 'Material Release' is not generated. Yes: Release of type 'Material Release' will be generated if applicable. iGenerateShippingSchedule No: Release of type 'Shipping Schedule' is not generated. Yes: Release of type 'Shipping Schedule' will be generated if applicable. iRebuildRelease Rebuild the release false: If the latest release line for the given schedule has status 'Scheduled', the release line is not rebuilt. The function will return an error in that case. true: If the latest release line for the given schedule has status 'Scheduled', the release line will be rebuilt and put to status 'Created'. Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID

	oExceptionID Return: 0 <> 0	An ID that refers to all error information. Use the functions in Exception to get all relevant information. The purchase release(s) is/are generated. An error occurred

PurchaseSchedule.Regenerate

DLL:	tdextpurapi	
	This function is available from KB2308628 .	
Syntax:	<pre>long PurchaseSchedule.Regenerate(domain tcorno iPurchaseSchedule, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: This function regenerates a purchase schedule, like session »~~Re~~Regenerate Schedules»~~Ä~~ (tdpur3211m000). Regenerating a schedule redetermines the requirement type of the schedule lines. In addition, the price and discounts of schedule lines are redetermined. The regeneration process is applicable only for PUSH purchase schedules. Pre: Caller must set retry-point Post: Caller must set commit/abort transaction Input: iPurchaseSchedule - Purchase Schedule (mandatory) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 -> No error <> 0 -> Error occurred	

PurchaseSchedule.StartMultiMain

DLL:	tdextpurapi	
	This function is available from KB3632431 .	
Syntax:	<pre>long PurchaseSchedule.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcorno iPurchaseSchedule, domain tdstyp iScheduleType, ref domain tcmcs.s999m oExceptionMessage mb,</pre>	

	ref	long	oExceptionID)
Usage:	Expl:	This function starts the Multi-Main session Purchase Schedule (tdpur3610m000).	
	Input:	iStartMode	Specifies the start mode for the session (mandatory).
		Possible values are:	
		MODAL -	The parent session is blocked until the child session exits, the session will be started as a zoom session.
		MODELESS -	Parent and child are parallel sessions that can be manipulated simultaneously.
		iStartFilter	Not used
		iQueryExtend	A specific query to be used when zooming to this session (optional).
		iPurchaseSchedule	Purchase schedule (mandatory).
		iScheduleType	Schedule Type (mandatory)
			Values are: Push, Pull Forecast and Pull Call-off.
	Output:	oExceptionMessage	The last message if any message is found. If more than one message is given, these are present in the oExceptionID.
		oExceptionID	An ID that refers to the exception information. Use the functions in Exception to get all relevant information.
	Return:	0	Session started
		<> 0	An error occurred

PurchaseSchedule.Terminate

DLL:	tdextpurapi
Syntax:	<pre> long PurchaseSchedule.Terminate(domain tcorno iPurchaseSchedule, domain tcyesno iCopySchedule, domain tcyesno iCancelOpenScheduleLines, domain tdpur.stat iCancelUptoStatus, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function terminates the given purchase schedule. For Pull Call-off schedules, the purchase release is generated or updated for each cancelled schedule line. For Push and Pull Forecast schedules, Public Interface 'PurchaseSchedule.GenerateReleaseForPushAndPullForecast' must be called after termination of the purchase schedule. Purchase Schedules that are related to Direct Delivery cannot be terminated using this Public Interface. They must be terminated from the related Sales Schedule. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iPurchaseSchedule Purchase Schedule (Mandatory) iCopySchedule No: No new purchase schedule will be created. Yes: Copies the open schedule lines to a

	new schedule. This applies to Push and Pull-Forecast only. Lines having the following status are copied: - tdpur.stat.created - tdpur.stat.approved - tdpur.stat.order.generated iCancelOpenScheduleLines No: open lines will not be cancelled Yes: open schedule lines that have the proper status will be cancelled. Note that schedule lines that are generated by Assembly Control are never cancelled by this function. iCancelUptoStatus Schedule lines with a status less or equal to this value will be cancelled. If iCancelOpenScheduleLines equals 'No' then this input is ignored. Allowed values: - tdpur.stat.created - tdpur.stat.approved - tdpur.stat.disapproved - tdpur.stat.order.generated - tdpur.stat.asn.received Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 The purchase schedule is terminated An error occurred

Public Interfaces for PurchaseOrder

The following functions are available:

[PurchaseOrder.InitiateChangeRequest](#)

[PurchaseOrder.PrintPurchaseOrder](#)

[PurchaseOrder.RecalculateLandedCosts](#)

[PurchaseOrder.StartAutomaticProcessing](#)

[PurchaseOrder.StartIntakeWorkbench](#)

[PurchaseOrder.StartMultiMain](#)

[PurchaseOrder.StartPrintReminders](#)

[PurchaseOrders.StartOverview](#)

PurchaseOrder.InitiateChangeRequest

DLL:	tdextpurapi
	This function is available from KB2323365 .
Syntax:	<pre> long PurchaseOrder.InitiateChangeRequest(domain tcorno iPurchaseOrder, domain tdpur.corg iChangeRequestOrigin, ref domain tcorno oChangeRequest, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function initiates a new change request for the given purchase order. If the status of the given purchase order is 'Closed', then it will first be re-opened. The initiation consists of two parts: 1. Determine if initiate change request is allowed. 2. Initiate the change request. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iPurchaseOrder - Purchase Order (mandatory) iChangeRequestOrigin - Origin of the Change Request (mandatory) Output: oChangeRequest - New Change Request oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 -> The change request has been initiated <> 0 -> An error occurred

PurchaseOrder.PrintPurchaseOrder

DLL:	tdextpurapi
	This function is available from KB3554192 .
Syntax:	<pre> long PurchaseOrder.PrintPurchaseOrder(domain tcorno iPurchaseOrder, long iProcessingOptionSet, ref boolean oOrderPrinted, ref boolean oProcessStopped, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>

Usage:

Expl: This function handles the printing of the Purchase Order for the given Purchase Order.

Closing of reports:

During the printing of the purchase order it is possible that one or more reports will be opened. Closing of these reports is also done within this function.

Transaction handling:

Retry-point and commit/abort transaction is handled within this function.

Note:

This function does not start the execution of automatic order steps. A separate Public Interface can be used to start automatic order steps if necessary:

'PurchaseOrder.StartAutomaticProcessing'

Pre: -

Post: -

Input: iPurchaseOrder - Purchase Order (mandatory)

iProcessingOptionSet - Processing Option Set (Optional).

If 0, the default printing options are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcextxtapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete()

- If Processing Option "ReportType" is '1' (Purchase Order Acknowledgement) the other Processing Options have a direct relationship with the form fields on session "Print Purchase Orders" (tdpur4401m000) and are not explained in further detail here. Please refer to the session help for additional information. Print options which are not available as Processing Options will get defaulted in accordance with the session logic. Processing Options which are set while a required Implemented Software Component is not available are ignored.

- If Processing Option "ReportType" is '2' (Purchase Order Preview) the other Processing Options are ignored. Printing is based on session "Preview Purchase Orders" (tdpur4401m200).

NAME	TYPE	DEFAULT
ReportType	long	1
Draft	boolean	false
OrderLineTypesToPrint	domain tdpur.kofl	
tdpur.kofl.exc.printed		
OnlyNewCanceledLines	domain tcyesno	tcyesno.yes
ExcludeLinesWithChangeCodes	domain tcyesno	tcyesno.no
LinesToPrint	domain tdsls.koqu	
tdsls.koqu.ordered.quant		
QuantitiesPrintedIn	domain tdgen.utop	
tdgen.utop.order.unit		
PrintVariantOptions	domain tcyesno	tcyesno.yes
PrintTaxText	domain tcyesno	tcyesno.yes
PrintPrices	domain tcyesno	tcyesno.yes
PrintCustomsValue	domain tcyesno	tcyesno.no
PrintDeliveryAddressPerLine	domain tcyesno	tcyesno.no
PrintManufacturerPartNumbers	domain tcyesno	tcyesno.no
PrintLandedCosts	domain tcyesno	tcyesno.no
PrintMaterialPriceInformation	domain tcyesno	tcyesno.no
PrintAppendixMaterialSupplyInformation	domain tcyesno	tcyesno.yes
PrintAppendixLandedCosts	domain tcyesno	tcyesno.yes
PrintAppendixConformanceReporting	domain tcyesno	tcyesno.yes

	<pre> PrintAppendixAdditionalInformation domain tyesno tyesno.yes PrintAppendixEndUserDeclarations domain tyesno tyesno.yes PrintAppendixLicenses domain tyesno tyesno.yes PrintAppendixLettersOfCredit domain tyesno tyesno.no PrintAppendixBankGuaranteeApplicant domain tyesno tyesno.yes PrintAppendixBankGuaranteeBeneficiary domain tyesno tyesno.yes PrintToPredefinedDevice domain tyesno tyesno.yes PrintingDevice domain tcmcs.str14 "" PrintingFileoutPathAndName domain tcmcs.str100 "" </pre> - PrintingDevice: This field can be filled with the Device to were the reports are printed. If not provided a dialog will be shown to enter the Device or when PrintToPredefinedDevice is set to "Yes" and defaults are present then printing will be done to the default printing device(s). - PrintingFileoutPathAndName: Depending on the device this field can be be filled (if needed) with the output path and filename for storing the file of the printed report. Output: oOrderPrinted - True: the purchase order has been printed. False: the purchase order is not printed. oProcessStopped - True: the printing process is stopped by the user or due to an error. False: the printing process was completed. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Possible that order has been printed <> 0 - An error occurred

PurchaseOrder.RecalculateLandedCosts

DLL:	tdextpurapi
	This function is available from KB3634332 .
Syntax:	<pre> long PurchaseOrder.RecalculateLandedCosts (domain tcorno iPurchaseOrder, domain tcpono iPurchaseOrderLine, domain tcpono iPurchaseOrderSequence, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function handles the recalculation of Landed Costs for a Purchase Order or for a Purchase Order Line. - Recalculation of Landed Cost for a Purchase Order: If i.purchase.order.line IS NOT PROVIDED (i.e., equals 0), the landed cost recalculation will be performed at the Purchase Order header level for the specified Purchase Order. In this situation the optional input arguments OrderHeaderLandedCosts and OrderLinesLandedCosts, if

	provided, can be set to value 'tctyesno.yes' or 'tctyesno.no'.																																	
	- Recalculation of Landed Cost for a Purchase Order Line: If i.purchase.order.line IS PROVIDED, the recalculation will be executed at the Purchase Order Line level for the specified Purchase Order Line. In this situation the optional input arguments OrderHeaderLandedCosts and OrderLinesLandedCosts, if provided, must be set to value 'tctyesno.no'.																																	
	Supported Processing Options and their defaults:																																	
	<table><tr><td>NAME</td><td>TYPE</td><td>DEFAULT</td></tr><tr><td>OrderHeaderLandedCosts</td><td>domain tctyesno</td><td>If</td></tr><tr><td>i.purchase.order.line = 0 the</td><td></td><td>default is set to</td></tr><tr><td>'tctyesno.yes',</td><td></td><td>otherwise to</td></tr><tr><td>'tctyesno.no'</td><td></td><td></td></tr><tr><td>OrderLinesLandedCosts</td><td>domain tctyesno</td><td>If</td></tr><tr><td>i.purchase.order.line = 0 the</td><td></td><td>default is set to</td></tr><tr><td>'tctyesno.yes',</td><td></td><td>otherwise to</td></tr><tr><td>'tctyesno.no'</td><td></td><td></td></tr><tr><td>OverwriteManualAndModifiedLandedCosts</td><td>domain tctyesno</td><td>The default is set</td></tr><tr><td>to 'tctyesno.no'</td><td></td><td></td></tr></table>	NAME	TYPE	DEFAULT	OrderHeaderLandedCosts	domain tctyesno	If	i.purchase.order.line = 0 the		default is set to	'tctyesno.yes',		otherwise to	'tctyesno.no'			OrderLinesLandedCosts	domain tctyesno	If	i.purchase.order.line = 0 the		default is set to	'tctyesno.yes',		otherwise to	'tctyesno.no'			OverwriteManualAndModifiedLandedCosts	domain tctyesno	The default is set	to 'tctyesno.no'		
NAME	TYPE	DEFAULT																																
OrderHeaderLandedCosts	domain tctyesno	If																																
i.purchase.order.line = 0 the		default is set to																																
'tctyesno.yes',		otherwise to																																
'tctyesno.no'																																		
OrderLinesLandedCosts	domain tctyesno	If																																
i.purchase.order.line = 0 the		default is set to																																
'tctyesno.yes',		otherwise to																																
'tctyesno.no'																																		
OverwriteManualAndModifiedLandedCosts	domain tctyesno	The default is set																																
to 'tctyesno.no'																																		
	Supported Processing Options and their meaning:																																	
	OrderHeaderLandedCosts - This option is only considered in case i.purchase.order.line IS NOT PROVIDED (meaning; acting on header level). If it is set to tctyesno.yes, the system will execute a recalculation of all Landed Cost Lines related to the specified Purchase Order header. Otherwise this is not done.																																	
	OrderLinesLandedCosts - This option is only considered in case i.purchase.order.line IS NOT PROVIDED (meaning; acting on header level). If it is set to tctyesno.yes, the system will execute a recalculation of all Landed Cost Lines related to all Purchase Order Lines of the specified Purchase Order header. Otherwise this is not done.																																	
	OverwriteManualAndModifiedLandedCosts - This option indicates whether manual and modified Landed Costs are allowed to be overwritten during recalculation.																																	

	Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iPurchaseOrder - Purchase Order (Mandatory) iPurchaseOrderLine - Purchase Order Line (Optional) iPurchaseOrderSequence - Purchase Sequence number (Optional) iProcessingOptionSet - Processing Option Set (Optional). If 0, the default recalculate options are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcontextapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete() Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Recalculation was successful, or no recalculation needs to be done. <> 0 An error occurred

PurchaseOrder.StartAutomaticProcessing

DLL:	tdextpurapi
Syntax:	<pre>long PurchaseOrder.StartAutomaticProcessing(domain tcorno iPurchaseOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts processing the next purchase order activities that are set to execute automatically. This function must not be called within a logical transaction as this function will have its own transaction handling. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iPurchaseOrder - Purchase Order (mandatory) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Automatic Processing finished or next activities are not automatic. <> 0 - An error occurred

PurchaseOrder.StartIntakeWorkbench

DLL:	tdextpurapi
Syntax:	<pre> long PurchaseOrder.StartIntakeWorkbench(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccwoc iPurchaseOffice, domain tcemno iBuyer, domain tccom.bpid iBuyFromBusinessPartner, domain tcsite iSite, domain tcitem iItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Purchase Order Intake Workbench (tdpur460lm200). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend Not Used. Following input variables form the filtering fields, these fields are not mandatory. iPurchaseOffice Purchase Office iBuyer Buyer iBuyFromBusinessPartner Buy-from Business Partner iSite Site iItem Item Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - An error occurred

PurchaseOrder.StartMultiMain

DLL:	tdextpurapi
Syntax:	<pre> long PurchaseOrder.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcorno iPurchaseOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the Multi-Main session Purchase Order (tdpur4100m900). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iQueryExtend A specific query to be used when zooming to this session. iPurchaseOrder Purchase Order Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

PurchaseOrder.StartPrintReminders

DLL:	tdextpurapi
	This function is available from KB3643777 .
Syntax:	<pre> long PurchaseOrder.StartPrintReminders(long iStartMode, boolean iIgnoreSelectionFields, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function starts the session Print Purchase Order Reminders (tdpur4403m000). Depending on the main table of the calling session, Non-Consecutive Record Selection (NCRS) is applied. Specifically, when the main table is either:

	* Purchase Orders (tdpur400), or * Purchase Order Lines (tdpur401), the session processes the selected records accordingly. If the parameter iIgnoreSelectionFields is set to false, any ranges provided in the option set are applied as additional filters to further refine the selected records. Pre: N/A Post: N/A Input: iStartMode - Not used iIgnoreSelectionFields - If true, the session is started with the session-defaults, instead of filling the fields through this Public Interface. iProcessingOptionSet - Processing Option Set (optional). If 0, the default options are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcextxtapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete() Processing Options have a direct relationship with the form fields on session Print Purchase Order Reminders (tdpur4403m000)) and are not explained in further detail here. Please refer to the session help for additional information. Processing Options that are set while a required Implemented Software Component is not available are ignored. Explanation about setting of defaults: Minimum Value: Minimum value of domain is taken as default value. Maximum Value: Maximum value of domain is taken as default value. Supported Processing Options and their defaults: <table><tr><th>NAME</th><th>TYPE</th><th>DEFAULT</th></tr><tr><td>BuyFromBPFrom</td><td>domain tccom.bpid</td><td>Minimum Value</td></tr><tr><td>BuyFromBPTo</td><td>domain tccom.bpid</td><td>Maximum Value</td></tr><tr><td>ShipFromBPFrom</td><td>domain tccom.bpid</td><td>Minimum Value</td></tr><tr><td>ShipFromBPTo</td><td>domain tccom.bpid</td><td>Maximum Value</td></tr><tr><td>PurchaseOfficeFrom</td><td>domain tccwoc</td><td>Minimum Value</td></tr><tr><td>PurchaseOfficeTo</td><td>domain tccwoc</td><td>Maximum Value</td></tr><tr><td>BuyerFrom</td><td>domain tcemno</td><td>Minimum Value</td></tr><tr><td>BuyerTo</td><td>domain tcemno</td><td>Maximum Value</td></tr><tr><td>PlannerFrom</td><td>domain tcemno</td><td>Minimum Value</td></tr><tr><td>PlannerTo</td><td>domain tcemno</td><td>Maximum Value</td></tr><tr><td>PurchaseOrderFrom</td><td>domain tcorno</td><td>Minimum Value</td></tr><tr><td>PurchaseOrderTo</td><td>domain tcorno</td><td>Maximum Value</td></tr><tr><td>PurchaseOrderLineFrom</td><td>domain tcpono</td><td>Minimum Value</td></tr><tr><td>PurchaseOrderLineTo</td><td>domain tcpono</td><td>Maximum Value</td></tr><tr><td>ItemFrom</td><td>domain tcitem</td><td>Minimum Value</td></tr><tr><td>ItemTo</td><td>domain tcitem</td><td>Maximum Value</td></tr><tr><td>ReferenceDate</td><td>domain tcdate</td><td>Current Date/Time</td></tr><tr><td>DateOnReminder</td><td>domain tcdate</td><td>Current Date/Time</td></tr><tr><td>AddressTo</td><td>domain tccom.bprp</td><td>tccom.bprp.buy.from</td></tr><tr><td>SortedBy</td><td>domain tdpur.orso</td><td>tdpur.orso.otbp</td></tr><tr><td>PrintRemindersByBuyFromBP</td><td>domain tcyesno</td><td>tcyesno.yes</td></tr><tr><td>IgnoreOrderLinesNotYetPrinted</td><td>domain tcyesno</td><td>tcyesno.no</td></tr><tr><td>PrintLinesWithUnconfirmedReceipt</td><td>domain tcyesno</td><td>tcyesno.yes</td></tr><tr><td>PrintToPredefinedDevice</td><td>domain tcyesno</td><td>tcyesno.yes</td></tr></table>			NAME	TYPE	DEFAULT	BuyFromBPFrom	domain tccom.bpid	Minimum Value	BuyFromBPTo	domain tccom.bpid	Maximum Value	ShipFromBPFrom	domain tccom.bpid	Minimum Value	ShipFromBPTo	domain tccom.bpid	Maximum Value	PurchaseOfficeFrom	domain tccwoc	Minimum Value	PurchaseOfficeTo	domain tccwoc	Maximum Value	BuyerFrom	domain tcemno	Minimum Value	BuyerTo	domain tcemno	Maximum Value	PlannerFrom	domain tcemno	Minimum Value	PlannerTo	domain tcemno	Maximum Value	PurchaseOrderFrom	domain tcorno	Minimum Value	PurchaseOrderTo	domain tcorno	Maximum Value	PurchaseOrderLineFrom	domain tcpono	Minimum Value	PurchaseOrderLineTo	domain tcpono	Maximum Value	ItemFrom	domain tcitem	Minimum Value	ItemTo	domain tcitem	Maximum Value	ReferenceDate	domain tcdate	Current Date/Time	DateOnReminder	domain tcdate	Current Date/Time	AddressTo	domain tccom.bprp	tccom.bprp.buy.from	SortedBy	domain tdpur.orso	tdpur.orso.otbp	PrintRemindersByBuyFromBP	domain tcyesno	tcyesno.yes	IgnoreOrderLinesNotYetPrinted	domain tcyesno	tcyesno.no	PrintLinesWithUnconfirmedReceipt	domain tcyesno	tcyesno.yes	PrintToPredefinedDevice	domain tcyesno	tcyesno.yes
NAME	TYPE	DEFAULT																																																																												
BuyFromBPFrom	domain tccom.bpid	Minimum Value																																																																												
BuyFromBPTo	domain tccom.bpid	Maximum Value																																																																												
ShipFromBPFrom	domain tccom.bpid	Minimum Value																																																																												
ShipFromBPTo	domain tccom.bpid	Maximum Value																																																																												
PurchaseOfficeFrom	domain tccwoc	Minimum Value																																																																												
PurchaseOfficeTo	domain tccwoc	Maximum Value																																																																												
BuyerFrom	domain tcemno	Minimum Value																																																																												
BuyerTo	domain tcemno	Maximum Value																																																																												
PlannerFrom	domain tcemno	Minimum Value																																																																												
PlannerTo	domain tcemno	Maximum Value																																																																												
PurchaseOrderFrom	domain tcorno	Minimum Value																																																																												
PurchaseOrderTo	domain tcorno	Maximum Value																																																																												
PurchaseOrderLineFrom	domain tcpono	Minimum Value																																																																												
PurchaseOrderLineTo	domain tcpono	Maximum Value																																																																												
ItemFrom	domain tcitem	Minimum Value																																																																												
ItemTo	domain tcitem	Maximum Value																																																																												
ReferenceDate	domain tcdate	Current Date/Time																																																																												
DateOnReminder	domain tcdate	Current Date/Time																																																																												
AddressTo	domain tccom.bprp	tccom.bprp.buy.from																																																																												
SortedBy	domain tdpur.orso	tdpur.orso.otbp																																																																												
PrintRemindersByBuyFromBP	domain tcyesno	tcyesno.yes																																																																												
IgnoreOrderLinesNotYetPrinted	domain tcyesno	tcyesno.no																																																																												
PrintLinesWithUnconfirmedReceipt	domain tcyesno	tcyesno.yes																																																																												
PrintToPredefinedDevice	domain tcyesno	tcyesno.yes																																																																												

	Output: oExceptionMessage oExceptionID Return: 0 <> 0	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information - Session started - An error occurred

PurchaseOrders.StartOverview

DLL:	tdextpurapi	
	This function is available from KB3501666 .	
Syntax:	<pre>long PurchaseOrders.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iOrder, domain tccom.bpid iBuyFromBusinessPartner, domain tcrefc iBuyFromBusinessPartnerOrder mb, domain tccwoc iPurchaseOffice, domain tcemno iBuyer, ref domain tcorno oOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: This function starts the Overview session Purchase Orders (tdpur4100m000). Input: iStartMode Specifies the start mode for the session (Mandatory). Possible values are: MODAL The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Start filter (Not used). iSessionIndex Session index Possible values are: 1 Purchase Order (default) 2 Buy-from Business Partner 3 Buy-from Business Partner Order 4 Buyer 5 Purchase Office iQueryExtend A specific query to be used when zooming to this session (Optional). iOrder Purchase Order (Optional). iBuyFromBusinessPartner Buy-from Business Partner (Mandatory if iStartMode is MODELESS and iSessionIndex is 2). iBuyFromBusinessPartnerOrder Buy-from Business Partner Order (Optional).	

	<pre> iPurchaseOffice iBuyer Output: For iStartMode MODAL: oOrder oExceptionMessage oExceptionID Return: 0 <> 0 </pre>	Purchase Office (Mandatory if iStartMode is MODELESS and iSessionIndex is 4). Buyer (Optional). Selected Order number The last message if any message is found. If more than one message is given, these are present in the oExceptionID. An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Session started An error occurred

Public Interfaces for PurchaseOrderLine

The following functions are available:

[PurchaseOrderLine.Approve](#)

[PurchaseOrderLine.CalculatePlannedReceiptDate](#)

[PurchaseOrderLine.Cancel](#)

[PurchaseOrderLine.ConfirmPotentialBackorder](#)

[PurchaseOrderLine.GetCurrentActivity](#)

[PurchaseOrderLine.ReleaseToWarehousing](#)

[PurchaseOrderLine.Split](#)

[PurchaseOrderLines.StartOverview](#)

PurchaseOrderLine.Approve

DLL:	tdextpurapi
Syntax:	<pre> long PurchaseOrderLine.Approve(domain tcorno iPurchaseOrder, domain tcpono iOrderLine, domain tcpono iSequence, ref domain tcmcs.s999m oExceptionMessage mb, </pre>

	ref	long	oExceptionID)
Usage:	Expl.:	This function can be used to approve the given purchase order line. Note that approval is not allowed on line level for the scenarios below: * If Workflow is active for purchase orders * If Change Requests are required Note: This function does not start the execution of automatic order steps. A separate Public Interface can be used to start automatic order steps if necessary: 'PurchaseOrder.StartAutomaticProcessing' Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iPurchaseOrder Purchase Order (Mandatory) iOrderLine Purchase Order Line (Mandatory) iSequence Purchase Order Line Sequence. (must be >= 0). If 0 is passed, then this will approve all sequences for the given order line that require approval. Otherwise, only the given sequence is approved. Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 The order line is approved <> 0 An error occurred	

PurchaseOrderLine.CalculatePlannedReceiptDate

DLL:	tdextpurapi
Syntax:	<pre> long PurchaseOrderLine.CalculatePlannedReceiptDate(domain tcorno iOrderNumber, domain tcdater iOrderDate, domain tcdater iLoadDate, domain tccom.bpid iBuyFromBusinessPartner, domain tccom.bpid iShipFromBusinessPartner, domain tcsite iSite, domain tccwar iWarehouse, domain tccwoc iPurchaseOffice, domain tccfrw iCarrier, domain tccom.cadr iShipFromAddress, domain tccom.cadr iReceiptAddress, domain tcitem iItem, domain tccrte iRoute, domain tcmcs.serv iFreightLevel, domain tccdec iDeliveryTerms, domain tcptpa iPointOfTitlePassage, </pre>

Public Interfaces for Purchase

	<pre> ref domain tcdatetime oPlannedReceiptDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>																																
Usage:	Expl: This function calculates the planned receipt date. If the planned load date is filled, then the function uses the planned load date and the transportation time to calculate the planned receipt date. In case no planned load date is available then the order date will be used, and the Item Supplier lead times will be used to calculate the planned receipt date. The calculated receipt date is shifted forward to the availability of the warehouse address calendar (calculated date must be a workable date in the warehouse calendar). Pre: Not Applicable Post: Not Applicable Input: <table> <tr> <td>iOrderNumber</td><td>Purchase Order (Mandatory)</td></tr> <tr> <td>iOrderDate</td><td>Order Date (Mandatory)</td></tr> <tr> <td>iLoadDate</td><td>Planned Load Date</td></tr> <tr> <td>iBuyFromBusinessPartner</td><td>Buy from Business Partner (Mandatory)</td></tr> <tr> <td>iShipFromBusinessPartner</td><td>Ship from Business Partner (Mandatory)</td></tr> <tr> <td>iSite</td><td>Site</td></tr> <tr> <td>iWarehouse</td><td>Warehouse</td></tr> <tr> <td>iPurchaseOffice</td><td>Purchase Office (Mandatory)</td></tr> <tr> <td>iCarrier</td><td>Carrier</td></tr> <tr> <td>iShipFromAddress</td><td>Ship from address (Mandatory)</td></tr> <tr> <td>iReceiptAddress</td><td>Receipt address (Mandatory if iWarehouse is empty)</td></tr> <tr> <td>iItem</td><td>Item (Mandatory)</td></tr> <tr> <td>iRoute</td><td>Route</td></tr> <tr> <td>iFreightLevel</td><td>Freight Level</td></tr> <tr> <td>iDeliveryTerms</td><td>Delivery Terms</td></tr> <tr> <td>iPointOfTitlePassage</td><td>Point of Title Passage</td></tr> </table> Output: oPlannedReceiptDate Planned Receipt Date Return: 0 Planned Receipt Date is calculated successfully <> 0 An error occurred.	iOrderNumber	Purchase Order (Mandatory)	iOrderDate	Order Date (Mandatory)	iLoadDate	Planned Load Date	iBuyFromBusinessPartner	Buy from Business Partner (Mandatory)	iShipFromBusinessPartner	Ship from Business Partner (Mandatory)	iSite	Site	iWarehouse	Warehouse	iPurchaseOffice	Purchase Office (Mandatory)	iCarrier	Carrier	iShipFromAddress	Ship from address (Mandatory)	iReceiptAddress	Receipt address (Mandatory if iWarehouse is empty)	iItem	Item (Mandatory)	iRoute	Route	iFreightLevel	Freight Level	iDeliveryTerms	Delivery Terms	iPointOfTitlePassage	Point of Title Passage
iOrderNumber	Purchase Order (Mandatory)																																
iOrderDate	Order Date (Mandatory)																																
iLoadDate	Planned Load Date																																
iBuyFromBusinessPartner	Buy from Business Partner (Mandatory)																																
iShipFromBusinessPartner	Ship from Business Partner (Mandatory)																																
iSite	Site																																
iWarehouse	Warehouse																																
iPurchaseOffice	Purchase Office (Mandatory)																																
iCarrier	Carrier																																
iShipFromAddress	Ship from address (Mandatory)																																
iReceiptAddress	Receipt address (Mandatory if iWarehouse is empty)																																
iItem	Item (Mandatory)																																
iRoute	Route																																
iFreightLevel	Freight Level																																
iDeliveryTerms	Delivery Terms																																
iPointOfTitlePassage	Point of Title Passage																																

PurchaseOrderLine.Cancel

DLL:	tdextpurapi
Syntax:	<pre> long PurchaseOrderLine.Cancel(domain tcorno iPurchaseOrder, domain tcpono iOrderLine, domain tcpono iOrderLineSequence, domain tdpur.corg iOriginForChangeRequest, domain tcmcs.str2 iOrderHeaderAcknowledgementCode, domain tcmcs.str8 iOrderHeaderChangeOrderSequence, domain tccdis iOrderHeaderChangeReason, domain tccdis iOrderChangeType, domain tcmcs.str2 iOrderLineAcknowledgementCode, domain tcmcs.str8 iOrderLineChangeOrderSequence, domain tccdis iOrderLineChangeReason, domain tccdis iOrderLineChangeType, </pre>

	<pre> domain tcmcs.str2 iSalesOrderLineAcknowledgementCode, domain tcmcs.str8 iSalesOrderLineChangeOrderSequence, domain tccdis iSalesOrderLineChangeReason, domain tccdis iSalesOrderLineChangeType, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function can be used to cancel the given purchase order line(s). It can be used to cancel all order lines, or only a subset. If there are subcontracting supply orders linked to an order line, these supply orders are also cancelled or deleted. Note: This function does not start the execution of automatic order steps. A separate Public Interface can be used to start automatic order steps if necessary: 'PurchaseOrder.StartAutomaticProcessing' In some cases, this may be needed: - If a production order is linked to the cancelled (back) order line, then there can still be inventory waiting to be transferred from the warehouse to the shopfloor. When the backorder is cancelled this transfer order still needs to get an update. - For order lines that are invoiced by stage payments it is possible that variance transactions are written after cancellation of an order line (last open detail/backorder line). In that case the variances must be processed automatically after the commit of the transaction, which can be achieved by calling Public Interface 'PurchaseOrder.StartAutomaticProcessing'. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: <pre> iPurchaseOrder Purchase Order (Mandatory) iOrderLine Purchase Order Line (Optional) If 0 is passed, then this will cancel all order lines for the given order. Otherwise, only the given order line is cancelled. iOrderLineSequence Purchase Order Line Sequence (Optional) If 0 is passed, then this will cancel all sequences for the given order line. Otherwise, only the given sequence is cancelled. iOriginForChangeRequest Origin for Change Request (Mandatory) If modifications to the given purchase order require a change request to be created, then the given origin will be used to create it. iOrderHeaderAcknowledgementCode Acknowledgement code purchase order header iOrderHeaderChangeOrderSequence Change sequence purchase order header iOrderHeaderChangeReason (Mandatory if 'Change Codes Mandatory' is set to Yes in the Purchase Order Parameters or Purchase Order Settings by Office). Change reason purchase order header iOrderChangeType (Mandatory if 'Change Codes Mandatory' is set to Yes in the Purchase Order </pre>

	Parameters or Purchase Order Settings by Office). Change type purchase order header iOrderLineAcknowledgementCode (Optional) Acknowledgement code purchase order line iOrderLineChangeOrderSequence (Optional) Change sequence purchase order line iOrderLineChangeReason (Mandatory if 'Change Codes Mandatory' is set to Yes in the Purchase Order Parameters or Purchase Order Settings by Office). Change reason purchase order line iOrderLineChangeType (Mandatory if 'Change Codes Mandatory' is set to Yes in the Purchase Order Parameters or Purchase Order Settings by Office). Change type purchase order line iSalesOrderLineAcknowledgementCode (Optional) Acknowledgement code sales order line. If there are subcontracting supply orders linked to an order line, then these are cancelled as well. If the supply order is a sales order line, then this acknowledgement code will be used to cancel it. iSalesOrderLineChangeOrderSequence (Optional) Change sequence sales order line. If there are subcontracting supply orders linked to an order line, then these are cancelled as well. If the supply order is a sales order line, then this change order sequence will be used to cancel it. iSalesOrderLineChangeReason (Mandatory if 'Change Codes Mandatory' is set to Yes in the Sales Order Parameters or Sales Order Settings by Office). Change reason sales order line. If there are subcontracting supply orders linked to an order line, then these are cancelled as well. If the supply order is a sales order line, then this change reason will be used to cancel it. iSalesOrderLineChangeType (Mandatory if 'Change Codes Mandatory' is set to Yes in the Sales Order Parameters or Sales Order Settings by Office). Change type sales order line. If there are subcontracting supply orders linked to an order line, then these are cancelled as well. If the supply order is a sales order line, then this change type will be used to cancel it. Output: oExceptionMessage The last message. This can also be filled if the return value equals 0. If more than one message is given,

	oExceptionID Return: 0 <> 0	these are present in the oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. The order line(s) is/are cancelled. An error occurred

PurchaseOrderLine.ConfirmPotentialBackorder

DLL:	tdextpurapi	
	This function is available from KB3546270 .	
Syntax:	long PurchaseOrderLine.ConfirmPotentialBackorder(domain tcorno iPurchaseOrder, domain tcpono iPurchaseOrderLine, domain tcpono iPurchaseOrderLineSequence, domain tcqrdl iBackorderQuantityToBeConfirmed, domain tccuni iBackorderUnit, domain tcdate iBackorderPlannedReceiptDate, domain tctxtn iNotesTextNumber, domain tccdis iPegChangeReason, domain tctyesno iApproveAndProcessChangeRequestAutomatically, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)	
Usage:	Expl.: This function can be used to confirm a Potential Backorder after a final Partial Receipt has been done. A new Purchase Backorder Line will be created. First the Purchase Backorder Data will be changed based on the input values if needed; then the Potential Backorder will be confirmed. This all is only possible if allowed; otherwise an error message is returned. If Peg Distribution is used on the purchase order line and constraint planning is implemented, then normally pegging relations will be updated in the next planning run; but not in this interface, because it has its own transaction logic. If there is a need for this, it must be met in another way. When the Potential Backorder has been confirmed successfully automatically processing of orders can be done afterwards with the Public Interface PurchaseOrder.StartAutomaticProcessing() if needed. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iPurchaseOrder - Purchase Order (mandatory) iPurchaseOrderLine - Purchase Order Line (mandatory) iPurchaseOrderLineSequence - Purchase Order Line Sequence (mandatory) iBackorderQuantityToBeConfirmed - Backorder Quantity to be confirmed (Quantity in Backorder Unit; if given value is zero no new receipt can be done anymore) iBackorderUnit - Backorder Unit (if given value is 'empty' default system value is used)	

	<code>iBackorderPlannedReceiptDate</code> - Backorder Planned Receipt Date (if given value is zero default system value is used) <code>iNotesTextNumber</code> - Notes that are related to the specific backorder line (if given Text Number is zero default system value is used) <code>iPegChangeReason</code> - The Peg Change Reason that will be stored in the Peg Audit History. Mandatory if parameter 'Audit Manual Project Peg Modifications' is ON. <code>iApproveAndProcessChangeRequestAutomatically</code> - Yes: If Change Requests are applicable the created change request will be approved and processed automatically. No: Approval and processing of the change request (if any) is not done automatically. Output: <code>oExceptionMessage</code> - The last message if any message is found. If more than one message is given, these are present in the <code>oExceptionID</code>. <code>oExceptionID</code> - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Potential Backorder is confirmed <> 0 - An error occurred

PurchaseOrderLine.GetCurrentActivity

DLL:	tdextpurapi
	This function is available from KB2301531 .
Syntax:	<pre> long PurchaseOrderLine.GetCurrentActivity(domain tcorno iPurchaseOrder, domain tcpono iPurchaseOrderLine, domain tcpono iPurchaseOrderLineSequence, ref domain tcmcs.st20m oCurrentActivity mb, ref domain tcmcs.str60m oDescription mb, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function retrieves the current activity and associated (status) description for the given purchase order line. Pre: Not Applicable Post: Not Applicable Input: <code>iPurchaseOrder</code> - Purchase Order (Mandatory) <code>iPurchaseOrderLine</code> - Purchase Order Line (Mandatory) <code>iPurchaseOrderLineSequence</code> - Purchase Order Line Sequence (Mandatory) Output: <code>oCurrentActivity</code> - Current Activity <code>oDescription</code> - Activity (Status) Description <code>oExceptionMessage</code> - The last message if any message is found. If more than one message is given, these are present in the <code>oExceptionID</code>. <code>oExceptionID</code> - An ID that refers to the exception

	information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - Function was executed successful - An error occurred during execution of the function.

PurchaseOrderLine.ReleaseToWarehousing

DLL:	tdextpurapi
Syntax:	<pre>long PurchaseOrderLine.ReleaseToWarehousing(domain tcorno iPurchaseOrder, domain tcpono iOrderLine, domain tcpono iOrderLineSequence, ref tcyesno oReleased, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl.: This function handles the release of a purchase order line to warehousing. If applicable, this function will execute the Price Stage blocking functionality. As a result, the order line can be blocked and no release is done. This is not an error situation so the function returns with 0. Moreover, the caller must still commit the transaction, otherwise the Price-Stage Blocking changes are not committed to the database. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iPurchaseOrder Purchase Order (Mandatory) iOrderLine Purchase Order Line (Mandatory) iOrderLineSequence Purchase Order Line Sequence. Output: oReleased Yes: The order line has been released to warehousing No: The order line has not been released to warehousing. Error or information messages may be present in oExceptionID. oExceptionMessage The last message. This can also be filled if the return value equals 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 The function has been executed successfully. Either Release to Warehousing has been executed, or the order line is blocked due to Price Stages. <> 0 An error occurred

PurchaseOrderLine.Split

DLL:	tdextpurapi This function is available from KB2312115 .																						
Syntax:	<pre> long PurchaseOrderLine.Split(domain tcorno iPurchaseOrder, domain tcpono iOrderLine, domain tcpono iOrderLineSequence, domain tcqrdl iQuantityForNewDetail, domain tcdate iPlannedReceiptDateForNewDetail, domain tcdis iPegChangeReason, domain tcyesno iKeepConfirmedReceiptDates, domain tcyesno iApproveAndProcessChangeRequestAutomatically, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>																						
Usage:	Expl: This function handles the split purchase order line or detail process for a public interface. It performs similar functionality as splitting order lines / details in the purchase order line / details sessions. Note: 1. In case change requests are applicable and required, a CR is initiated to perform the split action. An option is provided to automatically approve/process the change request to the order (it performs a straightforward approve; no recalculations etc.). The change request will have origin 'Manual'. 2. This function does not trigger any automatic processing afterwards, if applicable. 3. This function does not update the planning pegging data. Regular planning run will update this. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: <table border="0"> <tr> <td>iPurchaseOrder</td><td>Purchase Order (Mandatory)</td></tr> <tr> <td>iOrderLine</td><td>Purchase Order Line (Mandatory)</td></tr> <tr> <td>iOrderLineSequence</td><td>Purchase Order Line Sequence (Mandatory)</td></tr> <tr> <td>iQuantityForNewDetail</td><td>The quantity that will be used for the new detail line (Mandatory)</td></tr> <tr> <td>iPlannedReceiptDateForNewDetail</td><td>-</td></tr> <tr> <td></td><td>The planned receipt date that will be used for the new detail line (Mandatory)</td></tr> <tr> <td>iPegChangeReason</td><td>The Peg Change Reason that will be stored in the Peg Audit History. (Mandatory if parameter 'Audit Manual Project Peg Modifications' is set to Yes in Project Pegging Parameters.)</td></tr> <tr> <td>iKeepConfirmedReceiptDates</td><td>Yes: Indicates that the Confirmed Receipt Dates of the given purchase order line will also be used on the new detail line.</td></tr> <tr> <td></td><td>No: Confirmed Receipt Dates will be cleared on the new detail line.</td></tr> <tr> <td>iApproveAndProcessChangeRequestAutomatically</td><td>Yes: If a change request was created, then it will be approved and processed automatically.</td></tr> <tr> <td></td><td>No: If a change request was created,</td></tr> </table>	iPurchaseOrder	Purchase Order (Mandatory)	iOrderLine	Purchase Order Line (Mandatory)	iOrderLineSequence	Purchase Order Line Sequence (Mandatory)	iQuantityForNewDetail	The quantity that will be used for the new detail line (Mandatory)	iPlannedReceiptDateForNewDetail	-		The planned receipt date that will be used for the new detail line (Mandatory)	iPegChangeReason	The Peg Change Reason that will be stored in the Peg Audit History. (Mandatory if parameter 'Audit Manual Project Peg Modifications' is set to Yes in Project Pegging Parameters.)	iKeepConfirmedReceiptDates	Yes: Indicates that the Confirmed Receipt Dates of the given purchase order line will also be used on the new detail line.		No: Confirmed Receipt Dates will be cleared on the new detail line.	iApproveAndProcessChangeRequestAutomatically	Yes: If a change request was created, then it will be approved and processed automatically.		No: If a change request was created,
iPurchaseOrder	Purchase Order (Mandatory)																						
iOrderLine	Purchase Order Line (Mandatory)																						
iOrderLineSequence	Purchase Order Line Sequence (Mandatory)																						
iQuantityForNewDetail	The quantity that will be used for the new detail line (Mandatory)																						
iPlannedReceiptDateForNewDetail	-																						
	The planned receipt date that will be used for the new detail line (Mandatory)																						
iPegChangeReason	The Peg Change Reason that will be stored in the Peg Audit History. (Mandatory if parameter 'Audit Manual Project Peg Modifications' is set to Yes in Project Pegging Parameters.)																						
iKeepConfirmedReceiptDates	Yes: Indicates that the Confirmed Receipt Dates of the given purchase order line will also be used on the new detail line.																						
	No: Confirmed Receipt Dates will be cleared on the new detail line.																						
iApproveAndProcessChangeRequestAutomatically	Yes: If a change request was created, then it will be approved and processed automatically.																						
	No: If a change request was created,																						

	then it will NOT be approved and processed automatically. Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 The order line has been split. <> 0 An error occurred

PurchaseOrderLines.StartOverview

DLL:	tdextpurapi
Syntax:	<pre> long PurchaseOrderLines.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iOrder, domain tcpono iOrderLine, domain tcpono iOrderLineSequence, domain tcitem iItem, domain tcsite iSite, domain tccom.bpid iBuyFromBusinessPartner, domain tcdate iCurrentPlannedReceiptDate, ref domain tcorno oOrder, ref domain tcpono oOrderLine, ref domain tcpono oOrderLineSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Purchase Order Lines Overview (tdpur410lm000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex The session index that will be used. iQueryExtend A specific query to be used when zooming to this session. iOrder Order (Mandatory if iStartMode is MODELESS and iSessionIndex is 1) iOrderLine Order Line iOrderLineSequence Order Line Sequence iItem Item (Mandatory if iStartMode is MODELESS and iSessionIndex is 2) iSite Site

	<pre> iBuyFromBusinessPartner Buy-from Business Partner (Mandatory if iStartMode is MODELESS and iSessionIndex is 3) iCurrentPlannedReceiptDate Current Planned Receipt Date Output: for iStartMode MODAL: oOrder Order number of the selected order line oOrderLine Order Line of the selected order line oOrderLineSequence Sequence of the selected order line oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - An error occurred </pre>

Public Interfaces for PurchaseOrderReceipt

The following functions are available:

[PurchaseOrderReceipt.Confirm](#)

[PurchaseOrderReceipt.ConfirmV2](#)

[PurchaseOrderReceipt.StartDetail](#)

PurchaseOrderReceipt.Confirm

DLL:	tdextpurapi
	This function is available from KB3537140 .
Syntax:	<pre> long PurchaseOrderReceipt.Confirm(domain tcorno iPurchaseOrder, domain tcpono iOrderLine, domain tcpono iOrderLineSequence, domain tcpono iReceiptSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl.: *** Warning *** </pre>

	This public interface is deprecated. use: PurchaseOrderReceipt.ConfirmV2 ***** This function handles the confirm of the given purchase receipt line. Received quantity tolerances are checked before starting the confirm process of the purchase receipt line. This can result in a warning message that is put on the message stack. This check does not result in an error that will stop the process. After that checks are done to determine if it allowed to confirm the purchase receipt line. If it is not allowed an error will be given. Next if modeled so in the system, before the confirm process, Price Stage blocking functionality is executed and as a result order line can be blocked (no confirm is done) and the blocking signal during the blocking process can be set as an error message on the message stack. This is not an error situation so function returns with 0. iReceiptSequence is the sequence from that comes from the Purchase Actual Receipts (tdpur406) record. This record must be present and allowed to be confirmed. Note: This function does not start the execution of automatic order steps. A separate Public Interface can be used to start automatic order steps if necessary: 'PurchaseOrder.StartAutomaticProcessing' Pre: LN Application sets retry-point, so not by caller Post: LN Application sets commit/abort transaction, so not by caller Input: iPurchaseOrder Purchase Order (Mandatory) iOrderLine Purchase Order Line (Mandatory) iOrderLineSequence Purchase Order Line Sequence (Mandatory) iReceiptSequence Purchase Order Line Receipt Sequence (Mandatory) Output: oExceptionMessage The last message. This can also be filled if the return value equals 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Confirm successfull. <> 0 An error occurred

PurchaseOrderReceipt.ConfirmV2

DLL:	tdextpurapi
	This function is available from KB3537140 .

Syntax:	<pre> long PurchaseOrderReceipt.ConfirmV2(domain tcorno iPurchaseOrder, domain tcpono iOrderLine, domain tcpono iOrderLineSequence, domain tcpono iReceiptSequence, boolean ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function handles the confirm of the given purchase receipt line. Received quantity tolerances are checked before starting the confirm process of the purchase receipt line. This can result in a warning message that is put on the message stack. This check does not result in an error that will stop the process. After that checks are done to determine if it allowed to confirm the purchase receipt line. If it is not allowed an error will be given. Next if modeled so in the system, before the confirm process, Price Stage blocking functionality is executed and as a result order line can be blocked (no confirm is done) and the blocking signal during the blocking process can be set as an error message on the message stack. This is not an error situation so function returns with 0. iReceiptSequence is the sequence from that comes from the Purchase Actual Receipts (tdpur406) record. This record must be present and allowed to be confirmed. Note: This function does not start the execution of automatic order steps. A separate Public Interface can be used to start automatic order steps if necessary: 'PurchaseOrder.StartAutomaticProcessing' Pre: Setting a retry-point depends on 'iHandleTransaction': - true: No need to set a retry-point. This will be taken care of by the Public Interface. - false: Caller must set retry-point Post: Committing/aborting the transaction depends on 'iHandleTransaction': - true: No need to commit/abort. This will be taken care of by the Public Interface. - false: Caller must commit/abort the transaction. Input: - iPurchaseOrder Purchase Order (Mandatory) - iOrderLine Purchase Order Line (Mandatory) - iOrderLineSequence Purchase Order Line Sequence (Mandatory) - iReceiptSequence Purchase Order Line Receipt Sequence (Mandatory) - iHandleTransaction Handling of transaction management. Possible values: - true: Transaction management will be taken care of by the Public Interface. - false: Caller must handle the transaction management: set a retry-point and commit/abort the transaction. Output: oExceptionMessage The last message. This can also be filled if the return value equals 0. If more than one message is given,

	oExceptionID Return: 0 <> 0	these are present in the oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Function executed successfully. Note that if Price Stage blocking was applicable then the order line can be blocked and no confirm has been done. An error occurred

PurchaseOrderReceipt.StartDetail

DLL:	tdextpurapi
Syntax:	<pre>long PurchaseOrderReceipt.StartDetail(long iStartMode, domain tcorno iPurchaseOrder, domain tcpono iOrderLine, domain tcpono iOrderLineSequence, domain tcpono iReceiptSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the detail session Purchase Receipts (tdpur4106m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. MODAL+START.WITH.ADD.SET - Session is started in Add-mode, so a new receipt can be entered directly. The parent session is blocked until the child session exits. MODELESS+START.WITH.ADD.SET - Session is started in Add-mode, so a new receipt can be entered directly. Parent and child are parallel sessions that can be manipulated simultaneously. iPurchaseOrder Purchase Order (Mandatory) iOrderLine Purchase Order Line (Mandatory) iOrderLineSequence Purchase Sequence number Note that the given purchase order line must exist. iReceiptSequence Purchase Order Line Receipt Sequence Unless the session is started in Add-mode, the given receipt-sequence is mandatory and must exist. Output: oExceptionMessage The last message if the return

	<pre> oExceptionID Return: 0 <> 0 </pre>	value is not equal to 0. If more than one message is given, these are present in the oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Session started An error occurred

Public Interfaces for PurchasePayableReceipt

The following functions are available:

[PurchasePayableReceipt.ChangePriceAndDiscountAfterReceipt](#)

PurchasePayableReceipt.ChangePriceAndDiscountAfterReceipt

DLL:	tdextpurapi
Syntax:	<pre> long PurchasePayableReceipt.ChangePriceAndDiscountAfterReceipt(long iObjectType, domain tcorno iPurchaseOrderOrSchedule, domain tcpono iLine, domain tcpono iLineSequence, domain tcpono iReceiptSequence, domain tcpsno iPayableReceiptSequence, boolean iAllowPriceChangeInvoicedLines, domain tcyesno iApplyPriceAndDiscountsToOtherPayableReceipts, domain tcyesno iApplyPriceStageToOtherPayableReceipts, domain tcyesno iUpdatePriceInItemPurchase, domain tcyesno iUpdatePriceInItemPurchaseBySite, domain tcyesno iApproveAndProcessChangeRequestAutomatically, domain tcpric iPrice, domain tcprsg iPriceStage, const domain tcdisc iDiscountPercentage(), const domain tddiam iDiscountAmount(), const domain tddmth iDiscountMethod(), const domain tccdsc iDiscountCode() fixed, domain tcamnt iPayableAmount, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: Function can be used to change the Price and Discounts after Receipt. The change is done for the Purchase Payable Receipts.

If ION Workflow Document Approval is used, then the following applies:

- * if the Order is waiting for Approval in ION, then updating is not allowed.

Payable receipt lines are only updated if the invoicing status is free.

Price changes must be possible (meaning that e.g. price is not read-only (yet))

When a contract is linked and prices and/or discounts are changed the contract will be unlinked depending on the 'Always Use Contract Price and Discount' setting.

By default, changes are not done when the Invoice number on the Payable Receipt is already filled. Setting iAllowPriceChangeInvoicedLines can overrule this.

By setting iApplyPriceAndDiscountsToOtherPayableReceipts it is possible to apply the Price and Discount changes also to the other Payable Receipt of the same Line / Line Sequence.

By setting iApplyPriceStageToOtherPayableReceipts it is possible to apply the Price Stage changes also to the other Payable Receipt of the same Line / Line Sequence.

By setting iUpdatePriceInItemPurchase it is possible to update the Purchase Price in Item Purchase.

By setting iUpdatePriceInItemPurchaseBySite it is possible to update the Purchase Price in Item Purchase by Site. The Purchase Price is updated for the Site of the given Object.

If Change Requests are applicable for the given Purchase Order, then the following applies:

- * A change request will be created for the status change.
- * Depending on the input argument 'iApproveAndProcessChangeRequestAutomatically', Approving and Processing of the Change Request will be done automatically or not.

Pre: Caller must set retry-point

Post: Caller must commit/abort transaction

Input: iObjectType - Object Type;
Possible values:
1: Purchase Order
2: Purchase Schedule;
(Mandatory)

iPurchaseOrderOrSchedule - Purchase Order or
Schedule Number;
(Mandatory)

iLine - Order or Schedule Line;
(Mandatory)

iLineSequence - Line Sequence

iReceiptSequence - Receipt Sequence

iPayableReceiptSequence - Payable Receipt Sequence;
(Mandatory)

iAllowPriceChangeInvoicedLines - True: Allow price changes on
invoiced lines.
False: Price changes are not
not allowed on invoiced lines. Line will be skipped.

iApplyPriceAndDiscountsToOtherPayableReceipts -

	Yes: Other Payable Receipt of the same Line / Line Sequence will be updated with Price and Discounts. No: Other Payable Receipt of the same Line / Line Sequence will not be updated. iApplyPriceStageToOtherPayableReceipts - Yes: Other Payable Receipt of the same Line / Line Sequence will be updated with Price Stage. No: Other Payable Receipt of the same Line / Line Sequence will not be updated. iUpdatePriceInItemPurchase - Yes: Price in Item Purchase will be updated. No: Price is not updated iUpdatePriceInItemPurchaseBySite - Yes: Price in Item Purchase by Site will be updated. No: Price is not updated iApproveAndProcessChangeRequestAutomatically - Yes: The created change request will be approved and processed automatically. No: Approval and processing of the change request (if any) are not done automatically. iPrice - Price (In order Currency) iPriceStage - Price Stage iDiscountPercentage - Discount Percentage; Array of 11 elements iDiscountAmount - Discount Amount; Array of 11 elements (In order Currency) iDiscountMethod - Discount Method; Array of 11 elements iDiscountCode - Discount Code; Array of 11 elements iPayableAmount - Payable Amount; Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Price and Discounts are updated. <> 0 - An error occurred

Public Interfaces for PurchaseOrderMaterialSupplyLine

The following functions are available:

[PurchaseOrderMaterialSupplyLines.StartOverview](#)

PurchaseOrderMaterialSupplyLines.StartOverview

DLL:	tdxtpurapi This function is available from KB3605556 .
Syntax:	<pre> long PurchaseOrderMaterialSupplyLines.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iPurchaseOrder, domain tcpono iPurchaseOrderLine, domain tcpono iPurchaseOrderLineSequence, domain tcpono iMaterialSequence, domain tcitem iItem, domain tcorno iSupplyOrder, domain tcpono iSupplyOrderLine, ref domain tcorno oPurchaseOrder, ref domain tcpono oPurchaseOrderLine, ref domain tcpono oPurchaseOrderLineSequence, ref domain tcpono oMaterialSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Purchase Order Material Supply Lines Overview (tdpur4116m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used iSessionIndex Specifies the session-index that is to be used. Supported values: 1: sort by Purchase Order, Line, Sequence, Material Sequence (default) 2: sort by Item 3: sort by Supply Order, Supply Order Line iQueryExtend A specific query to be used when zooming to this session. iPurchaseOrder Purchase Order Mandatory if iStartMode = MODELESS and session-index 1 is used. iPurchaseOrderLine Purchase Order Line iPurchaseOrderLineSequence Purchase Order Line Sequence iMaterialSequence

	Material Sequence iItem Item Mandatory if iStartMode = MODELESS and session-index 2 is used. iSupplyOrder Supply Order iSupplyOrderLine Supply Order Line Output: for iStartMode MODAL: oPurchaseOrder The selected Purchase Order oPurchaseOrderLine The selected Purchase Order Line oPurchaseOrderLineSequence The selected Purchase Order Line Sequence oMaterialSequence The selected Material Sequence oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

Public Interfaces for PurchaseOrderChangeRequest

The following functions are available:

[PurchaseOrderChangeRequest.Approve](#)

[PurchaseOrderChangeRequest.Process](#)

PurchaseOrderChangeRequest.Approve

DLL:	tdextpurapi
	This function is available from KB2323365 .
Syntax:	<pre>long PurchaseOrderChangeRequest.Approve(domain tcorno iChangeRequest, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl.: This function approves a modified purchase order change request. If Change Requests are setup to be processed automatically after

	approval, then processing the Change Request will be done as well. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iChangeRequest - Purchase Order Change Request (mandatory) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 -> The change request has been approved <> 0 -> An error occurred

PurchaseOrderChangeRequest.Process

DLL:	tdextpurapi
	This function is available from KB2323365 .
Syntax:	<pre>long PurchaseOrderChangeRequest.Process(domain tcorno iChangeRequest, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl.: This function processes the given purchase order change request. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iChangeRequest - Purchase Order Change Request (mandatory) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 -> The change request has been processed <> 0 -> An error occurred

Chapter 13 Public Interfaces for Pricing

Public Interfaces for Pricing

The following functions are available:

[Pricing.CalculateAmounts](#)

[Pricing.GetGeneralSettings](#)

[Pricing.GetPurchaseSettings](#)

[Pricing.GetSalesSettings](#)

[Pricing.GetServiceSettings](#)

[Pricing.RetrievePriceBookDataForItem](#)

[Pricing.SimulatePurchasePrice](#)

[Pricing.SimulateSalesPrice](#)

[Pricing.SimulateServicePrice](#)

[Pricing.SimulateTransferPrice](#)

Pricing.CalculateAmounts

DLL:	tdextpcgapi
Syntax:	<pre> long Pricing.CalculateAmounts(domain tdpcg.tyor iTypeOfOrder, domain tcitem iItem, domain tcmpnr iManufacturerPartNumber, domain tcmcs.cmnf iManufacturer, domain tcqsl1 iQuantity, domain tccuni iQuantityUnit, domain tcconv iQuantityUnitConversionFactor, domain tcpric iPrice, domain tccuni iPriceUnit, domain tcconv iPriceUnitConversionFactor, domain tdpcg.prbk iPriceBook, </pre>

	<pre> domain tccur iTransactionCurrency, domain tccrnd iRoundingMethod, const domain tcdisc iDiscountPercentage(), const domain tddiam iDiscountAmount(), const domain tddmth iDiscountMethod(), const domain tdgen.dorg iDiscountOrigin(), const domain tdpcg.dssc iDiscountSchedule() fixed, domain tcdate iPriceDate, domain tccwoc iOffice, boolean iRoundAmounts, ref domain tcamnt oNetAmount, ref domain tcamnt oGrossAmount, ref domain tcamnt oTotalDiscountAmount, ref domain tcdisc oStructureDiscountPercentage, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface calculates the Net Amount, Gross Amount, the Discount Amount, and the Structure Discount Percentage. Definitions: 1) Structure Discount Amount is the sum of all Discount Amounts that are not Manual. 2) Total discount Amount is the sum of all Discount Amounts. Calculations: 1) Gross Amount = Quantity (in inv.unit) x Price (in inv.unit) 2) Total Discount Amount is based upon the Discount Arrays 3) Net Amount = Gross Amount - Total Discount Amount 4) Structure Discount Percentage = $\text{Structure Discount Amount} / \text{Gross Amount} \times 100\%$ In case the Type of Order is Purchase Contract (tdpcg.tyor.pc), Sales Contract (tdpcg.tyor.sc) or Request for Quotation (tdpcg.tyor.pq), then it is possible to retrieve the Price from the Contract/RFQ Price Book and the Discounts from the Contract/RFQ Discount Schedules. In that case no Price and Discounts must be passed along. This is illustrated in the tables below. legend: 'x' means: mandatory '-' means: leave empty PRICES: Type of Order = tdpcg.tyor.pc (Purchase Contract) <pre> ----- iPrice x - iPriceUnit x - iPriceUnitConversionFactor x - iPriceBook - x </pre> Explanation for the table above: Either iPrice, iPriceUnit and iPriceUnitConversionFactor must be filled (and iPriceBook must then be empty), OR iPrice, iPriceUnit and iPriceUnitConversionFactor must be EMPTY (and iPriceBook must then be filled). Type of Order = tdpcg.tyor.sc (Sales Contract) <pre> ----- iPrice x - iPriceUnit x - </pre>

iPriceUnitConversionFactor	x	-
iPriceBook	-	x
Type of Order = tdpcg.tyor.pq (RFQ)		

iPrice	x	-
iPriceUnit	x	-
iPriceUnitConversionFactor	x	-
iPriceBook	-	x
Type of Order = tdpcg.tyor.so/sq/po/tr/sr/gs/gp		

iPrice	x	
iPriceUnit	x	
iPriceUnitConversionFactor	x	
iPriceBook	-	

DISCOUNTS:		

Type of Order = tdpcg.tyor.pc (Purchase Contract)		

iDiscountPercentage	x	-
iDiscountAmount	x	-
iDiscountMethod	x	-
iDiscountOrigin	x	tdgen.dorg.not.applicable
iDiscountSchedule	-	x
Type of Order = tdpcg.tyor.sc (Sales Contract)		

iDiscountPercentage	x	-
iDiscountAmount	x	-
iDiscountMethod	x	-
iDiscountOrigin	x	tdgen.dorg.not.applicable
iDiscountSchedule	-	x
Type of Order = tdpcg.tyor.pq (RFQ)		

iDiscountPercentage	x	-
iDiscountAmount	x	-
iDiscountMethod	x	-
iDiscountOrigin	x	tdgen.dorg.not.applicable
iDiscountSchedule	-	x
Type of Order = tdpcg.tyor.so/sq/po/tr/sr/gs/gp		

iDiscountPercentage	x	
iDiscountAmount	x	
iDiscountMethod	x	
iDiscountOrigin	x	
iDiscountSchedule	-	
Pre:	NA	
Post:	NA	
Input:	iTypeOfOrder	- Mandatory
	All enum values are allowed:	
	tdpcg.tyor.so - Sales Order	
	tdpcg.tyor.sc - Sales Contract	
	tdpcg.tyor.sq - Sales Quotation	
	tdpcg.tyor.po - Purchase Order	

	tdpcg.tyor.pc - Purchase Contract tdpcg.tyor.pq - Request for Quotation tdpcg.tyor.tr - Transfer tdpcg.tyor.sr - General Service tdpcg.tyor.gs - General Sales tdpcg.tyor.gp - General Purchase tdpcg.tyor.fm - Freight Management tdpcg.tyor.sr.soc - Service Order tdpcg.tyor.sr.msc - Maintenance Sales Order tdpcg.tyor.sr.quote - Service Quote
iItem	- Item; Mandatory
iManufacturerPartNumber	- Manufacturer Part Number; Optional This field is only used when iPrice is zero and the price is to be found using the given price book.
iManufacturer	- Manufacturer; Optional This field is only used when iPrice is zero and the price is to be found using the given price book.
iQuantity	- Quantity; (zero is allowed)
iQuantityUnit	- Quantity Unit; Mandatory
iQuantityUnitConversionFactor	- Conversion Factor of the Quantity Unit; Mandatory
iPrice	- Price; Optional
iPriceUnit	- Price Unit; Mandatory if iPrice <> 0
iPriceUnitConversionFactor	- Conversion Factor of the Price Unit; Mandatory if iPrice <> 0
iPriceBook	- Price Book; Optional
iTransactionCurrency	- Transaction Currency; Mandatory
iRoundingMethod	- Rounding Method; Optional
iDiscountPercentage	- Discount Percentage; Optional
iDiscountAmount	- Discount Amount; Optional
iDiscountMethod	- Discount Method; Optional
iDiscountOrigin	- Discount Origin; Optional
iDiscountSchedule	- Discount Schedule; Optional
iPriceDate	- Date of the Order, Contract, RFQ, etc; Mandatory
iOffice	- Sales Office, Purchase Office, Contract Office, etc. Optional This field is only used when iPrice is zero and the price is to be found using the given price book.
iRoundAmounts	- true: The amounts will be rounded according to the given currency and rounding method. - false: The amounts will not be rounded.
Output: oNetAmount	- Net Amount
oGrossAmount	- Gross Amount
oTotalDiscountAmount	- Total Discount Amount
oStructureDiscountPercentage	- Structure Discount Percentage
oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
oExceptionID	- An ID that refers to all error information. Use the functions in

	Exception to get all relevant information. Return: 0 <> 0 - Function was executed successfull. - An error occurred.

Pricing.GetGeneralSettings

DLL:	tdextpcgapi
Syntax:	<pre> long Pricing.GetGeneralSettings(domain tcncmp iLogisticCompany, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves general pricing settings. Individual fields are retrieved, multiple fields can be retrieved in one call. Data is always read from parameter level (parameter table tdpcg000). Notes: 1. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when number of discount levels and price book number group are needed, the function must be called as follows: <pre> if Pricing.GetGeneralSettings(* Fixed arguments: company, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output * Variable arguments: "nmdl", --> input number.of.discount.levels, --> output "ngpb", --> input price.book.number.group) <> 0 then --> output * Error, do something Exception.Delete(exception.id) endif </pre> Pre: None Post: None Input: - iLogisticCompany - Logistic Company: Mandatory - iForceRead - Option to force new query in stead of using cached information - iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional

	<pre> ... Output: oExceptionMessage oExceptionID Return: 0 <> 0 </pre>	<pre> data. - The field mnemonic of the required field. - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - The value of the required field. - Data read - An error occurred </pre>

Pricing.GetPurchaseSettings

DLL:	tdextpcgapi
Syntax:	<pre> long Pricing.GetPurchaseSettings(domain tcncmp iLogisticCompany, domain tcsite iSite, domain tccwoc iPurchaseOffice, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves purchase pricing settings. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tdpcg000), or site or office level (settings table tdpcg080) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: - Fields that are only defined at parameter level (not defined per office or site) are also retrieved. - This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. - Several table-fields are stored as arrays. Individual elements of these table-fields can be retrieved by adding the element number to the field-mnemonic. Example: "clsi(2)", "rrsi(7)". Example: when target price currency and price book number group are needed, the function must be called as follows: <pre> if Pricing.GetPurchaseSettings(* Fixed arguments: company, --> input </pre>

	<pre> site, --> input purchase office, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output /* Variable arguments: "tcur", --> input target.price.currency, --> output "ngpb", --> input price.book.number.group) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif </pre>
Pre:	None
Post:	None
Input:	<pre> iLogisticCompany - Logistic Company: Mandatory iSite - Site: Not Mandatory iPurchaseOffice - Purchase Office: Not Mandatory iForceRead - Option to force new query in stead of using cached information iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. </pre>
Output:	<pre> oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. </pre>
Return:	<pre> 0 - Data read <> 0 - An error occurred </pre>

Pricing.GetSalesSettings

DLL:	tdextpcgapi
Syntax:	<pre> long Pricing.GetSalesSettings(domain tcncmp iLogisticCompany, domain tccwoc iSalesOffice, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, </pre>

Usage:	...) Expl: This function retrieves sales pricing settings. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tdpcg000), or site or office level (settings table tdpcg081) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when attribute set for sales price and default sales price book are needed, the function must be called as follows: <pre> if Pricing.GetSalesSettings(* Fixed arguments: company, --> input sales office, --> input site, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output * Variable arguments: "assp", --> input attribute.set.for.sales.price, --> output "sdpb", --> input default.sales.price.book) <> 0 then --> output * Error, do something Exception.Delete(exception.id) endif </pre> Pre: None Post: None Input: <table border="0"> <tr> <td>iLogisticCompany</td><td>- Logistic Company: Mandatory</td></tr> <tr> <td>iSalesOffice</td><td>- Sales Office: Not Mandatory</td></tr> <tr> <td>iSite</td><td>- Site: Not Mandatory</td></tr> <tr> <td>iForceRead</td><td>- Option to force new query in stead of using cached information</td></tr> <tr> <td>iContextIsMasterData</td><td>- True/False: Indicates if call is done to retrieve data for master data or transactional data.</td></tr> <tr> <td>...</td><td>- The field mnemonic of the required field.</td></tr> </table> Output: <table border="0"> <tr> <td>oExceptionMessage</td><td>- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID</td></tr> <tr> <td>oExceptionID</td><td>- An ID that refers to all error information. Use the functions in</td></tr> </table>	iLogisticCompany	- Logistic Company: Mandatory	iSalesOffice	- Sales Office: Not Mandatory	iSite	- Site: Not Mandatory	iForceRead	- Option to force new query in stead of using cached information	iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.	...	- The field mnemonic of the required field.	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID	oExceptionID	- An ID that refers to all error information. Use the functions in
iLogisticCompany	- Logistic Company: Mandatory																
iSalesOffice	- Sales Office: Not Mandatory																
iSite	- Site: Not Mandatory																
iForceRead	- Option to force new query in stead of using cached information																
iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.																
...	- The field mnemonic of the required field.																
oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID																
oExceptionID	- An ID that refers to all error information. Use the functions in																

	<pre> ... Return: 0 <> 0 </pre>	Exception to get all relevant information. - The value of the required field. - Data read - An error occurred

Pricing.GetServiceSettings

DLL:	tdextpcgapi
Syntax:	<pre> long Pricing.GetServiceSettings(domain tcncmp iLogisticCompany, domain tccwoc iServiceOffice, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves service pricing settings. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tdpcg000), or site or office level (settings table tdpcg082) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when default service price book is needed, the function must be called as follows: <pre> if Pricing.GetServiceSettings(/* Fixed arguments: company, --> input service.office, --> input site, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output /* Variable arguments: "sspb", --> input default.service.price.book) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif </pre>

Pre:	None	
Post:	None	
Input:		
	iLogisticCompany	- Logistic Company: Mandatory
	iServiceOffice	- Service Office: Not Mandatory
	iSite	- Site: Not Mandatory
	iForceRead	- Option to force new query in stead of using cached information
	iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.
	...	- The field mnemonic of the required field.
Output:		
	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
	...	- The value of the required field.
Return:	0	- Data read
	<> 0	- An error occurred

Pricing.RetrievePriceBookDataForItem

DLL:	tdextpcgapi
Syntax:	<pre> long Pricing.RetrievePriceBookDataForItem(domain tdpcg.prbk iPriceBook, domain tdpcg.prit iPriceType, domain tcdate iPriceDate, domain tcitem iItem, domain tcmpnr iManufacturerPartNumber, domain tcmcs.cmnf iManufacturer, domain tcqsl1 iOrderQuantity, domain tccuni iQuantityUnit, domain tcconv iQuantityUnitConversionFactor, domain tdpcg.prpc iPricePercentage, domain tccuni iPriceUnit, domain tcccur iOrderCurrency, domain tccur iHomeCurrency, domain tccom.bpid iBuyFromBusinessPartner, domain tccom.bpid iShipFromBusinessPartner, domain tcyesno iHighPrio, domain tdpcg.tyor iTypeOfOrder, domain tccwoc iOffice, domain boolean iIgnoreQuantityBreaks, ref domain tdpcg.prbk oPriceBook, ref domain tcmcs.long oPriceBookLine, ref domain tdpcg.prit oPriceType, ref domain tcrefa oReference mb, </pre>

	<pre> ref domain tcrefa oSource mb, ref domain tcyesno oUsedForQuotations, ref domain tctrns.date oEntryDate, ref domain tcefex.date oEffectiveDate, ref domain tcefex.date oExpiryDate, ref domain tcitem oItem, ref domain tcmpnr oManufacturerPartNumber, ref domain tcmcs.cmnf oManufacturer, ref domain tdpcg.brty oBreakType, ref domain tcqsl1 oBreakQuantityValue, ref domain tccuni oQuantityUnit, ref domain tdpcg.prpc oPricePercentage, ref domain tcpric oPrice, ref domain tccuni oPriceUnit, ref domain tcccur oCurrency, ref domain tcyesno oBasePrice, ref domain tcprsg oPriceStage, ref domain tcyesno oUpdateOrderLinesWithPriceUnitFromPriceBook, ref domain tcperc oPercentage, ref domain tdpcg.dssc oDiscountSchedule, ref domain tccom.bpid oBuyFromBusinessPartner, ref domain tccom.bpid oShipFromBusinessPartner, ref domain tcyesno oHighPriority, ref boolean oValidPriceFound, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>																										
Usage:	Expl.: This public interface searches the proper price book record for the given input. If a price book record was found, its price data is returned in the output variables. The price book instances that are found during searching will be rated according to the search mechanism. The valid price book instance having the highest rate will be applied. When determining the rate, the following price-book fields are considered. Fields that are mentioned first in this list will contribute more to the rate than fields that are lower in the list. 1. Manufacturer Part Number and Manufacturer are filled. 2. Buy-from and ship-from Business Partner are filled. 3. Search on order currency, then on home currency, then on any currency. 4. Search on order unit, then on base unit, then on price unit. Pre: NA Post: NA Input: <table> <tr> <td>iPriceBook</td><td>- Price Book; Mandatory</td></tr> <tr> <td>iPriceType</td><td>- Price Type; indicates the procurement scenario:</td></tr> <tr> <td></td><td>- Buying</td></tr> <tr> <td></td><td>- Item Subcontracting</td></tr> <tr> <td></td><td>- Operation Subcontracting</td></tr> <tr> <td></td><td>- Service Subcontracting</td></tr> <tr> <td></td><td>- Not Applicable.</td></tr> <tr> <td>iPriceDate</td><td>- Date of the Order, Contract, RFQ, etc.</td></tr> <tr> <td>iItem</td><td>- Item; Mandatory</td></tr> <tr> <td>iManufacturerPartNumber</td><td>- Manufacturer Part Number</td></tr> <tr> <td>iManufacturer</td><td>- Manufacturer</td></tr> <tr> <td>iOrderQuantity</td><td>- Order Quantity</td></tr> <tr> <td>iQuantityUnit</td><td>- Order Quantity Unit</td></tr> </table>	iPriceBook	- Price Book; Mandatory	iPriceType	- Price Type; indicates the procurement scenario:		- Buying		- Item Subcontracting		- Operation Subcontracting		- Service Subcontracting		- Not Applicable.	iPriceDate	- Date of the Order, Contract, RFQ, etc.	iItem	- Item; Mandatory	iManufacturerPartNumber	- Manufacturer Part Number	iManufacturer	- Manufacturer	iOrderQuantity	- Order Quantity	iQuantityUnit	- Order Quantity Unit
iPriceBook	- Price Book; Mandatory																										
iPriceType	- Price Type; indicates the procurement scenario:																										
	- Buying																										
	- Item Subcontracting																										
	- Operation Subcontracting																										
	- Service Subcontracting																										
	- Not Applicable.																										
iPriceDate	- Date of the Order, Contract, RFQ, etc.																										
iItem	- Item; Mandatory																										
iManufacturerPartNumber	- Manufacturer Part Number																										
iManufacturer	- Manufacturer																										
iOrderQuantity	- Order Quantity																										
iQuantityUnit	- Order Quantity Unit																										

	iQuantityUnitConversionFactor	- Conversion factor for the Order Quantity Unit
	iPricePercentage	- Indicates the kind of price book record that is retrieved. Price: A price book record is retrieved for which the field 'Price / Percentage' (tdpcg031.perc) has the value 'Price'. Percentage: A price book record is retrieved for which the field 'Price / Percentage' (tdpcg031.perc) has the value 'Percentage'.
	iPriceUnit	- Price Unit
	iOrderCurrency	- Order Currency
	iHomeCurrency	- Home Currency
	iBuyFromBusinessPartner	- Buy-from BP
	iShipFromBusinessPartner	- Ship-from BP
	iHighPrio	- High Priority (Yes/No) Search for price record that have the High Priority flag set to the given value.
	iTypeOfOrder	- Indicates the type of order for which a price book record is searched. Allowed values include (but not limited to): Sales Order Sales Contract Sales Quotation Purchase Order Purchase Contract Request for Quotation General (Sales) General (Purchase)
	iOffice	- Office
	iIgnorePriceBreaks	- True/False Indicates if quantity breaks must be ignored.
	Output: Data found on the price-book record:	
	oPriceBook	- Price Book
	oPriceBookLine	- Price Book Line
	oPriceType	- Price Type; indicates the procurement scenario: - Buying - Item Subcontracting - Operation Subcontracting - Service Subcontracting - Not Applicable.
	oReference	- Any informative description field used to refer to, for example: * The person or department with authorization to perform a specific task. * The business partner's contact. * The original invoice number.
	oSource	- The LN component in which the price is created, or the means by which the price is created.
	oUsedForQuotations	- Yes/No

	If this field is set to Yes, the price book line is allocated to quotations. oEntryDate - The entry date of the price book line oEffectiveDate - The date and time from which the price book is valid oExpiryDate - The expiry date and time of the price book. If this field is not filled, the price book is infinitely valid. oItem - Item oManufacturerPartNumber - The manufacturer part number (MPN) linked to the price book line. oManufacturer - The code of the manufacturer linked to the price book line. oBreakType - An entity used to specify how breaks between ranges of entities such as distances, amounts, or ordered quantities of items are defined. A break, in this case, is the first or the last number of a range. A break type has either of the following values: * Minimum: The break is the lowest number of a range. * Up To: The break is the highest number of a range. oBreakQuantityValue - The minimum or maximum order quantity or value to which the price book applies. oQuantityUnit - Quantity Unit oPricePercentage - Defines, for a cost or service item, whether you can enter a price or a percentage. oPrice - Price oPriceUnit - Price Unit oCurrency - Currency oBasePrice - Yes/No If this field is Yes, the default sales or purchase price book from the Pricing Parameters session is used to store the item's prices and discounts. oPriceStage - Price Stage oUpdateOrderLinesWithPriceUnitFromPriceBook - Yes/No If this field is Yes, and an order line is entered, the price unit on the order line is similar to the price book's Price Unit. oPercentage - The additional cost item's percentage, based on which an additional cost amount is calculated. oDiscountSchedule - Discount Schedule oBuyFromBusinessPartner - Buy From Business Partner oShipFromBusinessPartner - Ship From Business Partner oHighPriority - Yes/No If this field is Yes, the price of the buy-from business partner/ship-from business partner/item combination has priority over the prices stored under the normal matrix priority structure. oValidPriceFound - Price book retrieved (True/False) oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is

	given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Function was executed successfull. - An error occurred.
Return: 0 <> 0	

Pricing.SimulatePurchasePrice

DLL:	tdextpcgapi
Syntax:	<pre> long Pricing.SimulatePurchasePrice(domain tcncmp iLogisticCompany, domain tcncmp iFinancialCompany, domain tccom.bpid iBuyFromBusinessPartner, domain tcccur iCurrency, domain tcitem iItem, domain tccom.bpid iShipFromBusinessPartner, domain tccom.bpid iInvoiceFromBusinessPartner, domain tccom.bpid iPricingBusinessPartner, domain tccwoc iPurchaseOffice, domain tcrtyp iRateType, domain tcdate iRateDate, const domain tcratc iRate(), const domain tcratf iRateFactor(), domain tccplt iPriceList, domain tccotp iOrderType, domain tccreg iArea, domain tccdec iDeliveryTerms, domain tcpaym iPaymentMethod, domain tcolid iOptionListID, domain tccpva iProductVariant, domain tcuef.effn iEffectivityUnit, domain tcguid iSpecification, domain tcmpnr iManufacturerPartNumber, domain tcmcs.cmnf iManufacturer, domain tcyesno iSubcontracted, domain tcsite iSite, domain tcefex.date iPriceDate, domain tdpcg.prit iPriceType, domain tcqsl1 iOrderQuantity, domain tccuni iOrderQuantityUnit, domain tcconv iOrderQuantityUnitConversionFactor, domain tccono iContract, domain tcpono iContractLine, domain tccwoc iContractOffice, domain tcpono iContractSequence, ref domain tcpric oPrice, ref domain tccuni oPriceUnit, ref domain tcconv oPriceUnitConversionFactor, ref boolean oDerivedItemUsed, ref domain tdpcg.prbk oPriceBook, ref domain tcprsg oPriceStage, ref domain tdgen.porg oPriceOrigin, </pre>

Public Interfaces for Pricing

	<pre> ref domain tdpcg.made oPriceMatrixDefinition, ref domain tdpcg.prse oPriceMatrixSequence, ref domain tdpcg.maty oDiscountMatrixType(), ref domain tdgen.dorg oDiscountOrigin(), ref domain tdpcg.made oDiscountMatrixDefinition() fixed, ref domain tdpcg.prse oDiscountMatrixSequence(), ref domain tcdisc oDiscountPercentage(), ref domain tddiam oDiscountAmount(), ref domain tccdsc oDiscountCode() fixed, ref domain tddmth oDiscountMethod(), ref domain tdpcg.dssc oDiscountSchedule() fixed, ref domain tccono oContract, ref domain tcpono oContractLine, ref domain tccwoc oContractOffice, ref domain tcpono oContractSequence, ref domain tcmpr.pagr oMaterialPriceAgreement, ref domain tcprip oTotalMaterialPriceSurcharges, ref domain tcprip oTotalMaterialPrice, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl: This public interface simulates purchase price and discounts retrieval, including material price surcharge information. Price and discount retrieval is according to standard LN logic, searching through the hierarchical price structure. The output indicates from which level the information is retrieved. Pre: NA Post: NA Input: ***** * Mandatory input arguments. ***** iLogisticCompany - Logistic Company iFinancialCompany - Financial Company iBuyFromBusinessPartner - Buy-from Business Partner iCurrency - Currency iItem - Item ***** * Optional input arguments; if not filled, defaults are * retrieved. ***** iShipFromBusinessPartner - Ship-from Business Partner iInvoiceFromBusinessPartner - Invoice-from Business Partner iPricingBusinessPartner - Pricing Business Partner iPurchaseOffice - Purchase Office iRateType - Exchange Rate Type iRateDate - Rate Date; if zero, then Price Date is used. iRate - Rate; if first element zero, rate and rate factor are defaulted. iRateFactor - Rate Factor; if first element is zero, rate and rate factor are defaulted. iPriceList - Price List iOrderType - Order Type iArea - Area iDeliveryTerms - Delivery Terms iPaymentMethod - Payment Method iOptionListID - Option List ID iProductVariant - Product Variant iEffectivityUnit - Effectivity Unit iSpecification - Specification </pre>

	iManufacturerPartNumber - ManufacturerPartNumber iManufacturer - Manufacturer iSubcontracted - Subcontracted; if empty, 'no' is used iSite - Site iPriceDate - Price Date; of zero, then current date and time is used. iPriceType - Price Type; if empty, 'buying' is used iOrderQuantity - Order Quantity iOrderQuantityUnit - Order Quantity Unit iOrderQuantityUnitConversionFactor - Order Quantity Unit Conversion Factor iContract - Contract; if not filled, Contract, Contract Line and Contract Office are defaulted (if applicable) without user interaction. iContractLine - Contract Line; see Contract iContractOffice - Contract Office; see Contract iContractSequence - Contract Sequence
Output:	***** * Price Output. ***** oPrice - Price oPriceUnit - Price Unit oPriceUnitConversionFactor - Price Unit Conversion Factor oDerivedItemUsed - Derived Item Used oPriceBook - Price Book oPriceStage - Price Stage oPriceOrigin - Price Origin oPriceMatrixDefinition - Price Matrix Definition oPriceMatrixSequence - Price Matrix Sequence ***** * Discount Output; memory allocation (11 levels) for arrays * needed. ***** oDiscountMatrixType - Discount Matrix Type (array) oDiscountOrigin - Discount Origin (array) oDiscountMatrixDefinition - Discount Matrix Definition (array) oDiscountMatrixSequence - Discount Matrix Sequence (array) oDiscountPercentage - Discount Percentage (array) oDiscountAmount - Discount Amount (array) oDiscountCode - Discount Code (array) oDiscountMethod - Discount Method (array) oDiscountSchedule - Discount Schedule (array) ***** * Contract Output. ***** oContract - Contract; filled if price / discount origin is 'contract' oContractLine - Contract Line; see Contract oContractOffice - Contract Office; see Contract ***** * Material Price Output. ***** oMaterialPriceAgreement - Material Price Agreement oTotalMaterialPriceSurcharges - Material Price Surcharges

	<pre> oTotalMaterialPrice - Total Material Price ***** * Technical Output. ***** oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Price could be determined. <> 0 - Error occurred. </pre>

Pricing.SimulateSalesPrice

DLL:	tdextpcgapi
Syntax:	<pre> long Pricing.SimulateSalesPrice(domain tcncmp iLogisticCompany, domain tcncmp iFinancialCompany, domain tccom.bpid iSoldToBusinessPartner, domain tcccur iCurrency, domain tcitem iItem, domain tccom.bpid iShipToBusinessPartner, domain tccom.bpid iInvoiceToBusinessPartner, domain tccom.bpid iPricingBusinessPartner, domain tccwoc iSalesOffice, domain tcrtyp iRateType, domain tcdate iRateDate, const domain tcratc iRate(), const domain tcratf iRateFactor(), domain tccplt iPriceList, domain tccotp iOrderType, domain tccreg iArea, domain tccdec iDeliveryTerms, domain tdpcg.pror iPriceBookOrigin, domain tcpaym iPaymentMethod, domain tcmcs.chan iChannel, domain tccpva iProductVariant, domain tcuef.effn iEffectivityUnit, domain tcsite iSite, domain tcefex.date iPriceDate, domain tcqsl1 iOrderQuantity, domain tccuni iOrderQuantityUnit, domain tcconv iOrderQuantityUnitConversionFactor, domain tccono iContract, domain tcpono iContractLine, domain tccwoc iContractOffice, ref domain tcpric oPrice, ref domain tccuni oPriceUnit, ref domain tcconv oPriceUnitConversionFactor, ref boolean oDerivedItemUsed, ref domain tdpcg.prbk oPriceBook, </pre>

	<pre> ref domain tcprsg oPriceStage, ref domain tdgen.porg oPriceOrigin, ref domain tdpcg.made oPriceMatrixDefinition, ref domain tdpcg.prse oPriceMatrixSequence, ref domain tdpcg.maty oDiscountMatrixType(), ref domain tdgen.dorg oDiscountOrigin(), ref domain tdpcg.made oDiscountMatrixDefinition() fixed, ref domain tdpcg.prse oDiscountMatrixSequence(), ref domain tcdisc oDiscountPercentage(), ref domain tddiam oDiscountAmount(), ref domain tccdsc oDiscountCode() fixed, ref domain tddmth oDiscountMethod(), ref domain tdpcg.dssc oDiscountSchedule() fixed, ref domain tccono oContract, ref domain tcpono oContractLine, ref domain tccwoc oContractOffice, ref domain tcmpr.pagr oMaterialPriceAgreement, ref domain tcprp oTotalMaterialPriceSurcharges, ref domain tcprp oTotalMaterialPrice, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl: This public interface simulates sales price and discounts retrieval, including material price surcharge information. Price and discount retrieval is according to standard LN logic, searching through the hierarchical price structure. The output indicates from which level the information is retrieved. Pre: NA Post: NA Input: ***** * Mandatory input arguments. ***** iLogisticCompany - Logictic Company iFinancialCompany - Financial Company iSoldToBusinessPartner - Sold-to Business Partner iCurrency - Currency iItem - Item ***** * Optional input arguments; if not filled, defaults are * retrieved. ***** iShipToBusinessPartner - Ship-to Business Partner iInvoiceToBusinessPartner - Invoice-to Business Partner iPricingBusinessPartner - Pricing Business Partner iSalesOffice - Sales Office iRateType - Exchange Rate Type iRateDate - Rate Date; if zero, then Price Date is used. iRate - Rate; if first element zero, rate and rate factor are defaulted. iRateFactor - Rate Factor; if first element is zero, rate and rate factor are defaulted. iPriceList - Price List iOrderType - Order Type iArea - Area iDeliveryTerms - Delivery Terms iPriceBookOrigin - Price Book Origin ('sales' or 'service'); if empty, then ' sales' is used. iPaymentMethod - Payment Method </pre>

	iChannel iProductVariant iEffectivityUnit iSite iPriceDate iOrderQuantity iOrderQuantityUnit iOrderQuantityUnitConversionFactor iContract iContractLine iContractOffice	- Channel - Product Variant - Effectivity Unit - Site - Price Date; of zero, then current date and time is used. - Order Quantity - Order Quantity Unit - Order Quantity Unit Conversion Factor - Contract; if not filled, Contract, Contract Line and Contract Office are defaulted (if applicable) without user interaction. - Contract Line; see Contract - Contract Office; see Contract
Output:	<pre> ***** * Price Output. ***** oPrice - Price oPriceUnit - Price Unit oPriceUnitConversionFactor - Price Unit Conversion Factor oDerivedItemUsed - Derived Item Used oPriceBook - Price Book oPriceStage - Price Stage oPriceOrigin - Price Origin oPriceMatrixDefinition - Price Matrix Definition oPriceMatrixSequence - Price Matrix Sequence ***** * Discount Output; memory allocation (11 levels) for arrays * needed. ***** oDiscountMatrixType - Discount Matrix Type (array) oDiscountOrigin - Discount Origin (array) oDiscountMatrixDefinition - Discount Matrix Definition (array) oDiscountMatrixSequence - Discount Matrix Sequence (array) oDiscountPercentage - Discount Percentage (array) oDiscountAmount - Discount Amount (array) oDiscountCode - Discount Code (array) oDiscountMethod - Discount Method (array) oDiscountSchedule - Discount Schedule (array) ***** * Contract Output. ***** oContract - Contract; filled if if price / discount origin is 'contract' oContractLine - Contract Line; see Contract oContractOffice - Contract Office; see Contract ***** * Material Price Output. ***** oMaterialPriceAgreement - Material Price Agreement oTotalMaterialPriceSurcharges - Material Price Surcharges oTotalMaterialPrice - Total Material Price ***** * Technical Output. ***** </pre>	

	oExceptionMessage oExceptionID Return: 0 <> 0	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Price could be determined. - Error occurred.

Pricing.SimulateServicePrice

DLL:	tdextpcgapi
	This function is available from KB3527699 .
Syntax:	<pre> long Pricing.SimulateServicePrice(long iProcessingOptionSet, ref domain tcpric oPrice, ref domain tccuni oPriceUnit, ref domain tcconv oPriceUnitConversionFactor, ref domain tdpcg.made oPriceMatrixDefinition, ref domain tdpcg.prse oPriceMatrixSequence, ref domain tstdm.porg oPriceOrigin, ref domain tcprsg oPriceStage, ref domain tcdisc oDiscount, ref domain tcdisc oDiscountPercentage(), ref domain tcamnt oDiscountAmount(), ref domain tccdsc oDiscountCode() fixed, ref domain tddmth oDiscountMethod(), ref domain tdpcg.maty oDiscountType(), ref domain tdpcg.made oDiscountDefinition() fixed, ref domain tdpcg.prse oDiscountSequence(), ref domain tdgen.dorg oDiscountOrigin(), ref domain tcyesno oDetermining, ref domain tcyesno oEligible, ref boolean oDerivedItemUsed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface simulates service price and discounts retrieval. Price and discount retrieval is according to standard LN logic, searching through the hierarchical price structure. Via the Processing Option Set all details surrounding the price defaulting request, such as the type of order, the Item, Business Partners and original price data can be passed to this Public Interface. Pre: Call ProcessingOptionSet.Create() to obtain iProcessingOptionSet. Post: Delete the option set by calling ProcessingOptionSet.Delete(). Input: iProcessingOptionSet Processing Option Set: Mandatory, a processing option

set number referring to the processing option set created through ProcessingOptionSet.Create().			
NAME	TYPE	DEFAULT	
=====			
TypeOfOrder	domain	tdpcg.tyor	tdpcg.tyor.sr
The type of order to find a default price for. Allowed values: - tdpcg.tyor.sr General Service - tdpcg.tyor.sr.msc Maintenance Sales - tdpcg.tyor.sr.quote Service Quote - tdpcg.tyor.sr.soc (Field) Service			
OrderNumber	domain	tcorno	""
The Order- or Quote number of the order to find a default price for. This is only applicable and mandatory if TypeOfOrder is tdpcg.tyor.sr.msc, tdpcg.tyor.sr.quote or tdpcg.tyor.sr.soc.			
UseDiscountsFromPriceBooks	domain	tctyesno	tctyesno.no
Whether or not to search for default discounts.			
Item	domain	tcitem	""
Mandatory. The Cost Line Item to find a default price for.			
Site	domain	tcsite	""
The Site to find a default price for.			
SoldToBusinessPartner	domain	tccom.bpid	""
Mandatory. The Sold-to Business Partner to find a default price for.			
ShipToBusinessPartner	domain	tccom.bpid	""
The Ship-to Business Partner to find a default price for.			
InvoiceToBusinessPartner	domain	tccom.bpid	""
Optional* The Invoice-to Business Partner to find a default price for.			
PricingBusinessPartner	domain	tccom.bpid	""
The Pricing Business Partner to find a default price for.			
ServiceOffice	domain	tccwoc	""
The Service Office to find a default price for.			

FinancialDepartment	domain	tccwoc	""
The Financial Department to find a default price for.			
SalesPriceList	domain	tcclpt	""
The Sales Price List to find a default price for.			
SalesArea	domain	tccreg	""
The Sales Area to find a default price for.			
PaymentMethod	domain	tcpaym	""
The Payment Method to find a default price for.			
TermsOfDelivery	domain	tcdec	""
The Terms of Delivery to find a default price for.			
ServiceType	domain	tsmdm.cstp	""
The Service Type to find a default price for.			
InstallationGroup	domain	tsbsc.clst	""
The Installation Group to find a default price for.			
MaintainedItem	domain	tcitem	""
The maintained Item to find a default price for.			
MaintainedSerial	domain	tcibd.sern	""
The Serial Number of the maintained Item to find a default price for.			
ServiceArea	domain	tsmdm.csar	""
The Service Area to find a default price for.			
ReferenceActivity	domain	tsacm.cact	""
The Reference Activity to find a default price for.			
MasterRouting	domain	tsacm.cact	""
The Master Routing to find a default price for.			
RoutingOption	domain	tsacm.cact	""
The Routing Option to find a default price for.			
Quantity	domain	tsmdm.qmat	0
The Quantity of the cost line to find a default price for.			
QuantityUnit	domain	tccuni	""
Optional*			
The Quantity Unit of the cost line to find a default			

	price for.
QuantityUnitConversionFactor	domain tconv 0.0
Optional*	
	The Quantity Unit Conversion Factor of the cost line to find a default price for.
Currency	domain tccur ""
Optional*	
	The Currency to find a default price in.
CurrencyRate1	domain tcratc 0.0
	The Currency Rate from the local currency to the reference currency or given currency.
CurrencyRate2	domain tcratc 0.0
	The Currency Rate from reporting currency 1 to the reference currency or given currency.
CurrencyRate3	domain tcratc 0.0
	The Currency Rate from reporting currency 2 to the reference currency or given currency.
CurrencyRateFactor1	domain tcratf 0
	The Currency Rate Factor from the local currency to the reference currency or given currency.
CurrencyRateFactor2	domain tcratf 0
	The Currency Rate Factor from reporting currency 1 to the reference currency or given currency.
CurrencyRateFactor3	domain tcratf 0
	The Currency Rate Factor from reporting currency 2 to the reference currency or given currency.
CurrencyRateDate	domain tcdatc 0
	The Rate Date of the given Currency Rates and Factors.
CurrencyRateType	domain tcrtyp ""
	The Rate Type of the given Currency Rates and Factors.
DeliveryDate	domain tcdatc 0
	The Delivery Date to find a default price for.
OrderDate	domain tcdatc 0
	The Order Date to find a default price for.
CostType	domain tsmdm.cotp tsmdm.cotp.material

	The Cost Type to find a default price for. Contract domain tcorno "" The Contract to find a default price for. ContractChange domain tsctm.cchn 0 The Contract Change to find a default price for. ConfigurationLine domain tsmdm.seqn 0 The Contract Configuration Line to find a default price for. CoverageTime domain tsmdm.trdt 0 The Coverage Time to find a default price for. CoverageType domain tsmdm.cctp "" The Coverage Type to find a default price for. CostComponent domain tccpcp "" The Cost Component to find a default price for. Subcontractor domain tccom.bpid "" The Subcontractor to find a default price for. Price domain tcpric 0.0 The original Price of the cost line to find a default price for. PriceUnit domain tccuni "" The original Price Unit of the cost line to find a default price for. PriceUnitConversionFactor domain tcconv 0.0 The original Price Unit Conversion Factor to Inventory Unit of the cost line to find a default price for. PriceMatrixDefinition domain tdpcg.made "" The original Price Matrix Definition of the cost line to find a default price for. PriceMatrixSequence domain tdpcg.prse 0 The original Price Matrix Sequence of the cost line to find a default price for. PriceOrigin domain tstdm.porg tstdm.porg.not.applicable The original Price Origin of the cost line to find a default price for.

	Determining domain tcyesno tcyesno.no The original Determining flag from the cost line. Eligible domain tcyesno tcyesno.no The original Eligible flag from the cost line. * These fields will be defaulted if not provided. Output: oPrice The default Price found. oPriceUnit The default Price Unit found. oPriceUnitConversionFactor The default Price Unit Conversion Factor found. oPriceMatrixDefinition The default Price Matrix Definition found. oPriceMatrixSequence The default Price Matrix Sequence found. oPriceOrigin The Price Origin from which the default Price was obtained if any. oPriceStage The default Price Stage found. oDiscount The default Discount found. oDiscountPercentage The array of default Discount Percentages found. oDiscountAmount The array of default Discount Amounts found. oDiscountCode The array of default Discount Codes found. oDiscountMethod The array of default Discount Methods found. oDiscountType The array of default Discount Types found. oDiscountDefinition The array of default Discount Definitions found. oDiscountSequence The array of default Discount Sequences found. oDiscountOrigin The array of default Discount Origins found. oDetermining The default Determining flag found. oEligible The default Eligible flag found. oDerivedItemUsed Whether or not a derived Item was used. oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - No Error <> 0 - Error

Pricing.SimulateTransferPrice

DLL:	tdextpcgapi This function is available from KB2319624 .
Syntax:	<pre> long Pricing.SimulateTransferPrice(domain tcncmp iLogisticCompany, domain tcncmp iFinancialCompany, domain tccom.bpid iBuyFromBusinessPartner, domain tccom.bpid iSoldToBusinessPartner, domain tcccur iCurrency, domain tcitem iItem, domain tccom.bpid iInvoiceFromBusinessPartner, domain tccom.bpid iInvoiceToBusinessPartner, domain tccwoc iFinancialDepartment, domain tcsite iSite, domain tcmpnr iManufacturerPartNumber, domain tcmcs.cmnf iManufacturer, domain tccprg iPriceGroup, domain tcdate iRateDate, domain tcrtyp iRateType, domain tcqstl iQuantity, domain tccuni iQuantityUnit, domain tcconv iQuantityUnitConversionFactor, domain tcyesno iRepairPrice, domain tcyesno iSubcontracted, domain tdpcg.prit iPriceType, domain tcefex.date iPriceDate, ref domain tcpric oPrice, ref domain tccuni oPriceUnit, ref domain tcconv oPriceUnitConversionFactor, ref domain tcpcg.made oMatrixDefinition, ref domain tcpcg.prse oMatrixSequence, ref domain tdpcg.prbk oPriceBook, ref domain tdgen.porg oPriceOrigin, ref boolean oPriceRetrieved, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl: This public interface simulates transfer price retrieval. Price retrieval is according to standard LN logic, searching through the hierarchical price structure. Pre: NA Post: NA Input: ***** * Mandatory input arguments ***** iLogisticCompany - Logistic Company iFinancialCompany - Financial Company iBuyFromBusinessPartner - Buy-from Business Partner iSoldToBusinessPartner - Sold-to Business Partner iCurrency - Currency iItem - Item iPriceType - Price Type iRepairPrice - Repair Price iSubcontracted - Subcontracted ***** * Optional input arguments; if not filled, defaults are * retrieved </pre>

	<pre> ***** iInvoiceFromBusinessPartner - Invoice-from Business Partner iInvoiceToBusinessPartner - Invoice-to Business Partner iFinancialDepartment - Financial Department iSite - Site iManufacturerPartNumber - Manufacturer Part Number iManufacturer - Manufacturer iPriceGroup - Price group iRateDate - Rate Date; if zero, then Price Date is used. iRateType - Exchange Rate Type iQuantity - Quantity iQuantityUnit - Quantity Unit iQuantityUnitConversionFactor - Quantity Unit Conversion Factor iPriceDate - Price Date; if zero, then current date and time is used Output: ***** * Price Output ***** oPrice - Price oPriceUnit - Price Unit oPriceUnitConversionFactor - Price Unit Conversion Factor oPriceMatrixDefinition - Price Matrix Definition oPriceMatrixSequence - Price Matrix Sequence oPriceBook - Price Book oPriceOrigin - Price Origin oPriceRetrieved - Price Retrieved ***** * Technical Output ***** oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Price could be determined. <> 0 - Error occurred. </pre>

Public Interfaces for PriceBook

The following functions are available:

[PriceBook.StartMultiMain](#)

PriceBook.StartMultiMain

DLL:	tdextpcgapi
	This function is available from KB2285765 .
Syntax:	<pre> long PriceBook.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tdpcg.prbk iPriceBoook, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the Multi-Main session Price Book (tdpcg0611m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend A specific query to be used when zooming to this session; Optional iPriceBoook Price Book; Optional Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

Public Interfaces for PriceBookLine

The following functions are available:

[PriceBookLines.StartOverview](#)

PriceBookLines.StartOverview

DLL:	tdextpcgapi
Syntax:	<pre> long PriceBookLines.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tdpcg.prbk iPriceBook, domain tcitem iItem, domain tccom.bpid iBuyFromBusinessPartner, domain tccom.bpid iShipFromBusinessPartner, domain tcccur iCurrency, domain tccuni iQuantityUnit, domain tcqsl1 iBreakQuantityValue, domain tdpcg.prit iPriceType, domain tcefex.date iEffectiveDate, domain tcmcs.long iPriceBookLine, ref domain tdpcg.prbk oPriceBook, ref domain tcitem oItem, ref domain tccom.bpid oBuyFromBusinessPartner, ref domain tccom.bpid oShipFromBusinessPartner, ref domain tcccur oCurrency, ref domain tccuni oQuantityUnit, ref domain tcqsl1 oBreakQuantityValue, ref domain tdpcg.prit oPriceType, ref domain tcefex.date oEffectiveDate, ref domain tcmcs.long oPriceBookLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Price Book Lines Overview (tdpcg0131m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byPriceBook": Data is displayed by price book. The session will be started on table-index 1 view fields: Price Book "byItem": Data is displayed by item. The session will be started on table-index 2 view fields: Item iSessionIndex Specifies the table-index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session.

	iPriceBook Price Book Mandatory if iStartMode is MODELESS and session-index 1 or 3 is used. (Note that session index 1 will automatically be used if iStartFilter equals "byPriceBook") iItem Item Mandatory if iStartMode is MODELESS and session-index 2 is used. (Note that session index 2 will automatically be used if iStartFilter equals "byItem") iBuyFromBusinessPartner Buy-From Business Partner Not mandatory. iShipFromBusinessPartner Ship-From Business Partner Not mandatory. iCurrency The price book currency Not mandatory. iQuantityUnit The unit in which the item's quantity is expressed. Not mandatory. iBreakQuantityValue The minimum or maximum order quantity or value, in inventory units, to which the price book applies. Not mandatory. iPriceType The price type for which the price book line is applicable. Allowed values * empty * Not Applicable * Buying - Used for all non-subcontracting purchasing scenarios * Item Subcontracting - Used for item subcontracting * Operation Subcontracting - Used for operation subcontracting * Service Subcontracting - Used for service subcontracting Not mandatory. iEffectiveDate The date and time from which the price book is valid. Not mandatory. iPriceBookLine Price Book Line Not mandatory. Output: for iStartMode MODAL: oPriceBook The selected Price Book oItem Item of the selected Price Book Line oBuyFromBusinessPartner Buy-From Business Partner of the selected Price Book Line oShipFromBusinessPartner Ship-From Business Partner of the selected Price Book Line oCurrency Currency of the selected Price Book Line oQuantityUnit Quantity Unit of the selected Price Book Line oBreakQuantityValue Break Quantity/Value of the selected Price Book Line oPriceType Price Type of the selected Price Book Line

	oEffectiveDate	Effective Date of the selected Price Book Line
	oPriceBookLine	Price Book Line number of the selected Price Book Line
	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
Return: 0		- Session started
<> 0		- An error occurred

Public Interfaces for SalesPriceMatrices

The following functions are available:

[SalesPriceMatrices.StartOverview](#)

SalesPriceMatrices.StartOverview

DLL:	tdextpcgapi
	This function is available from KB2285765 .
Syntax:	<pre> long SalesPriceMatrices.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tdpcg.maty iMatrixType, domain tdpcg.malv iMatrixLevel, domain tdpcg.made iMatrixDefinition, domain tdpcg.prse iSequence, ref domain tdpcg.maty oMatrixType, ref domain tdpcg.malv oMatrixLevel, ref domain tdpcg.made oMatrixDefinition, ref domain tdpcg.prse oSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Sales Price Matrices Overview (tdpcg0130m010). Input: iStartMode Specifies the start mode for the session.

	Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used iSessionIndex Specifies the table-index that is to be used. Allowed values: 0 or 1. Both will start the session with the primary index. iQueryExtend A specific query to be used when zooming to this session. iMatrixType Matrix Type Mandatory. Should always be tdpdg.maty.sobook iMatrixLevel Matrix Level; Optional iMatrixDefinition Matrix Definition; Optional iSequence Sequence; Optional Output: for iStartMode MODAL: oMatrixType The selected Matrix Type oMatrixLevel The selected Matrix Level oMatrixDefinition The selected Matrix Definition oSequence The selected Sequence oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - An error occurred

Chapter 14 Public Interfaces for Landed Costs

Public Interfaces for LandedCosts

The following functions are available:

[LandedCosts.GetSettings](#)

LandedCosts.GetSettings

DLL:	tcextlctapi
Syntax:	<pre> long LandedCosts.GetSettings(domain tcncmp iLogisticCompany, domain tcsite iSite, domain tccwoc iPurchaseOffice, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves Landed Costs settings. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tclct000), or site or office level (settings table tclct080) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when landed costs search date and recalculate landed costs are needed, the function must be called as follows: <pre> if LandedCosts.GetSettings(</pre>

	<pre> /* Fixed arguments: company, --> input site, --> input purchase office, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output /* Variable arguments: "srdt", --> input landed.cost.search.date, --> output "rcal", --> input recalculate.landed.costs) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif </pre>
Pre:	None
Post:	None
Input:	<pre> iLogisticCompany - Logistic Company: Mandatory iSite - Site: Not Mandatory iPurchaseOffice - Purchase Office: Not Mandatory iForceRead - Option to force new query in stead of using cached information iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. </pre>
Output:	<pre> oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. </pre>
Return:	<pre> 0 - Data read <> 0 - Otherwise. </pre>

Chapter 15 Public Interfaces for Material Pricing

Public Interfaces for MaterialPricing

The following functions are available:

[MaterialPricing.GetPurchaseSettings](#)

[MaterialPricing.GetSalesSettings](#)

MaterialPricing.GetPurchaseSettings

DLL:	tcextmprapi
Syntax:	<pre> long MaterialPricing.GetPurchaseSettings(domain tcncmp iLogisticCompany, domain tcsite iSite, domain tccwoc iPurchaseOffice, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves Material Pricing settings for Procurement. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tcmpr000), or site or office level (settings table tcmpr080) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when search date and material pricing in procurement are needed, the function must be called as follows:

	<pre> if MaterialPricing.GetPurchaseSettings(/* Fixed arguments: company, --> input site, --> input purchase office, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output /* Variable arguments: "sctp", --> input search.date, --> output "mprp", --> input material.pricing.in.procurement) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif </pre>
Pre:	None
Post:	None
Input:	<pre> iLogisticCompany - Logistic Company: Mandatory iSite - Site: Not Mandatory iPurchaseOffice - Purchase Office: Not Mandatory iForceRead - Option to force new query in stead of using cached information iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. </pre>
Output:	<pre> oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. </pre>
Return:	<pre> 0 - Data read <> 0 - Otherwise. </pre>

MaterialPricing.GetSalesSettings

DLL:	tcextmprapi
Syntax:	<pre> long MaterialPricing.GetSalesSettings(domain tcncmp iLogisticCompany, </pre>

		domain tccwoc iSalesOffice, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...)												
Usage:	Expl:	This function retrieves Material Pricing settings for Sales. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tcmpr000), or site or office level (settings table tcmpr081) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when search date and material pricing in sales are needed, the function must be called as follows: <pre>if MaterialPricing.GetSalesSettings(* Fixed arguments: company, --> input site, --> input purchase office, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output * Variable arguments: "sdts", --> input search.date, --> output "mprs", --> input material.pricing.in.sales) <> 0 then --> output * Error, do something Exception.Delete(exception.id) endif</pre> Pre: None Post: None Input: <table><tr><td>iLogisticCompany</td><td>- Logistic Company: Mandatory</td></tr><tr><td>iSalesOffice</td><td>- Sales Office: Not Mandatory</td></tr><tr><td>iSite</td><td>- Site: Not Mandatory</td></tr><tr><td>iForceRead</td><td>- Option to force new query in stead of using cached information</td></tr><tr><td>iContextIsMasterData</td><td>- True/False: Indicates if call is done to retrieve data for master data or transactional data.</td></tr><tr><td>...</td><td>- The field mnemonic of the required field.</td></tr></table>	iLogisticCompany	- Logistic Company: Mandatory	iSalesOffice	- Sales Office: Not Mandatory	iSite	- Site: Not Mandatory	iForceRead	- Option to force new query in stead of using cached information	iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.	...	- The field mnemonic of the required field.
iLogisticCompany	- Logistic Company: Mandatory													
iSalesOffice	- Sales Office: Not Mandatory													
iSite	- Site: Not Mandatory													
iForceRead	- Option to force new query in stead of using cached information													
iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.													
...	- The field mnemonic of the required field.													

	Output: oExceptionMessage oExceptionID ... Return: 0 <> 0	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - The value of the required field. - Data read - Otherwise.

Chapter 16 Public Interfaces for Planning

Public Interfaces for PlanItem

The following functions are available:

[PlanItem.StartATPHandling](#)

PlanItem.StartATPHandling

DLL:	cpextrrpapi This function is available from KB2303683 .
Syntax:	<pre> long PlanItem.StartATPHandling(long iStartMode, domain cpcom.plnc iPlanningScenario, domain cpitem iPlanItem, domain tcncmp iCompany, domain tcdate iDeliveryDate, domain tcqsl1 iQuantity, domain tccuni iUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface starts the session ATP Handling (cprp4800m000). Pre: - Post: - Input: iStartMode Mandatory - Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS_ALWAYS - Parent and child are parallel sessions that can be manipulated simultaneously. iPlanningScenario Mandatory - The Planning Scenario. iPlanItem Mandatory - Plan Item. iCompany Mandatory - Company to deliver from. iDeliveryDate Mandatory - Delivery Date.

	iQuantity iUnit Output: oExceptionMessage oExceptionID Return: 0 <> 0	Mandatory - Required Quantity. Mandatory - Unit of Quantity. The last message if any message is found. If more than one message is found, these are present in the oExceptionID. An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Session started. Otherwise.

Public Interfaces for ItemMasterPlan

The following functions are available:

[ItemMasterPlan.Generate](#)

[ItemMasterPlan.StartPlan](#)

[ItemMasterPlan.Update](#)

ItemMasterPlan.Generate

DLL:	cpextmpapi This function is available from KB2309900 .		
Syntax:	<pre>long ItemMasterPlan.Generate(domain cpcom.plnc iScenario, domain cpitem iPlanItem, long iPeriodFrom, long iPeriodTo, boolean iGenerateWithinTimeFence, boolean iUpdateGoodsFlow, boolean iUpdateExceptionMessages, boolean iUpdateResourceMasterPlan, boolean iConsiderCapacityConstraints, boolean iConsiderMaterialConstraints, long iWorkLoadControlIterations, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>		
Usage:	Expl: This function generates a Master Planning for the given Scenario and Plan Item in the requested period. Retry point and commit / abort transaction are		

	executed within this Public Interface. Pre: N.A. Post: N.A. Input: iScenario Scenario (Mandatory). iPlanItem Plan Item (Mandatory). iPeriodFrom Period as defined in Scenario (Mandatory). iPeriodTo Period as defined in Scenario (Mandatory). iGenerateWithinTimeFence Control to allow generation also within the defined Time Fence. iUpdateGoodsFlow Control to update the goods flow data in the master plan before generating. iUpdateExceptionMessages Control to update the exception messages for the Plan Item involved. iUpdateResourceMasterPlan Control to update the Resource Master Plan. iConsiderCapacityConstraints When using Workload Control, whether or not to consider capacity constraints must be considered. Can only be set when workload control is implemented (cprpd000.iwlc). iConsiderMaterialConstraints When using Workload Control, whether or not to consider material constraints must be considered. Can only be set when workload control is implemented (cprpd000.iwlc). iWorkLoadControlIterations When using Workload Control, the maximum number of iterations used to optimize the schedule. only used when workload control is implemented (cprpd000.iwlc). Output: oExceptionMessage The last message if any message is found. If more than one message is found, these are present in the oExceptionID oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Function completed successfully. <> 0 Error occurred.

ItemMasterPlan.StartPlan

DLL:	cpextmpapi
Syntax:	<pre>long ItemMasterPlan.StartPlan(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(),</pre>

	domain cpcom.plnc iPlanningScenario, domain cpitem iPlanItem, domain cpcom.sern iPeriodNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function starts the session Item Master Plan (cprmp2101m000) Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex The index that will be used. Supported values: 1: sort by Scenario/Plan Item/Period 2: sort by Plan Item/Scenario/Period iQueryExtend Not Used. iPlanningScenario Planning Scenario iPlanItem Plan Item iPeriodNumber Period Number Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 Otherwise.

ItemMasterPlan.Update

DLL:	cpexttrmpapi
Syntax:	long ItemMasterPlan.Update(domain tcncmp iCompany, domain cpcom.plnc iPlanningScenario, domain cpitem iPlanItem, domain tcmcs.chan iChannel, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: Update the item master plan data for the given Planning Scenario and Plan Item. Pre: Retry point must be set. Post: Abort or commit the transaction. Input: iCompany Company (mandatory). iPlanningScenario Scenario (mandatory). iPlanItem Plan Item (mandatory).

	iChannel	Channel (optional).
Output:	oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.
	oExceptionID	An ID that refers to all error information. Use the functions in Exception to get all relevant information.
Return:	0	Function has completed.
	<> 0	A fatal problem occurred.

Public Interfaces for ItemOrderPlan

The following functions are available:

[ItemOrderPlan.Get](#)

[ItemOrderPlan.StartPlan](#)

[ItemOrderPlan.StartPlanForProject](#)

[ItemOrderPlan.StartPlanForProjectPeg](#)

ItemOrderPlan.Get

DLL:	cpextrrpapi		
Syntax:	<pre> long ItemOrderPlan.Get(domain cpcom.plnc iPlanningScenario, domain cpitem iPlanItem, domain cprrp.perl iPeriodLength, boolean iShowOrderDistribution, ref long oNumberOfRows, ref domain cporno oOrderNumber() fixed, ref domain tckoor oOrderType(), ref domain tcpono oLinePositionNumber(), ref domain tcpono oSequenceNumber(), ref domain tcpono oDistributionLine(), ref domain tckotr oTransactionType(), ref domain cpitem oPlannedItem() fixed, ref domain tcdate oFinishDate(), ref domain tcqivl oScheduledReceipt(), ref domain tcqivl oPlannedDemand(), ref domain tcqivl oActualDemand(), </pre>		

	ref domain tcqivl oUnconsumedForecast(), ref domain tcqivl oOnHand(), ref domain tcqivl oNetRequirement(), ref domain tcqivl oPlannedAvailable(), ref domain tcqivl oSafetyStockLevel(), ref domain tcqivl oProductionPlan(), ref domain tcqivl oPurchasePlan(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)																																																
Usage:	Expl: This function calculates the Item Order Plan for the specific Scenario and Plan Item and returns the results in arrays. The results are detailed or grouped per day/week/2weeks/4weeks. Limitations: This function does not support Project Pegging. For a single order, retrieved plan does not include the derived from items. Pre: Technical: Output arrays must be defined as based arrays. Post: Technical: Execute free.mem() for the output arrays (after usage of the data). Input: <table> <tr> <td>iPlanningScenario</td><td>Planning Scenario (Mandatory).</td></tr> <tr> <td>iPlanItem</td><td>Plan Item (Mandatory).</td></tr> <tr> <td>iPeriodLength</td><td>Possible values: (Mandatory). 1day (= grouped in 1 day periods), 7days (= grouped in 1 week periods), 14days (= grouped in 2 week periods), 28days (= grouped in 4 week periods), detail (= not grouped), plan (= grouped in Plan Periods),</td></tr> <tr> <td>iShowOrderDistribution</td><td>If true and period length is set to 'Detail', then the output will be on the most detailed level (order distribution). Else output is aggregated on Order Position level. Values: true, false</td></tr> </table> Output: <table> <tr> <td>oNumberOfRows</td><td>Number of item orders.</td></tr> <tr> <td>oOrderNumber</td><td>Numbers array.</td></tr> <tr> <td>oOrderType</td><td>Type array.</td></tr> <tr> <td>oPositionNumber</td><td>Position Number array.</td></tr> <tr> <td>oSequenceNumber</td><td>Sequence number array.</td></tr> <tr> <td>oDistributionLine</td><td>Distribution line position array.</td></tr> <tr> <td>oTransactionType</td><td>Transaction type array.</td></tr> <tr> <td>oPlannedItem</td><td>Planned Item or Plan Item materials.</td></tr> <tr> <td>oFinishDate</td><td>Finish date.</td></tr> <tr> <td>oScheduledReceipt</td><td>Scheduled receipt array.</td></tr> <tr> <td>oPlannedDemand</td><td>Planned demand array.</td></tr> <tr> <td>oActualDemand</td><td>Actual demand array.</td></tr> <tr> <td>oUnconsumedForecast</td><td>Forecast array.</td></tr> <tr> <td>oOnHand</td><td>On hand array.</td></tr> <tr> <td>oNetRequirement</td><td>Net requirement array.</td></tr> <tr> <td>oPlannedAvailable</td><td>Planned availability array.</td></tr> <tr> <td>oSafetyStockLevel</td><td>Safety stock level array.</td></tr> <tr> <td>oProductionPlan</td><td>Production plan array.</td></tr> <tr> <td>oPurchasePlan</td><td>Purchase plan array.</td></tr> <tr> <td>oExceptionMessage</td><td>The last message if the return value is</td></tr> </table>	iPlanningScenario	Planning Scenario (Mandatory).	iPlanItem	Plan Item (Mandatory).	iPeriodLength	Possible values: (Mandatory). 1day (= grouped in 1 day periods), 7days (= grouped in 1 week periods), 14days (= grouped in 2 week periods), 28days (= grouped in 4 week periods), detail (= not grouped), plan (= grouped in Plan Periods),	iShowOrderDistribution	If true and period length is set to 'Detail', then the output will be on the most detailed level (order distribution). Else output is aggregated on Order Position level. Values: true, false	oNumberOfRows	Number of item orders.	oOrderNumber	Numbers array.	oOrderType	Type array.	oPositionNumber	Position Number array.	oSequenceNumber	Sequence number array.	oDistributionLine	Distribution line position array.	oTransactionType	Transaction type array.	oPlannedItem	Planned Item or Plan Item materials.	oFinishDate	Finish date.	oScheduledReceipt	Scheduled receipt array.	oPlannedDemand	Planned demand array.	oActualDemand	Actual demand array.	oUnconsumedForecast	Forecast array.	oOnHand	On hand array.	oNetRequirement	Net requirement array.	oPlannedAvailable	Planned availability array.	oSafetyStockLevel	Safety stock level array.	oProductionPlan	Production plan array.	oPurchasePlan	Purchase plan array.	oExceptionMessage	The last message if the return value is
iPlanningScenario	Planning Scenario (Mandatory).																																																
iPlanItem	Plan Item (Mandatory).																																																
iPeriodLength	Possible values: (Mandatory). 1day (= grouped in 1 day periods), 7days (= grouped in 1 week periods), 14days (= grouped in 2 week periods), 28days (= grouped in 4 week periods), detail (= not grouped), plan (= grouped in Plan Periods),																																																
iShowOrderDistribution	If true and period length is set to 'Detail', then the output will be on the most detailed level (order distribution). Else output is aggregated on Order Position level. Values: true, false																																																
oNumberOfRows	Number of item orders.																																																
oOrderNumber	Numbers array.																																																
oOrderType	Type array.																																																
oPositionNumber	Position Number array.																																																
oSequenceNumber	Sequence number array.																																																
oDistributionLine	Distribution line position array.																																																
oTransactionType	Transaction type array.																																																
oPlannedItem	Planned Item or Plan Item materials.																																																
oFinishDate	Finish date.																																																
oScheduledReceipt	Scheduled receipt array.																																																
oPlannedDemand	Planned demand array.																																																
oActualDemand	Actual demand array.																																																
oUnconsumedForecast	Forecast array.																																																
oOnHand	On hand array.																																																
oNetRequirement	Net requirement array.																																																
oPlannedAvailable	Planned availability array.																																																
oSafetyStockLevel	Safety stock level array.																																																
oProductionPlan	Production plan array.																																																
oPurchasePlan	Purchase plan array.																																																
oExceptionMessage	The last message if the return value is																																																

	not equal to 0. If more than one message is given, these are present in the oExceptionID.
oExceptionID	An ID that refers to all error information. Use the functions in Exception to get all relevant information.
Return: 0 <> 0	Function is executed successfully An error occurred

ItemOrderPlan.StartPlan

DLL:	cpextrrpapi																								
Syntax:	<pre> long ItemOrderPlan.StartPlan(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain cpcom.plnc iPlanningScenario, domain cpitem iPlanItem, domain cprrp.perl iPeriodLength, domain cpcom.date iStartDate, domain tcyesno iSkipEmptyDays, domain tcyesno iShowDistribution, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>																								
Usage:	Expl: This function starts the Item Order Plan (cprrp0520m000), but only if Planning is implemented. The Item Order Plan is started in the Item View. Input: <table> <tr> <td>iStartMode</td><td>Not used. Session is always started MODELESS (Parent and child are parallel sessions that can be manipulated simultaneously).</td></tr> <tr> <td>iStartFilter</td><td>Not Used.</td></tr> <tr> <td>iSessionIndex</td><td>Not Used.</td></tr> <tr> <td>iQueryExtend</td><td>Not Used</td></tr> <tr> <td>iPlanningScenario</td><td>When empty, the Actual Scenario will be used.</td></tr> <tr> <td>iPlanItem</td><td>Plan Item, Mandatory</td></tr> <tr> <td>iPeriodLength</td><td>When empty, then Detail is used</td></tr> <tr> <td>iStartDate</td><td>When zero, then current date is used</td></tr> <tr> <td>iSkipEmptyDays</td><td>When empty, then Yes is used</td></tr> <tr> <td>iShowDistribution</td><td>When empty, then Yes is used</td></tr> </table> Output: No information of selected records is returned. <table> <tr> <td>oExceptionMessage</td><td>- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID</td></tr> <tr> <td>oExceptionID</td><td>- An ID that refers to all error</td></tr> </table>	iStartMode	Not used. Session is always started MODELESS (Parent and child are parallel sessions that can be manipulated simultaneously).	iStartFilter	Not Used.	iSessionIndex	Not Used.	iQueryExtend	Not Used	iPlanningScenario	When empty, the Actual Scenario will be used.	iPlanItem	Plan Item, Mandatory	iPeriodLength	When empty, then Detail is used	iStartDate	When zero, then current date is used	iSkipEmptyDays	When empty, then Yes is used	iShowDistribution	When empty, then Yes is used	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID	oExceptionID	- An ID that refers to all error
iStartMode	Not used. Session is always started MODELESS (Parent and child are parallel sessions that can be manipulated simultaneously).																								
iStartFilter	Not Used.																								
iSessionIndex	Not Used.																								
iQueryExtend	Not Used																								
iPlanningScenario	When empty, the Actual Scenario will be used.																								
iPlanItem	Plan Item, Mandatory																								
iPeriodLength	When empty, then Detail is used																								
iStartDate	When zero, then current date is used																								
iSkipEmptyDays	When empty, then Yes is used																								
iShowDistribution	When empty, then Yes is used																								
oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID																								
oExceptionID	- An ID that refers to all error																								

	information. Use the functions in Exception to get all relevant information.
Return: 0 <> 0	Session started Otherwise.

ItemOrderPlan.StartPlanForProject

DLL:	cpextrrpapi
Syntax:	<pre> long ItemOrderPlan.StartPlanForProject(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain cpcom.plnc iPlanningScenario, domain cpitem iPlanItem, domain tccprj iProject, domain cprrp.perl iPeriodLength, domain cpcom.date iStartDate, domain tcyesno iSkipEmptyDays, domain tcyesno iShowDistribution, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the Item Order Plan (cprrp0520m000), but only if Planning and if Project Pegging is implemented. The Item Order Plan is started in the Item - Project View. Input: iStartMode Not used. Session is always started MODELESS (Parent and child are parallel sessions that can be manipulated simultaneously). iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend Not Used iPlanningScenario When empty, the Actual Scenario will be used. iPlanItem Plan Item, Mandatory iProject Project (used in Project Pegging) Mandatory iPeriodLength When empty, then Detail is used iStartDate When zero, then current date is used iSkipEmptyDays When empty, then Yes is used iShowDistribution When empty, then Yes is used Output: No information of selected records is returned. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error

	information. Use the functions in Exception to get all relevant information.
Return: 0 <> 0	Session started Otherwise.

ItemOrderPlan.StartPlanForProjectPeg

DLL:	cpextrrpapi																										
Syntax:	<pre> long ItemOrderPlan.StartPlanForProjectPeg(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain cpcom.plnc iPlanningScenario, domain cpitem iPlanItem, domain tccprj iProject, domain tccspa iElement, domain tccact iActivity, domain cprrp.perl iPeriodLength, domain cpcom.date iStartDate, domain tcyesno iSkipEmptyDays, domain tcyesno iShowDistribution, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>																										
Usage:	Expl: This function starts the Item Order Plan (cprrp0520m000), but only if Planning and if Project Pegging is implemented. The Item Order Plan is started in the Item - Project Peg View. Input: <table> <tr> <td>iStartMode</td><td>Not used. Session is always started MODELESS (Parent and child are parallel sessions that can be manipulated simultaneously).</td></tr> <tr> <td>iStartFilter</td><td>Not Used.</td></tr> <tr> <td>iSessionIndex</td><td>Not Used.</td></tr> <tr> <td>iQueryExtend</td><td>Not Used</td></tr> <tr> <td>iPlanningScenario</td><td>When empty, the Actual Scenario will be used.</td></tr> <tr> <td>iPlanItem</td><td>Plan Item, Mandatory</td></tr> <tr> <td>iProject</td><td>Project (used in Project Pegging) Mandatory</td></tr> <tr> <td>iElement</td><td>Mandatory, depending on project defintion</td></tr> <tr> <td>iActivity</td><td>Mandatory, depending on project defintion</td></tr> <tr> <td>iPeriodLength</td><td>When empty, then Detail is used</td></tr> <tr> <td>iStartDate</td><td>When zero, then current date is used</td></tr> <tr> <td>iSkipEmptyDays</td><td>When empty, then Yes is used</td></tr> <tr> <td>iShowDistribution</td><td>When empty, then Yes is used</td></tr> </table> Output: No information of selected records is returned. oExceptionMessage - The last message if the return	iStartMode	Not used. Session is always started MODELESS (Parent and child are parallel sessions that can be manipulated simultaneously).	iStartFilter	Not Used.	iSessionIndex	Not Used.	iQueryExtend	Not Used	iPlanningScenario	When empty, the Actual Scenario will be used.	iPlanItem	Plan Item, Mandatory	iProject	Project (used in Project Pegging) Mandatory	iElement	Mandatory, depending on project defintion	iActivity	Mandatory, depending on project defintion	iPeriodLength	When empty, then Detail is used	iStartDate	When zero, then current date is used	iSkipEmptyDays	When empty, then Yes is used	iShowDistribution	When empty, then Yes is used
iStartMode	Not used. Session is always started MODELESS (Parent and child are parallel sessions that can be manipulated simultaneously).																										
iStartFilter	Not Used.																										
iSessionIndex	Not Used.																										
iQueryExtend	Not Used																										
iPlanningScenario	When empty, the Actual Scenario will be used.																										
iPlanItem	Plan Item, Mandatory																										
iProject	Project (used in Project Pegging) Mandatory																										
iElement	Mandatory, depending on project defintion																										
iActivity	Mandatory, depending on project defintion																										
iPeriodLength	When empty, then Detail is used																										
iStartDate	When zero, then current date is used																										
iSkipEmptyDays	When empty, then Yes is used																										
iShowDistribution	When empty, then Yes is used																										

	value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 Session started Otherwise.

Public Interfaces for PlanItemExceptionMessageTotal

The following functions are available:

[PlanItemExceptionMessageTotals.StartOverview](#)

PlanItemExceptionMessageTotals.StartOverview

DLL:	cpextraoapi
Syntax:	<pre>long PlanItemExceptionMessageTotals.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcemno iPlannerID, domain cpcom.plnc iPlanningScenario, domain cpitem iPlanItem, domain tcdater iReferenceDateFrom, domain tcdater iReferenceDateTo, domain cpcao.prio iPriorityFrom, domain cpcao.prio iPriorityTo, const string iSortBy(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the Exception Messages by Planner overview session (cpcao1120m000). Input: iStartMode: Not used. Session will always be started modeless. MODELESS: Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used.

iSessionIndex	Not Used.
iQueryExtend	Not Used.
iPlannerID	The employee number of the planner, or empty (for unlinked messages)
iPlanningScenario	The Planning Scenario. When empty, the Actual Scenario will be used.
iPlanItem	Filter field. Optional. When empty: all items are shown, when filled: only items matching the filter string are shown. Eg. "CL1PCS000123" will display only cluster CL1 plan items for project PCS000123.
iReferenceDateFrom	Filter field. Optional. Default: 0
iReferenceDateTo	Filter field. Optional. Default: 31-12-9999
iPriorityFrom	Filter field. Optional. Default: 0
iPriorityTo	Filter field. Optional. Default: 100
iSortBy	Possible values: "item" (= plan item), "number" (= number of messages"), "priority" (= highest priority), "date" (= reference date). Optional. When empty, the sort order is undefined.
Output: As cpraol120m000 displays aggregated data, no information about selected records is returned.	
oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
Return: 0 <> 0	Session started Otherwise.

Public Interfaces for PlannedOrder

The following functions are available:

[PlannedOrder.Replan](#)

[PlannedOrders.StartGenerate](#)

[PlannedOrders.StartOverview](#)

	Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS_ALWAYS - Parent and child are parallel sessions that can be manipulated simultaneously. iPlanningScenario The Planning Scenario (Mandatory). iPlanItem Plan Item. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Generate Order Planning (Item) Session started. <> 0 Otherwise.

PlannedOrders.StartOverview

DLL:	cpextrrpapi
Syntax:	<pre>long PlannedOrders.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain cpcom.plnc iPlanningScenario, domain tckoor iOrderType, domain cprrp.orno iPlannedOrder, domain tceemm.clus iPlanningCluster, domain tcsite iSite, domain tcecmno iPlanner, domain tccom.bpid iBuyFromBusinessPartner, domain tcecmno iBuyer, ref domain cpcom.plnc oPlanningScenario, ref domain tckoor oOrderType, ref domain cprrp.orno oPlannedOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the session Planned Orders (cprrp1100m000) in overview mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel

	sessions that can be manipulated simultaneously. Not Used. The index that will be used. Supported values: 1: sort by order type 2: sort by cluster / Site 3: sort by Start Date 4: sort by Required Date 5: sort by Planner / Start Date 6: sort by Planner / Order Type 7: sort by Supplier 8: sort by Buyer A specific query to be used when zooming to this session. Planning Scenario (Mandatory if iStartMode is MODELESS) Order Type (Mandatory if iStartMode is MODELESS and iSessionIndex is 1 or 6). Planned Order Planning Cluster Site Planner Buy-from Business Partner Buyer Output: Variables below contain the values of the selected record. They are only filled if iStartMode is MODAL and 1 record has been selected. - oPlanningScenario Planning Scenario - oOrderType Order Type - oPlannedOrder Planned Order - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 Session started Otherwise.

PlannedOrders.TransferOrderSet

DLL:	cpextrrpapi
Syntax:	<pre>long PlannedOrders.TransferOrderSet(long iStartMode, long iNumberOfOrders, ref domain cprrp.orno iPlannedOrder() fixed, ref domain tckoor iOrderType(), ref domain tcmcs.s999m oExceptionMessage mb,</pre>

	ref	long	oExceptionID)
Usage:	Expl:	This function starts the process session Transfer Order Planning (cppat1210m000). The session will be started to process the planned orders given in the input arrays. If i.number.of.orders is 0, the session will be started without a preselection and the Planned Orders can be selected using the standard selection ranges.	
	Pre:	-	
	Post:	-	
	Input:	iStartMode	Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously.
		iNumberOfOrders	Number of Planned Orders to be transferred.
		iPlannedOrder	Array with the planned orders that will be attempted to be transferred (Mandatory).
		iOrderType	Array with the order types of the orders in the iPlannedOrder array (Mandatory).
	Output:	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
		oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
	Return:	0	Transfer Order Session started.
		<> 0	Otherwise.

Public Interfaces for PlannedProductionOrder

The following functions are available:

[PlannedProductionOrder.Transfer](#)

PlannedProductionOrder.Transfer

DLL:	cpextrrpapi
	This function is available from KB2263675 .
Syntax:	<pre> long PlannedProductionOrder.Transfer(domain cprp.orno iPlannedProductionOrder, domain tcseri iOrderSeries, boolean iTransferText, ref domain tcorno oProductionOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface transfers iPlannedProductionOrder, which must exist in the Actual Scenario. The created Shop Floor Production Order is returned in oProductionOrder. Transaction management is handled by this Public Interface. Pre: - Post: - Input: iPlannedProductionOrder - The Planned Production Order to be transferred (Mandatory). iOrderSeries - The first free number to be used to create the new Production Order (Optional). iTransferText - If true, the text of the Planned Production Order will be transferred. Output: oProductionOrder - The Order number of the created Production Order. Return: 0 Planned Order successfully transferred <> 0 Otherwise.

Public Interfaces for PlannedPeggingRelations

The following functions are available:

[PlannedPeggingRelations.FindDemandForOrder](#)

[PlannedPeggingRelations.FindSupplyForOrder](#)

PlannedPeggingRelations.FindDemandForOrder

DLL:	cpextrrpapi

	This function is available from KB2305358 .																			
Syntax:	<pre>long PlannedPeggingRelations.FindDemandForOrder(domain cpcom.plnc iPlanningScenario, domain tckoor iOrderType, domain cporno iOrderNumber, domain tcpono iOrderLine, domain tcpono iSequence, domain tcpono iMaterialPosition, ref domain tccom.long oParentTransactionArray(), ref domain tcncmp oParentCompanyArray(), ref domain cpcom.plnc oParentScenarioArray() fixed, ref domain tccom.long oChildTransactionArray(), ref domain tcncmp oChildCompanyArray(), ref domain cpcom.plnc oChildScenarioArray() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>																			
Usage:	Expl: This public interface returns the chain of planned demand for a specified supply order, line / position, sequence. The returned chain of planned demand is the same as the information that is shown in the Downstream view of the Order Pegging session (cprrp0740m200). Limitations: Pegging Relations are volatile, they can change substantially with each new planning run, even if there are no changes for the plan item. Only Items having Order System = Planned can have (planned) orders with planned supply. Distribution Lines are not supported as input Pre: functional: Generate order planning (cprrp1210m000) must have been executed for the plan item of the specified order and its child items, with the option 'Update Pegging Relations'. technical: Output arrays must be defined as based arrays. Post: technical: Execute free.mem() for the output arrays (after usage of the data). Input: <table><tr><td>iPlanningScenario</td><td>Mandatory. The planning scenario that is used for creating the pegging relations (generate order planning).</td></tr><tr><td>iOrderType</td><td>Mandatory. Type of the supply order (E.g. tckoor.act.sfc or tckoor.cp.sfc or tckoor.act.pur or tckoor.cp.pur)</td></tr><tr><td>iOrderNumber</td><td>Mandatory. The order number holding a supply.</td></tr><tr><td>iOrderLine</td><td>Order Line or Position.</td></tr><tr><td>iSequence</td><td>Sequence number.</td></tr><tr><td>iMaterialPosition</td><td>The material position number (for Line number). iOrderLine or iMaterialPosition cannot be filled both.</td></tr></table> Output: <table><tr><td>oParentTransactionArray</td><td>Parent transaction numbers, as used in table Pegging Transactions (cprrp041).</td></tr><tr><td>oParentCompanyArray</td><td>Parent companies (relevant in multi-company scenarios).</td></tr><tr><td>oParentScenarioArray</td><td>Parent scenarios (relevant in multi-company scenarios).</td></tr></table>		iPlanningScenario	Mandatory. The planning scenario that is used for creating the pegging relations (generate order planning).	iOrderType	Mandatory. Type of the supply order (E.g. tckoor.act.sfc or tckoor.cp.sfc or tckoor.act.pur or tckoor.cp.pur)	iOrderNumber	Mandatory. The order number holding a supply.	iOrderLine	Order Line or Position.	iSequence	Sequence number.	iMaterialPosition	The material position number (for Line number). iOrderLine or iMaterialPosition cannot be filled both.	oParentTransactionArray	Parent transaction numbers, as used in table Pegging Transactions (cprrp041).	oParentCompanyArray	Parent companies (relevant in multi-company scenarios).	oParentScenarioArray	Parent scenarios (relevant in multi-company scenarios).
iPlanningScenario	Mandatory. The planning scenario that is used for creating the pegging relations (generate order planning).																			
iOrderType	Mandatory. Type of the supply order (E.g. tckoor.act.sfc or tckoor.cp.sfc or tckoor.act.pur or tckoor.cp.pur)																			
iOrderNumber	Mandatory. The order number holding a supply.																			
iOrderLine	Order Line or Position.																			
iSequence	Sequence number.																			
iMaterialPosition	The material position number (for Line number). iOrderLine or iMaterialPosition cannot be filled both.																			
oParentTransactionArray	Parent transaction numbers, as used in table Pegging Transactions (cprrp041).																			
oParentCompanyArray	Parent companies (relevant in multi-company scenarios).																			
oParentScenarioArray	Parent scenarios (relevant in multi-company scenarios).																			

	oChildTransactionArray Child transaction numbers, as used in table Pegging Transactions (cprp041). oChildCompanyArray Child companies (relevant in multi-company scenarios). oChildScenarioArray Child scenarios (relevant in multi-company scenarios). Example: A purchase order for item A is used to supply 2 production Orders to produce items B and C. These items B and C are produced to supply two sales orders. Pegging Browser: Purchase order for item A (40) Production order for item B (50) Sales order for item C (60) Production order for item B (70) Sales order for item C (80) (N) = transaction number Resulting Parent and Child Transaction arrays when executing the public interface for the sls order: <table> <tr> <th>oParentTransactionArray</th><th>oChildTransactionArray</th></tr> <tr> <td>0</td><td>40</td></tr> <tr> <td>40</td><td>50</td></tr> <tr> <td>50</td><td>60</td></tr> <tr> <td>40</td><td>70</td></tr> <tr> <td>70</td><td>80</td></tr> </table> The transaction id's in the arrays refer to table cprp041 (Pegging Transactions), where the details (order type, number, position, sequence, demand or supply) of each transaction are stored. oExceptionMessage The last message if any message is found. If more than one message is found, these are present in the oExceptionID oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Public Interface succesfully completed <> 0 Otherwise.	oParentTransactionArray	oChildTransactionArray	0	40	40	50	50	60	40	70	70	80
oParentTransactionArray	oChildTransactionArray												
0	40												
40	50												
50	60												
40	70												
70	80												

PlannedPeggingRelations.FindSupplyForOrder

DLL:	cpextrrpapi
Syntax:	long PlannedPeggingRelations.FindSupplyForOrder(domain cpcom.plnc iPlanningScenario,

	domain tckoor iOrderType, domain cporno iOrderOrigin, domain tcpono iOrderLine, domain tcpono iSequence, domain tcpono iMaterialPosition, ref domain tccom.long oParentTransactionArray(), ref domain tcncmp oParentCompanyArray(), ref domain cpcom.plnc oParentScenarioArray() fixed, ref domain tccom.long oChildTransactionArray(), ref domain tcncmp oChildCompanyArray(), ref domain cpcom.plnc oChildScenarioArray() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This public interface returns the chain of planned supply for a specified demand order, line / position, sequence. The returned chain of planned supply is the same as the information that is shown in the Upstream view of the Order Pegging session (cprrp0740m200). Limitations: Pegging Relations are volatile, they can change substantially with each new planning run, even if there are no changes for the plan item. Only Items having Order System = Planned can have (planned) orders with planned supply. Distribution Lines are not supported as input Pre: functional: Generate order planning (cprrp1210m000) must have been executed for the plan item of the specified order and its child items, with the option 'Update Pegging Relations'. technical: Output arrays must be defined as based arrays. Post: technical: Execute free.mem() for the output arrays (after usage of the data). Input: iPlanningScenario Mandatory. The planning scenario that is used for creating the pegging relations (generate order planning). iOrderOrigin Mandatory. Type of the demand order (E.g. tckoor.act.sls or tckoor.act.sfc) iOrderNumber Mandatory. The order number holding a demand. iOrderLine Order Line or Position. iSequence Sequence number. iMaterialPosition The material position number (for Job Shop orders). iOrderLine or iMaterialPosition cannot be filled both. Output: oParentTransactionArray Parent transaction numbers, as used in table Pegging Transactions (cprrp041). oParentCompanyArray Parent companies (relevant in multi-company scenarios). oParentScenarioArray Parent scenarios (relevant in multi-company scenarios). oChildTransactionArray Child transaction numbers, as used in table Pegging Transactions (cprrp041). oChildCompanyArray Child companies (relevant in multi-company scenarios).

	<code>oChildScenarioArray</code> Child scenarios (relevant in multi-company scenarios). Example: A sales order for item A is supplied by 2 production orders. Both production orders are supplied for a component by one purchase order. Pegging Browser: <pre> sls order for item A (10) prod order for item A (11) pur order for component B (12) prod order for item A (13) pur order for component B (12) </pre> (N) = transaction number Resulting Parent and Child Transaction arrays when executing the public interface for the sls order: <table> <thead> <tr> <th><code>oParentTransactionArray</code></th><th><code>oChildTransactionArray</code></th></tr> </thead> <tbody> <tr><td>0</td><td>10</td></tr> <tr><td>10</td><td>11</td></tr> <tr><td>11</td><td>12</td></tr> <tr><td>10</td><td>13</td></tr> <tr><td>13</td><td>12</td></tr> </tbody> </table> The transaction id's in the arrays refer to table <code>cprrp041</code> (Pegging Transactions), where the details (order type, number, position, sequence, demand or supply) of each transaction are stored. <code>oExceptionMessage</code> - The last message if the return value is not equal to 0. If more than one message is given, these are present in the <code>oExceptionID</code>. <code>oExceptionID</code> - An ID that refers to all error information. Use the functions in <code>Exception</code> to get all relevant information. Return: 0 Public Interface successfully completed <> 0 Otherwise.	<code>oParentTransactionArray</code>	<code>oChildTransactionArray</code>	0	10	10	11	11	12	10	13	13	12
<code>oParentTransactionArray</code>	<code>oChildTransactionArray</code>												
0	10												
10	11												
11	12												
10	13												
13	12												

Public Interfaces for ProductAvailability

The following functions are available:

[ProductAvailability.GetWhenAvailableSchedule](#)

ProductAvailability.GetWhenAvailableSchedule

DLL:	cpextmpapi																																				
	This function is available from KB3625733 .																																				
Syntax:	<pre>long ProductAvailability.GetWhenAvailableSchedule(domain tcplnc iScenario, domain tcitem iItem, domain tcncmp iCompany, domain tcqstl iQuantity, domain tccuni iQuantityUnit, long oProcessingOptionSet, ref long oNumberOfScheduleLines, ref domain tcqsl1 oAvailabilitySchedule(), ref domain tcdate oWhenAvailableSchedule(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>																																				
Usage:	Expl: This Public Interface can be used to return the list of quantities and dates detailing when certain quantities of a given plan item are planned to be in a given warehouse. Returns the same data as the When Available form command on session cprrp4800m000. Pre: - Post: - Input: <table><tr><td>iScenario</td><td>- Planning Scenarion. Mandatory.</td></tr><tr><td>iItem</td><td>- Item. Mandatory</td></tr><tr><td>iCompany</td><td>- Ordering Company. Mandatory.</td></tr><tr><td>iQuantity</td><td>- Ordered Quantity. Mandatory.</td></tr><tr><td>iQuantityUnit</td><td>- Ordered Quantity Unit. Mandatory.</td></tr><tr><td>iProcessingOptionSet</td><td>- Processing Option Set (Optional). If 0, the default printing options are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcextxtapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete()</td></tr></table> Processing Options have a direct relationship with the form fields on session "ATP Handling" (cprrp4800m000) and are not explained in further detail here. Please refer to the session help for additional information. Print options which are not available as Processing Options will get defaulted in accordance with the session logic. Processing Options which are set while a required Implemented Software Component is not available are ignored. All the warning messages that are generated will be stored in the oExceptionID. oExceptionMessage will have the last generated warning message and oExceptionID will have all the generated warning messages in XML structure. <table><tr><th>NAME</th><th>TYPE</th><th>DEFAULT</th></tr><tr><td>Channel</td><td>domain tcmcs.chan</td><td>""</td></tr><tr><td>ProductVariant</td><td>domain tccpva</td><td>0</td></tr><tr><td>EffectiveUnit</td><td>domain tcuef.effn</td><td>0</td></tr><tr><td>Warehouse</td><td>domain tccwar</td><td>""</td></tr><tr><td>DeliveryDate</td><td>domain tcdate</td><td>Current date</td></tr><tr><td>OrderType</td><td>domain tckoor</td><td>tckoor.not.appl</td></tr><tr><td>OrderNumber</td><td>domain cporno</td><td>""</td></tr></table>	iScenario	- Planning Scenarion. Mandatory.	iItem	- Item. Mandatory	iCompany	- Ordering Company. Mandatory.	iQuantity	- Ordered Quantity. Mandatory.	iQuantityUnit	- Ordered Quantity Unit. Mandatory.	iProcessingOptionSet	- Processing Option Set (Optional). If 0, the default printing options are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcextxtapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete()	NAME	TYPE	DEFAULT	Channel	domain tcmcs.chan	""	ProductVariant	domain tccpva	0	EffectiveUnit	domain tcuef.effn	0	Warehouse	domain tccwar	""	DeliveryDate	domain tcdate	Current date	OrderType	domain tckoor	tckoor.not.appl	OrderNumber	domain cporno	""
iScenario	- Planning Scenarion. Mandatory.																																				
iItem	- Item. Mandatory																																				
iCompany	- Ordering Company. Mandatory.																																				
iQuantity	- Ordered Quantity. Mandatory.																																				
iQuantityUnit	- Ordered Quantity Unit. Mandatory.																																				
iProcessingOptionSet	- Processing Option Set (Optional). If 0, the default printing options are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcextxtapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete()																																				
NAME	TYPE	DEFAULT																																			
Channel	domain tcmcs.chan	""																																			
ProductVariant	domain tccpva	0																																			
EffectiveUnit	domain tcuef.effn	0																																			
Warehouse	domain tccwar	""																																			
DeliveryDate	domain tcdate	Current date																																			
OrderType	domain tckoor	tckoor.not.appl																																			
OrderNumber	domain cporno	""																																			

OrderTransactionType	domain	tckotr	tckotr.requirement.
OrderLinePositionNumber	domain	tcpono	0
OrderLineSequenceNumber	domain	tcpono	0
LotSize	domain	tdqsl1	0
CheckFamily	domain	tcyesno	tcyesno.yes
CheckCapacity	domain	tcyesno	tcyesno.yes
CheckComponent	domain	tcyesno	tcyesno.yes
CheckChannel	domain	tcyesno	tcyesno.yes
DemandPeggingType	domain	tcpgtp	tcpgtp.not.appl
SoldToBusinessPartner	domain	tccom.bpid	""
ShipToBusinessPartner	domain	tccom.bpid	""
BusinessObjectType	domain	tcalbt	tcalbt.not.appl
BusinessObject	domain	tcboid	""
BusinessObjectReference	domain	tcborf	""
Reference	domain	tcrefa	""
UseUnallocatedInventory	domain	tcyesno	tcyesno.yes
ProjectPegProject	domain	tccprj	""
ProjectPegElement	domain	tccspa	""
ProjectPegActivity	domain	tccact	""
Output: oNumberOfScheduleLines	- The number of schedule lines.		
oAvailabilitySchedule	- An array of length oNumberOfScheduleLines. Every entry denotes the quantity of the Plan Item that becomes available at the related date from oWhenAvailableSchedule. Quantities are not cumulative. Must be declared based.		
oWhenAvailableSchedule	- An array of length oNumberOfScheduleLines. Every entry denotes the date at which the related quantity from oAvailabilitySchedule is available. Must be declared based.		
oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.		
oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.		
Return: 0	- OK.		
<> 0	- Otherwise.		

Public Interfaces for OrderPegging

The following functions are available:

[OrderPegging.StartSelect](#)

OrderPegging.StartSelect

DLL:	cpextrrpapi																																
	This function is available from KB3531509 .																																
Syntax:	<pre>long OrderPegging.StartSelect (long iStartMode, domain cpcom.plnc iScenario, domain cpstream iStreamingMethod, domain tckoor iOrderType, domain cporno iOrderNumber, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>																																
Usage:	Expl: This Public Interface starts the process session Browse Order Pegging (cprp0740m000). This function makes use of a Processing Option Set, which can be created via a call to ProcessingOptionSet.Create(), and cleaned up after use, via a call to ProcessingOptionSet.Delete(). Pre: N.A. Post: N.A. Input: iStartMode Specifies the start mode for the session.(Mandatory). Possible values are: MODAL The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS_ALWAYS Parent and child are parallel sessions that can be manipulated simultaneously, even if the session is a Dialog. iScenario Planning Scenario (Mandatory). iStreamingMethod Streaming Method. iOrderType Order Type. iOrderNumber Order Number. iProcessingOptionSet A Processing Option Set can be created via a call to ProcessingOptionSet.Create(). If 0, then user default/session default values are applied. Processing Options have a direct relationship with the form fields on session Browse Order Pegging (cprp0740m000) and are not explained in further detail here. Please refer to the session help for additional information. Browse Order Pegging options which are not available as Processing Options will get defaulted in accordance with the session logic. <table><tr><th>NAME</th><th>TYPE</th><th>DEFAULT</th></tr><tr><td>SetEffectivityUnit</td><td>domain tcyesno</td><td>tcyesno.no</td></tr><tr><td>EffectivityUnit</td><td>domain tcuef.effn</td><td>0</td></tr><tr><td>SetPlanner</td><td>domain tcyesno</td><td>tcyesno.no</td></tr><tr><td>Planner</td><td>domain tcemno</td><td>""</td></tr><tr><td>SetPlanItem</td><td>domain tcyesno</td><td>tcyesno.no</td></tr><tr><td>PlanItem</td><td>domain cpitem</td><td>""</td></tr><tr><td>SetWorkCenter</td><td>domain tcyesno</td><td>tcyesno.no</td></tr><tr><td>WorkCenter</td><td>domain tccwoc</td><td>""</td></tr><tr><td>SetProjectPCS</td><td>domain tcyesno</td><td>tcyesno.no</td></tr></table>			NAME	TYPE	DEFAULT	SetEffectivityUnit	domain tcyesno	tcyesno.no	EffectivityUnit	domain tcuef.effn	0	SetPlanner	domain tcyesno	tcyesno.no	Planner	domain tcemno	""	SetPlanItem	domain tcyesno	tcyesno.no	PlanItem	domain cpitem	""	SetWorkCenter	domain tcyesno	tcyesno.no	WorkCenter	domain tccwoc	""	SetProjectPCS	domain tcyesno	tcyesno.no
NAME	TYPE	DEFAULT																															
SetEffectivityUnit	domain tcyesno	tcyesno.no																															
EffectivityUnit	domain tcuef.effn	0																															
SetPlanner	domain tcyesno	tcyesno.no																															
Planner	domain tcemno	""																															
SetPlanItem	domain tcyesno	tcyesno.no																															
PlanItem	domain cpitem	""																															
SetWorkCenter	domain tcyesno	tcyesno.no																															
WorkCenter	domain tccwoc	""																															
SetProjectPCS	domain tcyesno	tcyesno.no																															

ProjectPCS	domain	tccprj	""
SetProductionOrderGroup	domain	tcyesno	tcyesno.no
ProductionOrderGroup	domain	tccprj	""
SetPlannedOrderGroup	domain	tcyesno	tcyesno.no
PlannedOrderGroup	domain	tccprj	""
SetOrderType	domain	tcyesno	tcyesno.no
FromOrderNumber	domain	cporno	iOrderNumber
ToOrderNumber	domain	cporno	"ZZZZZZZZZ"
SetBuyFromBusinessPartner	domain	tcyesno	tcyesno.no
BuyFromBusinessPartner	domain	tccom.bpid	""
ShipFromBusinessPartner	domain	tccom.bpid	""
SetSoldToBP	domain	tcyesno	tcyesno.no
SoldToBP	domain	tccom.bpid	""
ShipToBP	domain	tccom.bpid	""
SetProjectPeg	domain	tcyesno	tcyesno.no
Project	domain	tccprj	""
Element	domain	tccspa	""
Activity	domain	tccact	""
FromDate	domain	tcdate	0
ToDate	domain	tcdate	max value of domain
CriticalBranches	domain	tcyesno	tcyesno.no
CriticalPriority	domain	cpao.prio	0
AnalyzeWhereUsedQuantity	domain	tcyesno	tcyesno.no
ExpandedLevels	domain	tcsrnb	2
PrintProjectPeg	domain	tcyesno	tcyesno.no
PrintOptionList	domain	tcyesno	tcyesno.no
PrintEffectivityUnit	domain	tcyesno	tcyesno.no
PrnitExceptionMessages	domain	tcyesno	tcyesno.no
Output: oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.		
oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.		
Return: 0	Session started succesfully.		
<> 0	Otherwise.		

Chapter 17 Public Interfaces for Standard Costs

Public Interfaces for SimulatedPurchasePrices

The following functions are available:

[SimulatedPurchasePrices.StartOverview](#)

SimulatedPurchasePrices.StartOverview

DLL:	tiextcprapi
Syntax:	<pre> long SimulatedPurchasePrices.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccpcc iCostCalculationCode, domain tcitem iItem, domain tcsite iSite, ref domain tccpcc oCostCalculationCode, ref domain tcitem oItem, ref domain tcsite oSite, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Simulated Purchase Prices in overview mode (ticpr1170m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex The index that will be used. Supported values: 1: sort by Cost Calculation Code/Item/Site iQueryExtend A specific query to be used when zooming to this session.

	<pre> iCostCalculationCode Cost Calculation Code iItem Item iSite Site Output: Variables below contain the values of the selected record. They are only filled if iStartMode is MODAL and 1 record has been selected. oCostCalculationCode Cost Calculation Code oItem Item oSite Site oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 Otherwise. </pre>

Public Interfaces for CalculatedItemCosts

The following functions are available:

[CalculatedItemCosts.StartMultiMain](#)

[CalculatedItemCosts.StartOverview](#)

CalculatedItemCosts.StartMultiMain

DLL:	tiextcprapi
	This function is available from KB2303582 .
Syntax:	<pre> long CalculatedItemCosts.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccpcc iCostCalculationCode, domain tcitem iItem, domain tcemm.grid iEnterpriseUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>

Usage:	Expl: This function starts the session Item Costs by Calculation Code in Multi Main view (ticpr2601m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend A specific query to be used when zooming to this session. iCostCalculationCode Cost Calculation Code iItem Item iEnterpriseUnit Enterprise Unit Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 Otherwise.

CalculatedItemCosts.StartOverview

DLL:	tiextcprapi
	This function is available from KB2303582 .
Syntax:	<pre>long CalculatedItemCosts.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccpcc iCostCalculationCode, domain tcitem iItem, domain tcemm.grid iEnterpriseUnit, ref domain tccpcc oCostCalculationCode, ref domain tcitem oItem, ref domain tcemm.grid oEnterpriseUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the session Item Costs by Calculation Code in overview mode (ticpr2501m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session.

	MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend A specific query to be used when zooming to this session. iCostCalculationCode Cost Calculation Code iItem Item iEnterpriseUnit Enterprise Unit Output: Variables below contain the values of the selected record. They are only filled if iStartMode is MODAL and 1 record has been selected. oCostCalculationCode Cost Calculation Code oItem Item oEnterpriseUnit Enterprise Unit oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 Otherwise.

Public Interfaces for StandardCosts

The following functions are available:

[StandardCosts.Actualize](#)

[StandardCosts.Calculate](#)

[StandardCosts.CalculateAndGetStandardCost](#)

[StandardCosts.CalculateForNewItem](#)

[StandardCosts.CalculateForSimulation](#)

[StandardCosts.CalculateSingleItem](#)

[StandardCosts.PrintMultilevelCostCalculation](#)

StandardCosts.Actualize

DLL:	tiextcprapi																																															
	This function is available from KB3536706 .																																															
Syntax:	long StandardCosts.Actualize(domain tcitemiItem, domain tcecm.gridiEnterpriseUnit, longiProcessingOptionSet, ref domain tcmcs.s999moExceptionMessage mb, ref longoExceptionID)																																															
Usage:	Expl: Actualize and revalue the current calculated standard cost for the given Item. The process behind this function is similar to the execution of the session Actualize Standard Cost and Revalue Inventory (ticpr2220m000). Transaction management is executed within this function, so no pending transactions should be present before calling this function. Pre: N.A. Post: N.A. Input: iItem Item should be standard item and project should be empty. iEnterpriseUnit Enterprise Unit iProcessingOptionSet Processing Option Set (Optional). If 0, then user default/session default values are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcecmapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete(). Processing Options have a direct relationship with the form fields on session Actualize Standard Cost and Revalue Inventory (ticpr2220m000) and are not explained in further detail here. Please refer to the session help for additional information. <table><thead><tr><th>NAME</th><th>TYPE</th><th>DEFAULT</th></tr></thead><tbody><tr><td>EnterpriseUnitFrom</td><td>tcecm.grid (string)</td><td>""</td></tr><tr><td>EnterpriseUnitTo</td><td>tcecm.grid (string)</td><td>"ZZZZZZZZZ"</td></tr><tr><td>OnlyItemsWithoutStandardCost</td><td>tctyesno</td><td>tctyesno.no</td></tr><tr><td>DeleteStandardCostHistoryData</td><td>tctyesno</td><td>tctyesno.no</td></tr><tr><td>EffectiveDate</td><td>tcdte</td><td>current date</td></tr><tr><td>LastDateOfPreviousPeriod</td><td>tctyesno</td><td>tctyesno.no</td></tr><tr><td>ReverseToOriginal</td><td>tctyesno</td><td>tctyesno.no</td></tr><tr><td>PrintHoursValues</td><td>tctyesno</td><td>tctyesno.yes</td></tr><tr><td>PrintReport</td><td>tctyesno</td><td>tctyesno.no</td></tr><tr><td>PrintingDevice</td><td>tcmcs.str14</td><td>""</td></tr><tr><td>PrintingFileoutPathAndName</td><td>tcmcs.str100</td><td>""</td></tr><tr><td>PrintErrorReport</td><td>tctyesno</td><td>tctyesno.no</td></tr><tr><td>PrintingDeviceErrorReport</td><td>tcmcs.str14</td><td>""</td></tr><tr><td>PrintingFileoutPathAndNameErrorReport</td><td>tcmcs.str100</td><td>""</td></tr></tbody></table> Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception			NAME	TYPE	DEFAULT	EnterpriseUnitFrom	tcecm.grid (string)	""	EnterpriseUnitTo	tcecm.grid (string)	"ZZZZZZZZZ"	OnlyItemsWithoutStandardCost	tctyesno	tctyesno.no	DeleteStandardCostHistoryData	tctyesno	tctyesno.no	EffectiveDate	tcdte	current date	LastDateOfPreviousPeriod	tctyesno	tctyesno.no	ReverseToOriginal	tctyesno	tctyesno.no	PrintHoursValues	tctyesno	tctyesno.yes	PrintReport	tctyesno	tctyesno.no	PrintingDevice	tcmcs.str14	""	PrintingFileoutPathAndName	tcmcs.str100	""	PrintErrorReport	tctyesno	tctyesno.no	PrintingDeviceErrorReport	tcmcs.str14	""	PrintingFileoutPathAndNameErrorReport	tcmcs.str100	""
NAME	TYPE	DEFAULT																																														
EnterpriseUnitFrom	tcecm.grid (string)	""																																														
EnterpriseUnitTo	tcecm.grid (string)	"ZZZZZZZZZ"																																														
OnlyItemsWithoutStandardCost	tctyesno	tctyesno.no																																														
DeleteStandardCostHistoryData	tctyesno	tctyesno.no																																														
EffectiveDate	tcdte	current date																																														
LastDateOfPreviousPeriod	tctyesno	tctyesno.no																																														
ReverseToOriginal	tctyesno	tctyesno.no																																														
PrintHoursValues	tctyesno	tctyesno.yes																																														
PrintReport	tctyesno	tctyesno.no																																														
PrintingDevice	tcmcs.str14	""																																														
PrintingFileoutPathAndName	tcmcs.str100	""																																														
PrintErrorReport	tctyesno	tctyesno.no																																														
PrintingDeviceErrorReport	tcmcs.str14	""																																														
PrintingFileoutPathAndNameErrorReport	tcmcs.str100	""																																														

	information. Use the functions in Exception to get all relevant information. Successfully standardized costs. Errors occurred.
Return: 0 <> 0	

StandardCosts.Calculate

DLL:	tiextcprapi
Syntax:	<pre>long StandardCosts.Calculate(domain tcitem iItem, domain tcomm.grid iEnterpriseUnit, domain tcrvdt iWhichDateForRevalue, domain tcdte iSpecificDate, domain tcyesno iDeleteHistory, boolean iRevaluationReport, domain tcmcs.str14 iDevice, domain tcmcs.str1 iLanguage, long iReportOpenMode, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function calculates and actualizes the standard costs and revaluates the inventory (if needed) for the specified item. This function is not allowed and will return an error when: - economic inventory exists for the item and the user does not have permission for session ticpr2225m000 - the item does not have economic inventory and the Calculate and Actualize from Items - Costing Session, as defined in the Standard Costs Parameters (ticpr0100m000), is not allowed - the item is customized and does have economic inventory. The function will calculate and actualize and the standard costs and revalue the inventory if needed: - for standard and customized items, but not for customized items having economic inventory - using the standard price calculation code for standard items and the project's price calculation code for customized items - standard items are calculated single level, customized items are calculated top.down (as in StandardCosts.CalculateForNewItem and as in ticpr0107m000) - optionally a report can be printed to show the revaluation values. Retry points and commit / abort transactions are set and executed within this public interface. Input: iItem - Item, having no economic inventory iEnterpriseUnit - Enterprise Unit (mandatory filled when the concept Standard Cost per EU is active, otherwise empty) iWhichDateForRevalue Date used for evaluation: (Mandatory):

	. current date (tcrvdt.current.date), or . last date of previous period (tcrvdt.last.prev.peri), or . a specific date (tcrvdt.specific.date). The last 2 options are not applicable when the Standard Cost Calculation Parameter Backdate Valuation Price Allowed = No. In that case the current date will be used. Only applicable if iWhichDateForRevaluate has the value 'a specific date'. iSpecificDate Delete history, as defined in the Calculation Parameters (ticpr0100m000). If true and revaluation of inventory is done, then a report with the old and new value of the inventory will be created; else no report. iDeleteHistory Is a device is already opened in the session, then that device will be used. If no device is opened yet, then specify the device using the 3 arguments below. iRevaluationReport The device for the RevaluationReport. Mandatory when iRevaluationReport = True and iReportOpenMode = 0 and no device is opened yet. iDevice Optional. If empty then the user's language will be used. iLanguage Open mode for the Revaluation Report: 0 - User is not prompted to select a device. 1 - User is prompted to select a device. With Close (= cancel) button. 2 - User is prompted to select a device. No cancel button iReportOpenMode Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Return: 0 - Standard Cost calculated and actualized <> 0 - Standard Costs could not be calculated or actualized

StandardCosts.CalculateAndGetStandardCost

DLL:	tiextcprapi
Syntax:	<pre>long StandardCosts.CalculateAndGetStandardCost (domain tccpcc iCalculationCode, domain tcitem iItem, domain tceмм.grid iEnterpriseUnit, domain tcuef.effn iEffectivityUnit, domain tcdate iStandardCostDate,</pre>

	<pre> boolean iUseSpecifiedOrderQuantity, domain tiqepl iOrderQuantity, ref long oNumberOfCurrencies, domain tcccur oCurrencies() fixed, ref long oNumberOfCostComponents, domain tccpcp oCostComponents() fixed, domain tcqnul oCostNumberOfUnits(), domain tccopr oCostTotal(,), domain tccopr oCostFixed(,), domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function calculates standard costs for the given Calculation Code, Item and Enterprise Unit. The calculated standard cost is returned in the output arrays. NOTE: The calculated standard cost is not stored in the standard cost records for the item. Input: <pre> iCalculationCode Calculation Code (Mandatory) iItem Item (Mandatory) iEnterpriseUnit Enterprise Unit (mandatory filled when the concept Standard Cost per EU is active, otherwise empty) iEffectivityUnit Effectivity Unit, use 0 when not used. iStandardCostDate Reference Date for calculation. iUseSpecifiedOrderQuantity When true: Calculate standard cost based on the given order quantity iOrderQuantity Order Quantity </pre> Output: <pre> oNumberOfCurrencies Number of currencies oCurrencies Currencies in CostTotal/CostFixed arrays Array of Home Currencies, allocated by caller. oNumberOfCostComponents Number of cost components in the output arrays. oCostComponents Cost Component arrays (allocated by function) oCostNumberOfUnits Cost number of units for cost component (allocation by function) oCostTotal Cost total for cost component (allocation by function) oCostFixed Fixed Cost for cost component (allocation by function) oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. </pre> Return: 0 Standard Cost calculated and stored in ouput arrays <> 0 failure

StandardCosts.CalculateForNewItem

DLL:	tiextcprapi
Syntax:	<pre> long StandardCosts.CalculateForNewItem(domain tcitem iItem, domain tcomm.grid iEnterpriseUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function calculates and actualizes the standard costs for a new item. This function is not allowed and will return an error when: - Calculate and Actualize from Items - Costing Session, as defined in the Standard Costs Parameters (ticpr0100m000), is not allowed - the item does have economic inventory. The function will calculate and actualize the standard costs using the same calculation logic als the Calculate and Actualize Standard Costs option in session ticpr0107m000: - using the standard price calculation code for standard items and the project's price calculation code for customized items - using the current date as calculation and revaluation date - standard items are calculated single level, customized items are calculated top.down. Retry points and commit / abort transactions are set and executed within this API. Input: - iItem - Item, having no economic inventory - iEnterpriseUnit - Enterprise Unit (mandatory filled when the concept Standard Cost per EU is active, otherwise empty) Output: - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Return: 0 - Standard Cost calculated and actualized <> 0 - Standard Costs could not be calculated or actualized

StandardCosts.CalculateForSimulation

DLL:	tiextcprapi
Syntax:	<pre> long StandardCosts.CalculateForSimulation(domain tcitem iItem, domain tcomm.grid iEnterpriseUnit, </pre>

	<pre> domain tccpcc iCalculationCode, domain tcdate iCalculationDate, domain tcccmnt iMethod, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function calculates the standard costs for the specified item. When standard items are calculated, the calculation code should not be the Standard Cost Calculation Code as defined in Standard Cost Calculation Parameters (ticpr0100m000). Retry points and commit / abort transactions are set and executed within this public interface. Pre: None. Post: None. Input: iItem Item (Mandatory) iEnterpriseUnit Enterprise Unit (Mandatory when the concept Standard Cost per EU is active, otherwise empty). iCalculationCode Calculation Code (Mandatory). iCalculationDate Reference Date for calculation. iMethod - Top Down - Single Level Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Return: 0 Standard Cost calculated for simulation. <> 0 Failure.

StandardCosts.CalculateSingleItem

DLL:	tiextcprapi
	This function is available from KB3537464 .
Syntax:	<pre> long StandardCosts.CalculateSingleItem(domain tccpcc iCalculationCode, domain tcitem iItem, domain tcemm.grid iEnterpriseUnit, domain tcccmnt iCalculationMethod, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface allows for calculation of standard cost starting off of a single item. The calculation may be executed top-down or single-level. When executed top-down, all items and enterprise units in the product structure starting at the top level Item/Enterprise Unit(s) will get addressed.

Pre:	N.A.	
Post:	N.A.	
Input:	iCalculationCode	Cost Calculation Code (Mandatory)
	iItem	Item (Mandatory) Top-level item to calculate. Only supports single item calculations. It is not allowed to calculate the cost for a project item.
	iEnterpriseUnit	Enterprise Unit May only be given when the concept Standard Cost by Enterprise Unit is active. Mandatory for the case, when the Item uses Standard Cost by Enterprise Unit and the Calculation Method is single-level. If left empty, then, by default, standard cost is calculated for all related Enterprise Units. The Enterprise Unit that serves as the starting point for the calculation. Note that in case of top-down calculation, cost for other Enterprise Units will get calculated because of dependencies.
	iCalculationMethod	Calculation Method (Mandatory) Either top-down or single-level
	iProcessingOptionSet	Processing Option Set (Optional). If 0, then user default/session default values are applied. A Processing Option Set can be created via a call to ProcessingOptionSet.Create() in DLL tcextxtapi. After the call the option set can be deleted by calling ProcessingOptionSet.Delete().
Processing Options have a direct relationship with the form fields on session Calculate Standard Cost (ticpr2210m000) and are not explained in further detail here. Please refer to the session help for additional information.		
NAME	TYPE	DEFAULT
EnterpriseUnitFrom	tcemm.grid (string)	""
EnterpriseUnitTo	tcemm.grid (string)	"ZZZZZZZZZ"
CalculationDate	tcdate	Current date
PrintErrorReport	tctyesno	tctyesno.no
PrintingDeviceErrorReport	tcmcs.str14	""
PrintingFileoutPathAndNameErrorReport	tcmcs.str100	""
If PrintErrorReport is tctyesno.yes then the error report will be printed. If it is tctyesno.no then errors will be logged in exception structure.		
Output:		
	oExceptionMessage	The last message, if any message is found. If more than one message is given, these are present in the oExceptionID.
	oExceptionID	An ID that refers to the exception information. Use the functions in Exception to get all relevant information.

	Return: 0 <> 0	Successfully calculated the standard costs. Errors occurred.

StandardCosts.PrintMultilevelCostCalculation

DLL:	tiextcprapi																																																
	This function is available from KB2329136 .																																																
Syntax:	<pre>long StandardCosts.PrintMultilevelCostCalculation(domain tcitem iItem, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>																																																
Usage:	Expl: This Public Interface is used to print Multilevel Cost Calculation report (ticpr2420m000). This function makes use of a Processing Option Set, which can be created via a call to ProcessingOptionSet.Create(), and cleaned up after use, via a call to ProcessingOptionSet.Delete(). Pre: N.A. Post: N.A. Input: iItem - Item for which the standard cost should be calculated (Optional). iProcessingOptionSet - A Processing Option Set can be created via a call to ProcessingOptionSet.Create(). If 0, then user default/session default values are applied (Optional). Processing Options have a direct relationship with the form fields on session Print Multilevel Cost Calculation (ticpr2420m000) and are not explained in further detail here. Please refer to the session help for additional information. Print Multilevel Cost Calculation options which are not available as Processing Options will get defaulted in accordance with the session logic. <table><tr><th>NAME</th><th>TYPE</th><th>DEFAULT</th></tr><tr><td>CostCalculationCode</td><td>domain tccpcc</td><td>standard cost calculation code</td></tr><tr><td>EffectivityUnit</td><td>domain tcuef.effn</td><td>0</td></tr><tr><td>CalculationDate</td><td>domain tiutcs</td><td>current date</td></tr><tr><td>EnterpriseUnitFrom</td><td>domain tcemm.grid</td><td>" "</td></tr><tr><td>EnterpriseUnitTo</td><td>domain tcemm.grid</td><td>"ZZZZZZZZZ"</td></tr><tr><td>Currency</td><td>domain tcccur</td><td>" "</td></tr><tr><td>PredefinedOrderQuantityMainItem</td><td>domain tcyesno</td><td>tcyesno.yes</td></tr><tr><td>UseOrderQuantity</td><td>domain tigepl</td><td>0.0</td></tr><tr><td>PredefinedOrderQuantitySubItems</td><td>domain tcyesno</td><td>tcyesno.yes</td></tr><tr><td>IncludeFixedCosts</td><td>domain tcyesno</td><td>tcyesno.yes</td></tr><tr><td>RecalculateStandardParts</td><td>domain tcyesno</td><td>tcyesno.no</td></tr><tr><td>PrintDetailsByComponent</td><td>domain tcyesno</td><td>tcyesno.no</td></tr><tr><td>PrintHighPrecision</td><td>domain tcyesno</td><td>tcyesno.yes</td></tr><tr><td>PrintingDevice</td><td>domain tcmcs.str14</td><td>" "</td></tr><tr><td>PrintingFileoutPathAndName</td><td>domain tcmcs.str100</td><td>" "</td></tr></table>	NAME	TYPE	DEFAULT	CostCalculationCode	domain tccpcc	standard cost calculation code	EffectivityUnit	domain tcuef.effn	0	CalculationDate	domain tiutcs	current date	EnterpriseUnitFrom	domain tcemm.grid	" "	EnterpriseUnitTo	domain tcemm.grid	"ZZZZZZZZZ"	Currency	domain tcccur	" "	PredefinedOrderQuantityMainItem	domain tcyesno	tcyesno.yes	UseOrderQuantity	domain tigepl	0.0	PredefinedOrderQuantitySubItems	domain tcyesno	tcyesno.yes	IncludeFixedCosts	domain tcyesno	tcyesno.yes	RecalculateStandardParts	domain tcyesno	tcyesno.no	PrintDetailsByComponent	domain tcyesno	tcyesno.no	PrintHighPrecision	domain tcyesno	tcyesno.yes	PrintingDevice	domain tcmcs.str14	" "	PrintingFileoutPathAndName	domain tcmcs.str100	" "
NAME	TYPE	DEFAULT																																															
CostCalculationCode	domain tccpcc	standard cost calculation code																																															
EffectivityUnit	domain tcuef.effn	0																																															
CalculationDate	domain tiutcs	current date																																															
EnterpriseUnitFrom	domain tcemm.grid	" "																																															
EnterpriseUnitTo	domain tcemm.grid	"ZZZZZZZZZ"																																															
Currency	domain tcccur	" "																																															
PredefinedOrderQuantityMainItem	domain tcyesno	tcyesno.yes																																															
UseOrderQuantity	domain tigepl	0.0																																															
PredefinedOrderQuantitySubItems	domain tcyesno	tcyesno.yes																																															
IncludeFixedCosts	domain tcyesno	tcyesno.yes																																															
RecalculateStandardParts	domain tcyesno	tcyesno.no																																															
PrintDetailsByComponent	domain tcyesno	tcyesno.no																																															
PrintHighPrecision	domain tcyesno	tcyesno.yes																																															
PrintingDevice	domain tcmcs.str14	" "																																															
PrintingFileoutPathAndName	domain tcmcs.str100	" "																																															

	Output: oExceptionMessage oExceptionID Return: 0 <> 0	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. - Print Multilevel Cost Calculation is successfully. - Otherwise.

Chapter 18 Public Interfaces for Manufacturing Master

Public Interfaces for ProductionBillOfMaterial

The following functions are available:

[ProductionBillOfMaterial.ApproveRevision](#)

[ProductionBillOfMaterial.CopyToJobShop](#)

[ProductionBillOfMaterial.ExpireRevision](#)

[ProductionBillOfMaterial.GenerateProductSubcontractingModel](#)

[ProductionBillOfMaterial.StartMultiMain](#)

[ProductionBillOfMaterial.ValidateRevision](#)

ProductionBillOfMaterial.ApproveRevision

DLL:	tiextmfcapi
Syntax:	<pre> long ProductionBillOfMaterial.ApproveRevision(domain tcitem iProduct, domain tibmr iRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this public interface to approve the specified Production Bill of Material Revision. In case Electronic Signature is required this function cannot be used. Pre: Multi Site setting 'Job Shop by Site' must be In Preparation or Active. Retry point must be set. Post: Commit or abort the transaction. Input: iProduct - Product ID. (mandatory). iRevision - Revision Number (mandatory). Output: oExceptionMessage - The last message if the return

	value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - PBOM Revision is set to Approved - PBOM Revision could not be Approved Return: 0 <> 0

ProductionBillOfMaterial.CopyToJobShop

DLL:	tiextmfcapi
Syntax:	<pre>long ProductionBillOfMaterial.CopyToJobShop(domain tcitem iProduct, domain tibmrv iRevision, domain tcsite iToSite, domain tcyesno iCreateNewBillOfMaterial, domain tibmdl iBillOfMaterialCode, domain tcseri iSeries, domain tcyesno iResetLogisticFields, domain tcyesno iCopyManualMaterialLines, ref domain tibmdl oGeneratedModel, ref domain tibmrv oGeneratedRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface generates a job shop bill of material from the given source production BOM. Pre: Multi Site setting 'Job Shop by Site' must be In Preparation or Active. Retry point must be set. Post: Commit or abort the transaction. Input: - iProduct - Main item (mandatory). - iRevision - Revision (mandatory). - iToSite - Site (mandatory). - iCreateNewBillOfMaterial - yes means create new BOM model, no is to add a revision to the current BOM. - iBillOfMaterialCode - BOM Model, mandatory if iCreateNewBillOfMaterial is no. - iSeries - BOM Model Series, mandatory if iCreateNewBillOfMaterial is yes. - iResetLogisticFields - yes means logistic fields are reset to the default value, no means that the value is taken from the previous Job Shop BOM revision (this is about fields that are not present in the Production BOM). This applies only when iCreateNewBillOfMaterial is no. - iCopyManualMaterialLines - Copy material lines that were added manually in the previous Job Shop BOM revision (which are not present in the

	Output: oGeneratedModel oGeneratedRevision oExceptionMessage oExceptionID Return: 0 <> 0	Production BOM). This applies only when iCreateNewBillofMaterial is no. - The generated BOM model - The generated BOM revision - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - PBOM copied successfully - PBOM could not be copied

ProductionBillOfMaterial.ExpireRevision

DLL:	tiextmfncapi	
Syntax:	<pre>long ProductionBillOfMaterial.ExpireRevision(domain tcitem iProduct, domain tibmrv iRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: This function expires the specified revision. If the revision is already expired an Exception message will be returned. Pre: Multi Site setting 'Job Shop by Site' must be In Preparation or Active. Retry point has been set. Specified Production Bill of Material is not yet expired. Post: Transaction is aborted or committed. Input: iProduct - Product (Mandatory). iRevision - Revision (Mandatory). Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Specified PBOM Revision has successfully been expired. <> 0 - Specified PBOM Revision has not been expired.	

ProductionBillOfMaterial.GenerateProductSubcontracting Model

DLL:	tiextmfcapi
	This function is available from KB2292734 .
Syntax:	<pre> long ProductionBillOfMaterial.GenerateProductSubcontractingModel(domain tcitem iProduct, domain tibmrv iRevision, domain tcsite iProductionSite, domain tccom.bpid iSubcontractor, domain tcsite iSubcontractorSite, domain tccwar iSubcontractorWarehouse, domain tccom.bpid iShipFromBusinessPartner, ref domain tirpt.revi oGeneratedRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface generates a Product Subcontracting Model from the given source production BOM. Pre: Multi Site setting 'Job Shop by Site' must be In Preparation or Active. Retry point must be set. Post: Commit or Abort the transaction. Input: - iProduct - Main item in Production BOM (Mandatory). - iRevision - Revision in Production BOM (Mandatory). - iProductionSite - Production Site Subcontracting Model (Mandatory). - iSubcontractor - The subcontractor. - iSubcontractorSite - Subcontractor Site (Mandatory if Resource by Site is active). - iSubcontractorWarehouse - Subcontractor Warehouse (Mandatory). - iShipFromBusinessPartner - Ship From Business Partner (Mandatory if Resource by Site is not active, derived from iSubcontractorSite otherwise). Output: - oGeneratedRevision - The generated Product Subcontracting Model Revision. - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Product Subcontracting Model Revision successfully generated. <> 0 - Generate failed.

ProductionBillOfMaterial.StartMultiMain

DLL:	tiextmfapi
Syntax:	<pre> long ProductionBillOfMaterial.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iProduct, domain tibmrv iRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session Production Bill of Material (timfc3600m000) pre: The session can only be started if the concept Job Shop by Site is active or in preperation. post: - Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used iSessionIndex Not used iQueryExtend Optional iProduct Optional iRevision Optional Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Otherwise.

ProductionBillOfMaterial.ValidateRevision

DLL:	tiextmfapi
Syntax:	<pre> long ProductionBillOfMaterial.ValidateRevision(domain tcitem iProduct, </pre>

	domain tibmrv iRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: Use this public interface to validate the specified Production Bill of Material Revision. Pre: Multi Site setting 'Job Shop by Site' must be In Preparation or Active. Input: iProduct Product ID (Mandatory). iRevision Revision Number (Mandatory). Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 PBOM Revision passed all the validation checks. <> 0 PBOM Revision did not pass all the validation checks.

Public Interfaces for ProductionWarehouseOrders

The following functions are available:

[ProductionWarehouseOrders.StartDetail](#)

ProductionWarehouseOrders.StartDetail

DLL:	tiextmfcapi
	This function is available from KB2286048 .
Syntax:	<pre>long ProductionWarehouseOrders.StartDetail(long iStartMode, domain tiinh.oorg iOrderOrigin, domain tcorno iOrder, domain tcpono iPosition, domain tcpono iSequenceNumber, domain tcinh.ittp iTransactionType, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the detail session Production Warehouse Orders (timfc0101m000). Input: iStartMode

	Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iOrderOrigin Order Origin - Mandatory. The Order Origin must have one of the following values: tiinh.oorg.production - JSC Production tiinh.oorg.product.sched - Production Schedule iOrder Order - Mandatory. iPosition Position iSequenceNumber Sequence Number - Mandatory. iTransactionType Transaction Type - Mandatory. The Transaction Type must have one of the following values: tcinh.ittp.receipt - Receipt tcinh.ittp.issue - Issue tcinh.ittp.transfer - Transfer Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started. <> 0 Otherwise.

Public Interfaces for ProductionItemBySite

The following functions are available:

[ProductionItemBySite.Start360](#)

ProductionItemBySite.Start360

DLL:	tiextmfapi
	This function is available from KB2303494 .
Syntax:	<pre> long ProductionItemBySite.Start360(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcsite iSite, domain tcitem iItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Production Item by Site 360 (timfc1500m100). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used. iSessionIndex Not used. iQueryExtend A specific query to be used when zooming to this session. Optional. iSite Site - Optional. iItem Item - Optional. Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started. <> 0 Error Occurred.

Public Interfaces for AsBuilt

The following functions are available:

[AsBuilt.StartMultiMain](#)

[AsBuilt.StartOverview](#)

AsBuilt.StartMultiMain

DLL:	tiextmfcapi This function is available from KB2304915 .
Syntax:	<pre> long AsBuilt.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain timfc.ord iOrderType, domain tcpdno iOrderNumber, domain tcponl iSchedulePosition, domain tcitem iItem, domain tcibd.sern iSerialNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the Multi Main session As-Built (timfc0610m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used. iSessionIndex Not used iQueryExtend A specific query to be used when zooming to this session. Optional. iOrderType Order Number can be either Production Order or Assembly Order - Optional iOrderNumber Order Number - Optional iSchedulePosition SchedulePosition - Optional iItem Item - Optional iSerialNumber SerialNumber - Optional Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started. <> 0 Error Occurred.

AsBuilt.StartOverview

DLL:	tiextmfapi This function is available from KB2304915 .
Syntax:	<pre> long AsBuilt.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain timfc.ord iOrderType, domain tcpdno iOrderNumber, domain tcponl iSchedulePosition, domain tcitem iItem, domain tcibd.sern iSerialNumber, ref domain timfc.ord oOrderType, ref domain tcpdno oOrderNumber, ref domain tcponl oSchedulePosition, ref domain tcitem oItem, ref domain tcibd.sern oSerialNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Serial End Item - As-Built Headers (timfc0110m000) in overview mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used. iSessionIndex Optional. The index that will be used. Supported values: 1: sort by Order Type, Order, Schedule Line, End Item, Serial Number. 2: sort by Order Type, Order, Schedule Line, Eff Unit, Status, Serial, End Item. 3: sort by End Item, Serial, Order Type, Order, Schedule Line. iQueryExtend A specific query to be used when zooming to this session. Optional. iOrderType Order Type - Optional iOrderNumber Order Number can be either Production Order or Assembly Order - Optional iSchedulePosition Schedule Position - Optional iItem Item - Optional iSerialNumber Serial Number - Optional Output: Variables below contain the values of the selected record. They are only filled if i.start.mode is MODAL and 1 record has been selected. oOrderType Order Type oOrderNumber Order Number can be either Production Order or Assembly Order oSchedulePosition Schedule Position oItem Item

	oSerialNumber oExceptionMessage oExceptionID Return: 0 <> 0 Serial Number The last message if any message is found. If more than one message is given, these are present in the oExceptionID. An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Session started. Otherwise.

Chapter 19 Public Interfaces for Job Shop

Public Interfaces for BillOfMaterialLines

The following functions are available:

[BillOfMaterialLines.StartOverview](#)

BillOfMaterialLines.StartOverview

DLL:	tiextbomapi
Syntax:	<pre> long BillOfMaterialLines.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iMainItem, domain tcpono iPosition, domain tcsern iSequence, ref domain tcitem oMainItem, ref domain tcpono oPosition, ref domain tcsern oSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session Bill of Material (tibom1110m000) Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Optional. Possible values are: "showAll": All BOM lines are displayed "showEffective": Only effective lines are shown (default) When empty: "showEffective" is applied

	<pre> iSessionIndex Not used. iQueryExtend A specific query to be used when zooming to this session. Optional iMainItem Mandatory. iPosition Optional iSequence Optional Output: for iStartMode MODAL: oMainItem - main item of selected line (= iMainItem) oPosition - position of selected line oSequence - Sequence number of selected line oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Otherwise. </pre>

Public Interfaces for ItemRoutingOperations

The following functions are available:

[ItemRoutingOperations.StartOverview](#)

ItemRoutingOperations.StartOverview

DLL:	tiextrouapi
Syntax:	<pre> long ItemRoutingOperations.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iMainItem, domain tirou.opro iRouting, domain tcopno iOperation, domain tirou.sern iSequenceNumber, </pre>

	<pre> ref domain tcitem oMainItem, ref domain tirou.opro oRouting, ref domain tcopno oOperation, ref domain tirou.sern oSequenceNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Routing Operations in overview mode (tirou1102m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Not Used. The session will always start with the first session index: Item, Routing code, Operation, Sequence Number iQueryExtend A specific query to be used when zooming to this session. Primary Key fields: iMainItem Main Item iRouting Routing (Mandatory if iSessionIndex = 1 and iStartMode = MODELESS) iOperation Operation (Mandatory if iSessionIndex = 1 and iStartMode = MODELESS) iSequenceNumber Sequence Number (Mandatory if iSessionIndex = 1 and iStartMode = MODELESS) Output: Variables below contain the values of the selected record. They are only filled if iStartMode is MODAL and 1 record has been selected. oMainItem Main Item oRouting Routing oOperation Operation oSequenceNumber Sequence Number oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 Otherwise.

Public Interfaces for JobShop

The following functions are available:

[JobShop.GetMasterDataSettings](#)

[JobShop.GetOrderSettings](#)

JobShop.GetMasterDataSettings

DLL:	tiextrouapi
Syntax:	<pre> long JobShop.GetMasterDataSettings(domain tcncmp iLogisticCompany, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves master data settings for job shop (table tirou000 / tirou080). One or more setting fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tirou000), or site level (settings table tirou080), depending on: - implementation phase of Job Shop by Site, - input argument iSite, - the specified input field(s) (mnemonic). Note: This function uses variable arguments, for input and output. Data is retrieved via name-value pairs: specify the field (input) and a variable for the field value (output). Example: when the settings 'default routing' (dfprt) and 'use production rate' (uprr) are needed, the function must be called with the following arguments: <pre> JobShop.GetMasterDataSettings(* Fixed arguments: company, --> input site, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output * Variable arguments: "dfprt", --> input default.routing, --> output "uprr", --> input use.production.rates) --> output </pre> Pre: None Post: None

	Input: - iLogisticCompany - Logistic Company: Mandatory - iSite - Site: Not Mandatory - iForceRead - Option to force reading new data in stead of using cached information - iContextIsMasterData - True/False (not used) ... - The field mnemonic of the required field(s) Output: - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field(s). Return: 0 - Data read <> 0 - Otherwise.

JobShop.GetOrderSettings

DLL:	tiextsfapi
Syntax:	<pre>long JobShop.GetOrderSettings(domain tcncmp iLogisticCompany, domain tcnsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...)</pre>
Usage:	Expl: This function retrieves settings for job shop orders (table tisfc000 / tisfc080). One or more settings fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tisfc000), or site level (settings table tisfc080), depending on: - implementation phase of Multi Site, - input argument iSite, - the specified input field(s) (mnemonic). Note: This function uses variable arguments, for input and output. Data is retrieved via name-value pairs: specify the field (input) and a variable for the field value (output). Example: when the settings 'production order series for ep' (psep) and 'step size for renumbering operations' (svro) are needed, the function must be called with the following arguments: <pre>JobShop.GetOrderSettings(</pre>

	<pre> * Fixed arguments: company, --> input site, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output * Variable arguments: "psep", --> input prod.ord.series.ep, --> output "svro", --> input step.size.renum.ops) --> output </pre>
Pre:	None
Post:	None
Input:	<pre> iLogisticCompany - Logistic Company: Mandatory iSite - Site: Not Mandatory iForceRead - Option to force reading new data in stead of using cached information iContextIsMasterData - True/False (not used) ... - The field mnemonic of the required field(s) </pre>
Output:	<pre> oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field(s). </pre>
Return:	<pre> 0 - Data read <> 0 - Otherwise. </pre>

Public Interfaces for JobShopBillOfMaterial

The following functions are available:

[JobShopBillOfMaterial.ApproveRevision](#)

[JobShopBillOfMaterial.Copy](#)

[JobShopBillOfMaterial.CreateNewRevision](#)

[JobShopBillOfMaterial.ExpireRevision](#)

[JobShopBillOfMaterial.Explode](#)

[JobShopBillOfMaterial.RemoveUseUp](#)

[JobShopBillOfMaterial.SetUseForCosting](#)[JobShopBillOfMaterial.SetUseForPlanning](#)[JobShopBillOfMaterial.StartMultiMain](#)[JobShopBillOfMaterial.StartOverview](#)[JobShopBillOfMaterial.ValidateRevision](#)

JobShopBillOfMaterial.ApproveRevision

DLL:	tiextbomapi												
Syntax:	<pre> long JobShopBillOfMaterial.ApproveRevision(domain tcsite iSite, domain tcitem iProduct, domain tibmdl iBillOfMaterialCode, domain tibmrv iRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>												
Usage:	Expl: Use this public interface to approve the specified Job Shop Bill of Material Revision. In case Electronic Signature is required, this function cannot be not be used for standard items; it can then only be used for customized items. Pre: Job Shop by Site must be In Preparation or Active. Retry point must be set. Post: Commit or abort the transaction. Input: <table> <tr><td>iSite</td><td>Site (mandatory).</td></tr> <tr><td>iProduct</td><td>Main item (mandatory).</td></tr> <tr><td>iBillOfMaterialCode</td><td>BOM Model (mandatory).</td></tr> <tr><td>iRevision</td><td>Revision (mandatory).</td></tr> </table> Output: <table> <tr> <td>oExceptionMessage</td> <td>- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID</td> </tr> <tr> <td>oExceptionID</td> <td>- An ID that refers to all error information. Use the functions in Exception to get all relevant information.</td> </tr> </table> Return: 0 <> 0 - JS BOM Revision is set to Approved - JS BOM Revision could not be Approved	iSite	Site (mandatory).	iProduct	Main item (mandatory).	iBillOfMaterialCode	BOM Model (mandatory).	iRevision	Revision (mandatory).	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
iSite	Site (mandatory).												
iProduct	Main item (mandatory).												
iBillOfMaterialCode	BOM Model (mandatory).												
iRevision	Revision (mandatory).												
oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID												
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.												

JobShopBillOfMaterial.Copy

DLL:	tiextbomapi
Syntax:	<pre> long JobShopBillOfMaterial.Copy(domain tcsite iSourceSite, domain tcitem iSourceProduct, domain tibmdl iSourceBillOfMaterialCode, domain tibmrv iSourceRevision, domain tcsite iTargetSite, domain tcitem iTargetProduct, domain tibmdl iTargetBillOfMaterialCode, domain tibmrv iTargetRevision, boolean iCopyExceptions, boolean iCopyReferenceDesignators, boolean iCopyMaterialsRoutingRelationships, boolean iCopyBomText, boolean iCopyPhantomRoutingRelationships, boolean iCopyAlternativeMaterials, ref domain tibmdl oNewBillOfMaterialCode, ref domain tibmrv oNewRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this public interface to copy a Job Shop Bill of Material to the target Job Shop Bill of Material where related data is copied based on input options. If the target model code does not exist, the target revision is ignored as a new Model code and revision are created by the system. If target model code does exist but revision number is missing, A new revision is created by the system. Pre: Job Shop by Site must be In Preparation or Active. Retry point has been set. Post: abort or commit the transaction Input: <pre> iSourceSite Source Site (Mandatory). iSourceProduct Source Product (Mandatory). iSourceBillOfMaterialCode Source BOM Model (Mandatory). iSourceRevision Source Revision (Mandatory). iTargetSite Target Site (Mandatory). iTargetProduct Target Product (Mandatory). iTargetBillOfMaterialCode Target Bill Of Material Code (Optional). iTargetRevision Target Revision (Optional). iCopyExceptions Copy Exceptions iCopyReferenceDesignators Copy Reference Designators iCopyMaterialRoutingRelationships Copy Material Routing Relationships iCopyBomText </pre>

	Copy BOM Text iCopyPhantomRoutingRelationships Copy Phantom Routing Relationships iCopyAlternativeMaterials Copy Alternative Materials Output: oNewBillOfMaterialCode New Job Shop BOM model. oNewRevision New Job Shop BOM revision. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Function is executed successfully. <> 0 - An error occurred. The Job Shop Bill of Material could not be copied.

JobShopBillOfMaterial.CreateNewRevision

DLL:	tiextbomapi
Syntax:	<pre>long JobShopBillOfMaterial.CreateNewRevision(domain tcsite iSite, domain tcitem iProduct, domain tibmdl iBillOfMaterialCode, domain tibmrv iRevision, ref domain tibmrv oNewRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this public interface to create a new Revision based on an existing Job Shop Bill of Material. The BOM lines of the original Bill of Material are copied to the new Revision. Pre: Retry point has been set. Post: Abort or commit the transaction. Input: iSite Site (Mandatory). iProduct Product (Mandatory). iBillOfMaterialCode BOM Model (Mandatory). iRevision Revision (Mandatory). Output: oNewRevision Created new Job Shop BOM revision. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Function is executed successfully.

	<> 0 - An error occurred. A new revision could not be created.

JobShopBillOfMaterial.ExpireRevision

DLL:	tiextbomapi
Syntax:	<pre> long JobShopBillOfMaterial.ExpireRevision(domain tcsite iSite, domain tcitem iProduct, domain tibmdl iBillofMaterialCode, domain tibmrv iRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function expires the specified revision. If the revision is already expired or not allowed, a failure message will be returned. It is not allowed to expire a revision if it is used by active Production Orders. Pre: Job Shop by Site must be In Preparation or Active. Retry point has been set. Input JSBOM is not yet expired. Post: Transaction is aborted or committed. Input: iSite Site (Mandatory). iProduct Product (Mandatory). iBillofMaterialCode BOM Model (Mandatory). iRevision Revision (Mandatory). Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Specified JS BOM Revision has successfully been expired. <> 0 - Specified JS BOM Revision has not been expired.

JobShopBillOfMaterial.Explode

DLL:	tiextbomapi
Syntax:	<pre> long JobShopBillOfMaterial.Explode(domain tcsite iSite, domain tcitem iProduct, </pre>

	<pre> domain tibmdl iBillOfMaterialCode, domain tibmr iRevision, domain tcuef.effn iEffectivityUnit, domain tcdate iReferenceDate, long iExplosionDepth, boolean iMergeComponents, boolean iApplyScrapYield, boolean iExplodeLowestPhantoms, ref domain tccprj oProject() fixed, ref domain tccwar oWarehouse() fixed, ref domain tcitem oItem() fixed, ref domain tcyesno oCustomizable(), ref domain tcpono oPositionNumber(), ref domain tcqivl oBomQuantity(), ref long oNumberOfMaterials, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this Public Interface to get "exploded" Bill of Material information in output arrays. The BOM tree is searched for: - direct non-phantom children of iProduct - non-phantom children of phantom children (multi-level; maximum number of levels: iExplosionDepth) The found children (= materials) are added to the output arrays. The output arrays will have a length of 3000. In case the concept Job Shop by Site is active, then the Job Shop BOM is searched else the 'classic' BOM is searched. When Job Shop by Site is active, then iSite must be filled and iBillOfMaterialCode / iRevision can be filled optionally. When iBillOfMaterialCode and iRevision are both filled, then the Job Shop BOM is read for the specified Model and Revision, else the 'Use for Planning' Job Shop BOM is read. When Job Shop by Site is not active, then iSite, iBillOfMaterialCode and iRevision are not used. Phantoms on iExplosionDepth are not exploded but are included in the output arrays. But if iExplodeLowestPhantoms = true, then the phantom children on iExplosionDepth are exploded anyway. iExplosionDepth = 0 is handled as if iExplosionDepth = 1. Only BOM lines that are valid on iReferenceDate and that are valid for iEffectivityUnit are read. Other BOM lines are skipped. E.g. if iEffectivityUnit = 0, then BOM lines that are not 'standard configuration' will be skipped. Note that if the concept Job Shop by Site is In Preperation and the Site is empty, a warning instead of an error is given in the returned ExceptionID structure. The function still returns 0 in this case. A warning is also given when the output arrays are completely filled, as that may mean that the output is not complete. Pre: Output arrays must be based. Post: NA

	Input: iSite Site (Mandatory when concept Job Shop by Site is active). iProduct Product (Mandatory). iBillofMaterialCode BOM Model (Optional). iRevision Source Revision (Optional). iEffectivityUnit Effectivity Unit (Optional). iReferenceDate Reference Date for filtering the BOM lines (Mandatory). iExplosionDepth Maximum dept (1 is direct children). iMergeComponents Merge the same components on different levels. iApplyScrapYield Apply defined scrap and yield to determine the quantity. iExplodeLowestPhantoms Continue exploding Phantoms when iExplosionDepth is reached. Output: oProject - Array of projects, must be a based array. oWarehouse - Array of warehouses, must be a based array. oItem - Array of Items, must be a based array. oCustomizable - Array of Customizable flags, must be a based array. oPositionNumber - Array of Position numbers, must be a based array. oBomQuantity - Array of BOM Quantities, must be a based array. oNumberOfMaterials - The number of materials found and filled in the arrays. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error/warning information. Use the functions in Exception to get all relevant information. Return: 0 - Function is executed successfully. <> 0 - An error occurred. The Job Shop Bill of Material could not be exploded.

JobShopBillOfMaterial.RemoveUseUp

DLL:	tiextbomapi
Syntax:	<pre>long JobShopBillOfMaterial.RemoveUseUp(domain tcsite iSite, domain tcitem iProduct, domain tibmdl iBillofMaterialCode, domain tibmrv iRevision,</pre>

		domain	tcqivl	iThresholdQuantity,
	ref	domain	tcmcs.s999m	oExceptionMessage mb,
	ref		long	oExceptionID)
Usage:	Expl:	Use this public interface to remove Use Up Materials that are not valid or not useful anymore from the specified BOM revision.		
		A Use Up Material is not valid if its last allowed order date is after the expiry date of the BOM revision.		
		A Use Up Material is not useful if its on-hand inventory quantity is lower than iThresholdQuantity or zero (when iThresholdQuantity is not specified).		
	Pre:	Job Shop by Site must be In Preparation or Active. Retry point must be set.		
	Post:	Abort or commit the transaction.		
	Input:	iSite	Site (mandatory).	
		iProduct	Main item (mandatory).	
		iBillOfMaterialCode	BOM Model (mandatory).	
		iRevision	Revision (mandatory).	
			The status of the specified BOM revision must be New.	
		iThresholdQuantity	Threshold quantity for inventory check (optional).	
	Output:	oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the	
			oExceptionID	
		oExceptionID	An ID that refers to all error information. Use the functions in Exception handling to get all relevant information.	
	Return:	0	- no use up materials were found, or - all found use up materials are valid and usefull, or - all invalid or not useful use up materials were succesfully removed.	
		<> 0	An invalid or not useful material could not be removed.	

JobShopBillOfMaterial.SetUseForCosting

DLL:	tiextbomapi
Syntax:	<pre> long JobShopBillOfMaterial.SetUseForCosting(domain tcsite iSite, domain tcitem iProduct, domain tibmdl iBillOfMaterialCode, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function sets the Use For Costing flag to true for a Job

	Shop Bill of Material. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iSite - Site (Mandatory). iProduct - Product (Mandatory). iBillOfMaterialCode - Bill Of Material Model (Mandatory). Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Use for Costing set to true. <> 0 - Otherwise.

JobShopBillOfMaterial.SetUseForPlanning

DLL:	tiextbomapi
Syntax:	<pre>long JobShopBillOfMaterial.SetUseForPlanning(domain tcsite iSite, domain tcitem iProduct, domain tibmdl iBillOfMaterialCode, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function sets the Use For Planning flag to true for a Job Shop Bill of Material. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iSite - Site (Mandatory). iProduct - Product (Mandatory). iBillOfMaterialCode - Bill Of Material Model (Mandatory). Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Use for Planning set to true. <> 0 - Otherwise.

JobShopBillOfMaterial.StartMultiMain

DLL:	tiextbomapi
Syntax:	<pre> long JobShopBillOfMaterial.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcsite iSite, domain tcitem iProduct, domain tibmdl iBillOfMaterialCode, domain tibmrsv iRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session Job Shop Bill of Material (tibom3600m000). The session can only be started when: 1. the concept job shop by site is active or in preparation, and 2. a job shop bill of material exists for the specified site / product / bill of material / revision Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used iSessionIndex Not used iQueryExtend Optional iSite Optional iProduct Optional iBillOfMaterialCode Optional iRevision Optional Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Otherwise.

JobShopBillOfMaterial.StartOverview

DLL:	tiextbomapi
Syntax:	<pre> long JobShopBillOfMaterial.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcsite iSite, domain tcitem iProduct, domain tibmdl iBillOfMaterialCode, domain tibmrv iRevision, domain tcsite oSite, domain tcitem oProduct, domain tibmdl oBillOfMaterialCode, domain tibmrv oRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session Job Shop Bill of Material (tibom3600m000). The session can only be started when: 1. the concept job shop by site is active or in preparation, and 2. a job shop bill of material exists for the specified site / product / bill of material / revision / query extend Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Optional. Possible values are: "showAll": All BOM lines are displayed "showEffective": Only effective lines are shown (default) "showCurrent": Only lines whiche are Approved an Effective When empty: "showEffective" is applied iSessionIndex Not used iQueryExtend A specific query to be used when starting the session. Using the query extend may lead to a "data not found, session not started" situation. Optional iSite Mandatory if i.mode = MODELESS, else optional iProduct Optional iBillOfMaterialCode Optional iRevision Optional Output: if iStartMode MODAL and 1 record is selected oSite oProduct oBillOfMaterialCode

	oRevision	
	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
	Return: 0	- Session started
	<> 0	- Otherwise.

JobShopBillOfMaterial.ValidateRevision

DLL:	tiextbomapi
Syntax:	<pre> long JobShopBillOfMaterial.ValidateRevision(domain tcsite iSite, domain tcitem iProduct, domain tibmdl iBillOfMaterialCode, domain tibmrv iRevision, domain tcyesno iCheckUseUp, domain tcqivl iThresholdQuantity, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this public interface to validate the specified Job Shop Bill of Material Revision. Input: iSite Site (mandatory). iProduct Main item (mandatory). iBillOfMaterialCode BOM Model (mandatory). iRevision Revision (mandatory). iCheckUseUp If yes, then the function checks if the definitions of the Use Up Items are: - valid (last allowed order date vs expiry date of the BOM) and - useful (depending on the Inventory level of the Use Up Item). If not valid or not useful, then an error is returned and if iRemoveUseUp = Yes, then the Use Up item will be removed. iThresholdQuantity If the on-hand Inventory quantity of a Use Up item is below iThresholdQuantity, (or equal in case iThresholdQuantity = 0.0), then the Use Up item is considered as not useful. Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error

	<pre> Return: 0 <> 0 </pre>	information. Use the functions in Exception to get all relevant information. JS BOM Revision passed all the validation checks. JS BOM Revision did not pass all the validation checks.

Public Interfaces for JobShopRouting

The following functions are available:

[JobShopRouting.ApproveRevision](#)

[JobShopRouting.ClearUseForPlanning](#)

[JobShopRouting.Copy](#)

[JobShopRouting.CopyOperations](#)

[JobShopRouting.CreateNewRevision](#)

[JobShopRouting.ExpireRevision](#)

[JobShopRouting.SetUseForCosting](#)

[JobShopRouting.SetUseForPlanning](#)

[JobShopRouting.StartMultiMain](#)

JobShopRouting.ApproveRevision

DLL:	tiextrouapi
Syntax:	<pre> long JobShopRouting.ApproveRevision(domain tcsite iSite, domain tcitem iProduct, domain tirou.rouc iRouting, domain tirou.revi iRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this public interface to approve the specified Job Shop Routing Revision. Both Standard and non-Standard Routings can be approved.

	In case Electronic Signature is required, this function cannot be not be used for standard items; it can then only be used for customized items. Pre: Job Shop by Site must be In Preparation or Active. Retry point must be set. Post: Commit or abort the transaction. Input: iSite Site (mandatory). iProduct Product. iRouting Routing (mandatory). iRevision Revision (mandatory). Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - JS Routing Revision is Approved <> 0 - JS Routing Revision could not be Approved

JobShopRouting.ClearUseForPlanning

DLL:	tiextrouapi
Syntax:	<pre>long JobShopRouting.ClearUseForPlanning(domain tcsite iSite, domain tcitem iProduct, domain tirou.rouc iJobShopRouting, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function sets the Use For Planning flag to false for a Job Shop Routing. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iSite - Site (Mandatory). iProduct - Product (Mandatory). iJobShopRouting - Job Shop Routing (Mandatory). Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Use for Planning set to false. <> 0 - Otherwise.

JobShopRouting.Copy

DLL:	tiextrouapi																																								
Syntax:	<pre> long JobShopRouting.Copy(domain tcsite iSourceSite, domain tcitem iSourceProduct, domain tirou.rouc iSourceRouting, domain tirou.revi iSourceRevision, domain tcitem iTargetProduct, domain tirou.rouc iTargetRouting, domain tirou.revi iTargetRoutingRevision, domain tcdsca iTargetRoutingRevisionDescription mb, boolean iCopyRelations, boolean iCopyOperationSteps, boolean iCopyTools, boolean iCopySkills, boolean iCopyProcessVariables, boolean iCopySetup, boolean iCopyInstructions, boolean iCopyExceptions, boolean iCopySetupClassStates, boolean iCopyText, ref domain tirou.revi oNewRoutingRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>																																								
Usage:	Expl: Use this public interface to copy a Job Shop Routing to the target Job Shop Routing where related data is copied based on input options. If the target revision number is empty, a new revision is created based on the existing revisions. Pre: Job Shop by Site must be In Preparation or Active. Retry point has been set. Post: Abort or commit the transaction. Input: <table> <tr><td>iSourceSite</td><td>Original Site (Mandatory).</td></tr> <tr><td>iSourceProduct</td><td>Original Product (Mandatory).</td></tr> <tr><td>iSourceRouting</td><td>Original Job Shop Routing (Mandatory).</td></tr> <tr><td>iSourceRevision</td><td>Original Revision (Mandatory).</td></tr> <tr><td>iTargetProduct</td><td>Target Product.</td></tr> <tr><td>iTargetRouting</td><td>Target Job Shop Routing.</td></tr> <tr><td>iTargetRoutingRevision</td><td>Target Revision.</td></tr> <tr><td>iTargetRoutingRevisionDescription</td><td>Target Job Shop Routing Description.</td></tr> <tr><td>iCopyRelations</td><td>Copy Relations</td></tr> <tr><td>iCopyOperationSteps</td><td>Copy Operation Steps</td></tr> <tr><td>iCopyTools</td><td>Copy Tools</td></tr> <tr><td>iCopySkills</td><td>Copy Skills</td></tr> <tr><td>iCopyProcessVariables</td><td>Copy Process Variables</td></tr> <tr><td>iCopySetup</td><td>Copy Setup</td></tr> <tr><td>iCopyInstructions</td><td>Copy Instructions</td></tr> <tr><td>iCopyExceptions</td><td>Copy Exceptions</td></tr> <tr><td>iCopySetupClassStates</td><td>Copy Setup Class States</td></tr> <tr><td>iCopyText</td><td>Copy Text</td></tr> </table> Output: <table> <tr><td>oNewRoutingRevision</td><td>New Job Shop Routing Revision.</td></tr> <tr><td>oExceptionMessage</td><td>The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.</td></tr> </table>	iSourceSite	Original Site (Mandatory).	iSourceProduct	Original Product (Mandatory).	iSourceRouting	Original Job Shop Routing (Mandatory).	iSourceRevision	Original Revision (Mandatory).	iTargetProduct	Target Product.	iTargetRouting	Target Job Shop Routing.	iTargetRoutingRevision	Target Revision.	iTargetRoutingRevisionDescription	Target Job Shop Routing Description.	iCopyRelations	Copy Relations	iCopyOperationSteps	Copy Operation Steps	iCopyTools	Copy Tools	iCopySkills	Copy Skills	iCopyProcessVariables	Copy Process Variables	iCopySetup	Copy Setup	iCopyInstructions	Copy Instructions	iCopyExceptions	Copy Exceptions	iCopySetupClassStates	Copy Setup Class States	iCopyText	Copy Text	oNewRoutingRevision	New Job Shop Routing Revision.	oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.
iSourceSite	Original Site (Mandatory).																																								
iSourceProduct	Original Product (Mandatory).																																								
iSourceRouting	Original Job Shop Routing (Mandatory).																																								
iSourceRevision	Original Revision (Mandatory).																																								
iTargetProduct	Target Product.																																								
iTargetRouting	Target Job Shop Routing.																																								
iTargetRoutingRevision	Target Revision.																																								
iTargetRoutingRevisionDescription	Target Job Shop Routing Description.																																								
iCopyRelations	Copy Relations																																								
iCopyOperationSteps	Copy Operation Steps																																								
iCopyTools	Copy Tools																																								
iCopySkills	Copy Skills																																								
iCopyProcessVariables	Copy Process Variables																																								
iCopySetup	Copy Setup																																								
iCopyInstructions	Copy Instructions																																								
iCopyExceptions	Copy Exceptions																																								
iCopySetupClassStates	Copy Setup Class States																																								
iCopyText	Copy Text																																								
oNewRoutingRevision	New Job Shop Routing Revision.																																								
oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.																																								

	<code>oExceptionID</code> An ID that refers to all error information. Use the functions in <code>Exception</code> to get all relevant information. Return: 0 Function is executed successfully. <> 0 An error occurred. The Job Shop Routing could not be copied.

JobShopRouting.CopyOperations

DLL:	tiextrouapi
Syntax:	<pre> long JobShopRouting.CopyOperations(domain tcsite iSourceSite, domain tcitem iSourceProduct, domain tirou.rouc iSourceRouting, domain tirou.revi iSourceRevision, domain tcopno iFromOperation, domain tcopno iToOperation, domain tcitem iTargetProduct, domain tirou.rouc iTargetRouting, domain tirou.revi iTargetRevision, boolean iCopyOperationSteps, boolean iCopyTools, boolean iCopySkills, boolean iCopyProcessVariables, boolean iCopySetup, boolean iCopyInstructions, boolean iCopyExceptions, boolean iCopySetupClassStates, boolean iAppend, boolean iCopyText, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this public interface to copy Job Shop Routing Operations of one Routing to another one based on input options. Pre: Job Shop by Site must be In Preparation or Active. Retry point has been set. Post: Abort or commit the transaction. Input: <code>iSourceSite</code> Original Site (Mandatory). <code>iSourceProduct</code> Original Product (Mandatory). <code>iSourceRouting</code> Original Job Shop Routing (Mandatory). <code>iSourceRevision</code> Original Revision (Mandatory). <code>iFromOperation</code> Start Operation range to be copied (Mandatory). <code>iToOperation</code> End Operations to be copied (Mandatory). <code>iTargetProduct</code> Target Product (Mandatory). <code>iTargetRouting</code> Target Job Shop Routing (Mandatory). <code>iTargetRevision</code> Target Revision (Mandatory). <code>iAppend</code> True (append) or False (overwrite). <code>iCopyOperationSteps</code> Copy Operation Steps. <code>iCopyTools</code> Copy Tools. <code>iCopySkills</code> Copy Skills. <code>iCopyProcessVariables</code> Copy Process Variables. <code>iCopySetup</code> Copy Setup.

	<pre> iCopyInstructions Copy Instructions. iCopyExceptions Copy Exceptions. iCopySetupClassStates Copy Setup Class States. iCopyText Copy Text. Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 The Job Shop Routing Operations are succesfully copied. <> 0 An error occurred. The Job Shop Routing Operations could not be copied. </pre>

JobShopRouting.CreateNewRevision

DLL:	tiextrouapi
Syntax:	<pre> long JobShopRouting.CreateNewRevision(domain tcsite iSite, domain tcitem iProduct, domain tirou.rouc iJobShopRouting, domain tirou.revi iJobShopRoutingRevision, ref domain tirou.revi oNewJobShopRoutingRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function creates a new Job Shop Routing revision based on the given revision. The operations in the new revision are copied from the set of operations of the original revision. Prerequisite: 'Job Shop by Site' is 'In Preparation' or 'Active' Example usage: <pre> JobShopRouting.CreateNewRevision(* Fixed arguments: site, --> input product, --> input routing --> input revision --> input new.revision --> output exception.message, --> output exception.id) --> output </pre> Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iSite Site: Mandatory iProduct Product: Mandatory iJobShopRouting Job Shop Routing: Mandatory iJobShopRoutingRevision Job Shop Routing Revision: Mandatory Output: oNewJobShopRoutingRevision

	oExceptionMessage oExceptionID Return: 0 <> 0	Newly created Job Shop Routing Revision The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Success Failure, exception message is present

JobShopRouting.ExpireRevision

DLL:	tiextrouapi
Syntax:	<pre>long JobShopRouting.ExpireRevision(domain tcsite iSite, domain tcitem iProduct, domain tirou.rouc iRouting, domain tirou.revi iRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this public interface to expire the specified Job Shop Routing Revision if the specified revision is not already expired. Both Standard and non-Standard Routings can be expired. Pre: Job Shop by Site must be In Preparation or Active. Retry point must be set. Post: Commit or abort the transaction. Input: iSite Site (mandatory). iProduct Product. iRouting Routing (mandatory). iRevision Revision (mandatory). Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - JS Routing Revision is Expired. <> 0 - JS Routing Revision could not be Expired.

JobShopRouting.SetUseForCosting

DLL:	tiextrouapi
Syntax:	<pre> long JobShopRouting.SetUseForCosting(domain tcsite iSite, domain tcitem iProduct, domain tirou.rouc iJobShopRouting, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function sets the Use For Costing flag to true for a Job Shop Routing. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: - iSite - Site (Mandatory). - iProduct - Product (Mandatory). - iJobShopRouting - Job Shop Routing (Mandatory). Output: - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Use for Costing set to true. <> 0 - Otherwise.

JobShopRouting.SetUseForPlanning

DLL:	tiextrouapi
Syntax:	<pre> long JobShopRouting.SetUseForPlanning(domain tcsite iSite, domain tcitem iProduct, domain tirou.rouc iJobShopRouting, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function sets the Use For Planning flag to true for a Job Shop Routing. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: - iSite - Site (Mandatory). - iProduct - Product (Mandatory). - iJobShopRouting - Job Shop Routing (Mandatory). Output: - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to all error information. Use the functions in

	Return: 0 <> 0 Exception to get all relevant information. - Use for Planning set to true. - Otherwise.

JobShopRouting.StartMultiMain

DLL:	tiextrouapi
Syntax:	<pre>long JobShopRouting.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcsite iSite, domain tcitem iProduct, domain tirou.rouc iJobShopRouting, domain tirou.revi iJobShopRoutingRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the session Job Shop Routing (tirou4600m000). Note that the session can only be started if the parameter 'Job Shop by Site' is 'In Preparation' or 'Active'. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iQueryExtend A specific query to be used when zooming to this session. Primary Key fields: iSite Site (Mandatory) iProduct Product (Mandatory) iJobShopRouting Job Shop Routing (Mandatory) iJobShopRoutingRevision Job Shop Routing Revision (Mandatory) Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 Session started Otherwise.

Public Interfaces for ItemRouting

The following functions are available:

[ItemRoutings.StartOverview](#)

ItemRoutings.StartOverview

DLL:	tiextrouapi
	This function is available from KB2307645 .
Syntax:	<pre> long ItemRoutings.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iMainItem, domain tirou.opro iRouting, ref domain tcitem oMainItem, ref domain tirou.opro oRouting, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Item Routings in overview mode (tirou101m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex 1 - The session will start with the first session index: Manufactured Item, Routing. 2 - The session will start with the second session index: Manufactured Item, Up to Order Quantity, Routing. 3 - The session will start with the third session index: Standard Routing, Manufactured Item, Routing. iQueryExtend A specific query to be used when zooming to this session. Primary Key fields: iMainItem Main Item - Optional iRouting Routing - Optional Output: for iStartMode MODAL:

	oMainItem	Main Item of selected line
	oRouting	Routing of selected line
	oExceptionMessage	The last message if any message is found. If more than one message is given, these are present in the oExceptionID.
	oExceptionID	An ID that refers to the exception information. Use the functions in Exception to get all relevant information.
	Return: 0	Session started
	<> 0	Otherwise.

Public Interfaces for ProductVariant

The following functions are available:

[ProductVariant.CreateWithOptions](#)

[ProductVariant.GenerateProductStructureWithoutProject](#)

[ProductVariant.GenerateProductStructureWithoutProjectV2](#)

[ProductVariant.GenerateStructure](#)

[ProductVariant.GenerateStructureV2](#)

[ProductVariant.RegenerateJobShopStructure](#)

[ProductVariant.RegenerateJobShopStructureV2](#)

[ProductVariant.StartConfigurator](#)

[ProductVariant.StartMultiMain](#)

[ProductVariant.StartValidateProductVariants](#)

ProductVariant.CreateWithOptions

DLL:	tiextpcfapi
	This function is available from KB3540107 .
Syntax:	<pre>long ProductVariant.CreateWithOptions(domain tcitem iGenericItem, domain tcreft iReferenceType,</pre>

	<pre> long iItemConfigurationStructure, long iProcessingOptions, ref domain tccpva oCreatedProductVariant, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface is used to create a Product Variant with the set of provided sales options. This function makes use of a Processing Option Set, which can be created via a call to ProcessingOptionSet.Create(), and cleaned up after use, via a call to ProcessingOptionSet.Delete(). Item Configuration Structure can be created by using XML related functions xmlNewNode(), xmlNewDataElement(), xmlSetAttribute(), etc. Example: Create iItemConfigurationStructure Node. A car with color red(r) and 1 seat of color brown(b). <pre> <ItemConfiguration> <ConfigurationComponent> <Name>Car</Name> <Sequence>1</Sequence> <ParentComponent></ParentComponent> <Feature> <Sequence>1</Sequence> <OptionClass type="StringType">Color </OptionClass> <OptionClassDescription/> <Option>r</Option> <OptionDescription/> <Note/> </Feature> <Feature> <Sequence>2</Sequence> <OptionClass type="StringType"> NumberOfSeats</OptionClass> <OptionClassDescription/> <Option>1</Option> <OptionDescription/> <Note/> </Feature> </ConfigurationComponent> <ConfigurationComponent> <Name>Seat</Name> <Sequence>2</Sequence> <ParentComponent>1</ParentComponent> <Feature> <Sequence>1</Sequence> <OptionClass type="StringType">Color </OptionClass> <OptionClassDescription/> <Option>b</Option> <OptionDescription/> <Note/> </Feature> </ConfigurationComponent> </ItemConfiguration> </pre> Code Snippet: <pre> ***** long iItemConfigurationStructure long cc.id </pre>

	<pre> long req.id iItemConfigurationStructure = xmlNewNode("ItemConfiguration") /* Configuration Component 1 cc.id = xmlNewNode("ConfigurationComponent",XML_ELEMENT, iItemConfigurationStructure) req.id = xmlNewDataElement("Name","Car",cc.id) req.id = xmlNewDataElement("Sequence","1",cc.id) req.id = xmlNewNode("ParentComponent",XML_ELEMENT,cc.id) /* Feature 1 add.feature.to.item.configuration.structure(cc.id, "1", "Color", "", "r", "", "") /* Feature 2 add.feature.to.item.configuration.structure(cc.id, "2", "NumberOfSeats", "", "1", "", "") /* Configuration Component 2 cc.id = xmlNewNode("ConfigurationComponent",XML_ELEMENT, iItemConfigurationStructure) req.id = xmlNewDataElement("Name","Seat",cc.id) req.id = xmlNewDataElement("Sequence","2",cc.id) req.id = xmlNewDataElement("ParentComponent","1",cc.id) /* Feature 1 add.feature.to.item.configuration.structure(cc.id, "1", "Color", "", "b", "", "") ***** function void add.feature.to.item.configuration.structure(long i.cc.id, string i.seq(3), string i.opt.cls(15), string i.opt.cls.dsca(30), string i.option(15), string i.opt.dsca(30), string i.note(256)) { long req.id long feature long opt.cls feature = xmlNewNode("Feature",XML_ELEMENT,i.cc.id) req.id = xmlNewDataElement("Sequence",i.seq,feature) opt.cls = xmlNewDataElement("OptionClass",i.opt.cls, feature) </pre>

	<pre>req.id = xmlSetAttribute(opt.cls,"type","StringType") req.id = xmlNewDataElement("OptionClassDescription", i.opt.cls.dsca,feature) req.id = xmlNewDataElement("Option",i.option,feature) req.id = xmlNewDataElement("OptionDescription", i.opt.dsca,feature) req.id = xmlNewDataElement("Note",i.note,feature) } ***** Pre: db.retry point must be set. Post: Transaction must be aborted or committed. Input: iGenericItem - Generic Item. (Mandatory). iReferenceType - Reference Type of Product Variant to be created. iItemConfigurationStructure - XML Structure with Sales Options for Product Variant to be created. iProcessingOptionSet - A Processing Option Set can be created via a call to ProcessingOptionSet.Create(). If 0, then default values are applied (Optional). Processing Options are used for detailed specification of product variant settings. Order - Sales Order Number of Item. OrderPosition - Sales Order Position of Item. AlternativeSalesQuotation - Alternative Sales Quotation of Item. BusinessPartner - Business Partner of Sales Order. SalesCurrency - Sales Currency of Sales Order. Quantity - Quantity Ordered of Item. SalesPriceUnit - Sales Price Unit of Sales Order. ShipFromWarehouse - Warehouse from where the Item will be shipped. ShipToAddress - Address to which the Item will be shipped. ReferenceDate - Reference Date of Sales Order. Processing Options which are not provided will be filled with default value. NAME TYPE DEFAULT Order domain tcorno "" OrderPosition domain tcpono 0 AlternativeSalesQuotation domain tcpono 0 BusinessPartner domain tccom.bpid "" SalesCurrency domain tcccur "" Quantity domain tcqsl1 0.0 SalesPriceUnit domain tccuni "" ShipFromWarehouse domain tccwar "" ShipToAddress domain tccom.cadr "" ReferenceDate domain tiutcs 0 Output: oCreatedProductVariant - The created Product Variant. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in</pre>

	Return: 0 <> 0 Exception to get all relevant information. - Product Variant is created. - Otherwise.

ProductVariant.GenerateProductStructureWithoutProject

DLL:	tiextpcfapi
	This function is available from KB2295738 .
Syntax:	<pre>long ProductVariant.GenerateProductStructureWithoutProject(domain tccpva iProductVariant, domain tcdate iReferenceDate, ref domain tcitem oItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface is depreciated. Please use ProductVariant.GenerateStructure. This Public Interface generates a Product Structure based on a configured Product Variant, without a Project. This can also be done using session tipcs2220m000. The configured Product Variant can be created using ProductVariant.StartConfigurator or via the Product Configurator (tipcf5120m000). Pre: Db.retry point must be set Product Variant must be configured. Post: Transaction must be aborted or committed. Input: iProductVariant - Product Variant (Mandatory). Cannot be for an Assembly Item iReferenceDate - Reference Date (Optional). If empty, the current date is used. Output: oItem - The created custom Item. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Structure generation completed. <> 0 - Otherwise.

ProductVariant.GenerateProductStructureWithoutProjectV2

DLL:	tiextpcfapi
	This function is available from KB3539640 .
Syntax:	<pre> long ProductVariant.GenerateProductStructureWithoutProjectV2(domain tccpva iProductVariant, domain tcdate iReferenceDate, domain tcyesno iRegenerate, ref domain tcitem oItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface generates a Product Structure based on a configured Product Variant, without a Project. This can also be done using session tipcs2220m000. The configured Product Variant can be created using ProductVariant.StartConfigurator or via the Product Configurator (tipcf5120m000). Pre: Db.retry point must be set Product Variant must be configured. Post: Transaction must be aborted or committed. Input: - iProductVariant - Product Variant (Mandatory). Cannot be for an Assembly Item - iReferenceDate - Reference Date (Optional). If empty, the current date is used. - iRegenerate - If yes, an existing structure for the Product Variant will be regenerated. Output: - oItem - The created custom Item. - oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Structure generation completed. <> 0 - Otherwise.

ProductVariant.GenerateStructure

DLL:	tiextpcfapi
	This function is available from KB2226156 .
Syntax:	<pre> long ProductVariant.GenerateStructure(domain tccprj iProject, domain tccpva iProductVariant, domain tcdate iReferenceDate, domain tcyesno iUpdateReferenceDate, </pre>

	<pre> domain tcyesno iCheckStock, ref domain tcitem oItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface generates a Product Structure based on a configured Product Variant, similar to tipcs2220m000. The configured Product Variant can be created using ProductVariant.StartConfigurator or via the Product Configurator (tipcf5120m000). Existing Sales Order lines and Quotation Order lines belonging to the Product Variant are updated. Transaction management is handled by the Public Interface. Pre: None. Post: None. Input: iProject - Project (Mandatory, if the Generic top item for the Product Variant is With PCS). iProductVariant - Product Variant (Mandatory). Cannot be for an Assembly Item iReferenceDate - Reference Date (Optional). If empty, the Reference Date for the given Project is taken. If Reference Date for the Project is empty, the current date is used. iUpdateReferenceDate - Update the Project Reference Date. iCheckStock - Check the Standard Item Inventory. Output: oItem - The created custom Item. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Structure generation completed. <> 0 - Otherwise.

ProductVariant.GenerateStructureV2

DLL:	tiextpcfapi
	This function is available from KB3500971 .
Syntax:	<pre> long ProductVariant.GenerateStructureV2(domain tccprj iProject, domain tccpva iProductVariant, domain tcdate iReferenceDate, domain tcyesno iRegenerate, domain tcyesno iUpdateReferenceDate, domain tcyesno iCheckStock, ref domain tcitem oItem, ref domain tcmcs.s999m oExceptionMessage mb, </pre>

	ref	long	oExceptionID)
Usage:	Expl:	This Public Interface generates a Product Structure based on a configured Product Variant, similar to tipcs2220m000. The configured Product Variant can be created using ProductVariant.StartConfigurator or via the Product Configurator (tipcf5120m000).	
		Existing Sales Order lines and Quotation Order lines belonging to the Product Variant are updated.	
		Transaction management is handled by the Public Interface.	
	Pre:	None.	
	Post:	None.	
	Input:	iProject	- Project (Mandatory, if the Generic top item for the Product Variant is With PCS).
		iProductVariant	- Product Variant (Mandatory). Cannot be for an Assembly Item
		iReferenceDate	- Reference Date. If empty, the Reference Date for the given Project is taken. If Reference Date for the Project is empty, the current date is used.
		iRegenerate	- If yes, the Product Variant is also Regenerated.
		iUpdateReferenceDate	- Update the Project Reference Date.
Output:		iCheckStock	- Check the Standard Item Inventory.
		oItem	- The created custom Item.
		oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.
		oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.
	Return:	0	- Structure generation completed.
		<> 0	- Otherwise.

ProductVariant.RegenerateJobShopStructure

DLL:	tiextpcfapi		
Syntax:	<pre>long ProductVariant.RegenerateJobShopStructure(domain tccpva iProductVariant, boolean ref domain tcitem iUpdateOrderPrice, ref domain tcmcs.s999m oItem, ref long oExceptionMessage mb, oExceptionID)</pre>		
Usage:	Expl:	This Public Interface regenerates a (Project) Structure for a Product Variant based on a Job Shop Generic Item. Transaction management is handled by the Public Interface.	
	Pre:	None.	
	Post:	None.	
	Input:	iProductVariant	Product Variant. (Mandatory)
		iUpdateOrderPrice	Updates price in case reference type is

	Output: oItem oExceptionMessage oExceptionID Return: 0 <> 0	Sales Order / Sales Quotation. - The created custom Item. - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Structure regeneration completed. - Otherwise.

ProductVariant.RegenerateJobShopStructureV2

DLL:	tiextpcfapi This function is available from KB2284269 .	
Syntax:	<pre>long ProductVariant.RegenerateJobShopStructureV2(domain tccpva iProductVariant, boolean iUpdateOrderPrice, domain tccwar iShipFromWarehouse, ref domain tcitem oItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: This Public Interface regenerates a (Project) Structure for a Product Variant based on a Job Shop Generic Item. Transaction management is handled by the Public Interface. Pre: None. Post: None. Input: iProductVariant Product Variant. (Mandatory) iUpdateOrderPrice Updates price in case reference type is Sales Order / Sales Quotation. iShipFromWarehouse Warehouse for Project Part. This is typically the Warehouse of the Sales Order Line or Sales Quotation Line from which the goods are shipped. Output: oItem oExceptionMessage oExceptionID Return: 0 <> 0 - The created custom Item. - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Structure regeneration completed. - Otherwise.	

ProductVariant.StartConfigurator

DLL:	tiextpcfapi This function is available from KB2226156 .
Syntax:	<pre> long ProductVariant.StartConfigurator(domain tcitem iGenericItem, domain tccprj iProject, domain tccpva iProductVariant, domain tcreft iReferenceType, domain tcccur iCurrency, domain tcqsl1 iOrderQuantity, domain tccwar iShipFromWarehouse, domain tccuni iSalesPriceUnit, ref domain tccpva oProductVariant, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface starts the Product Variant configurator. Transaction management is handled by the Public Interface. Pre: None. Post: None. Input: - iGenericItem - Generic Item. (Mandatory) - iProject - Project (Mandatory if the Generic Item is With PCS). - iProductVariant - Product Variant (Optional). The Generic top-item for the Product Variant should match iGenericItem. - iReferenceType - Reference Type (Mandatory). - iCurrency - Currency (Optional). If the project type is standard, the default currency is used. If the currency system is dependent or standard, the currency of the calculation office is used. In these situations, this input is ignored. - iOrderQuantity - Order Quantity (Not used unless iReferenceType is tcreft.sls.order or tcreft.sls.quotation and iProductVariant is not empty). - iShipFromWarehouse - Ship from Warehouse (Not used unless iReferenceType is tcreft.sls.order or tcreft.sls.quotation and iProductVariant is not empty). - iSalesPriceUnit - Sales Price Unit (Not used unless iReferenceType is tcreft.sls.order or tcreft.sls.quotation and iProductVariant is not empty). Output: - oProductVariant - The created Product Variant. - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Configurator started successfully.

	<> 0	- Otherwise.

ProductVariant.StartMultiMain

DLL:	tiextpcfapi
Syntax:	<pre> long ProductVariant.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccpva iProductVariant, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface starts session Product Variant (tipcf5600m00) Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used iSessionIndex Not used iQueryExtend Not Used iProductVariant Mandatory. The Product Variant for which the session will be started Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Otherwise.

ProductVariant.StartValidateProductVariants

DLL:	tiextpcfapi This function is available from KB3647786 .
Syntax:	<pre> long ProductVariant.StartValidateProductVariants(long iStartMode, domain tccpva iProductVariantFrom, domain tccpva iProductVariantTo, domain tccom.bpid iBusinessPartnerFrom, domain tccom.bpid iBusinessPartnerTo, domain tcyesno iOverwriteSalesPrice, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface can be used to start process session Validate Product Variants(tipcf5200m000). Pre: N.A. Post: N.A. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS_ALWAYS - Parent and child are parallel sessions that can be manipulated simultaneously, even if the session is a Dialog. iProductVariantFrom From Product Variant - Optional. The default is the minimum domain value iProductVariantTo To Product Variant - Optional. The default is equal to iProductVariantFrom if it is given. The maximum domain value otherwise. iBusinessPartnerFrom From Business Partner - Optional. The default is the minimum domain value. iBusinessPartnerTo To Business Partner - Optional. The default is equal to iBusinessPartnerFrom if it is given. The maximum value otherwise. iOverwriteSalesPrice Control to overwrite the sales price - Optional Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 Error Occurred

Public Interfaces for ConfigurableItemConstraintIDs

The following functions are available:

[ConfigurableItemConstraintIDs.StartOverview](#)

ConfigurableItemConstraintIDs.StartOverview

DLL:	tiextpcfapi
Syntax:	<pre> long ConfigurableItemConstraintIDs.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, ref domain tcitem oItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface starts session Configurable Item - Constraint IDs (tipcf2101m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used. iSessionIndex Not used. iQueryExtend A specific query to be used when zooming to this session. iItem The Configurable Item for which the session will be started. Mandatory. Output: Variables below contain the values of the selected record. They are only filled if iStartMode is MODAL and 1 record has been selected. - oItem - Configurable Item. - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant

	Return: 0 <> 0	information. - Session started. - Otherwise.

Public Interfaces for ConfigurableItemGenericBillOfMaterial

The following functions are available:

[ConfigurableItemGenericBillOfMaterial.StartOverview](#)

ConfigurableItemGenericBillOfMaterial.StartOverview

DLL:	tiextpcfapi
Syntax:	<pre> long ConfigurableItemGenericBillOfMaterial.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, ref domain tcitem oItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface starts session Generic Bill of Material (tipcf3110m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used. iSessionIndex Not used. iQueryExtend A specific query to be used when zooming to this session. iItem The Configurable Item for which the session will be started. Mandatory. Output: Variables below contain the values of the selected record.

	They are only filled if iStartMode is MODAL and 1 record has been selected. - oItem - Configurable Item. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started. <> 0 - Otherwise.

Public Interfaces for ConfigurableItemGenericPriceLists

The following functions are available:

[ConfigurableItemGenericPriceLists.StartOverview](#)

ConfigurableItemGenericPriceLists.StartOverview

DLL:	tiextpcfapi
Syntax:	<pre>long ConfigurableItemGenericPriceLists.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, ref domain tcitem oItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This Public Interface starts session Generic Price Lists (tipcf4101m000). Input: iStartMode Specifies the start mode for the session. Possible values are: - MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. - MODELESS - Parent and child are parallel sessions that can be manipulated

	simultaneously. iStartFilter Not used. iSessionIndex Not used. iQueryExtend A specific query to be used when zooming to this session. iItem The Configurable Item for which the session will be started. Mandatory. Output: Variables below contain the values of the selected record. They are only filled if iStartMode is MODAL and 1 record has been selected. - oItem - Configurable Item. - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Return: 0 - Session started. <> 0 - Otherwise.

Public Interfaces for ConfigurableItemGenericRouting

The following functions are available:

[ConfigurableItemGenericRouting.StartOverview](#)

ConfigurableItemGenericRouting.StartOverview

DLL:	tiextpcfapi
Syntax:	<pre>long ConfigurableItemGenericRouting.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, ref domain tcitem oItem,</pre>

	<pre> ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface starts session Generic Routing (tipcf3120m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used. iSessionIndex Not used. iQueryExtend A specific query to be used when zooming to this session. iItem The Configurable Item for which the session will be started. Mandatory. Output: Variables below contain the values of the selected record. They are only filled if iStartMode is MODAL and 1 record has been selected. - oItem - Configurable Item. - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Return: 0 - Session started. - <> 0 - Otherwise.

Public Interfaces for ConfigurableItemProductFeatures

The following functions are available:

[ConfigurableItemProductFeatures.StartOverview](#)

ConfigurableItemProductFeatures.StartOverview

DLL:	tiextpcfapi
Syntax:	<pre> long ConfigurableItemProductFeatures.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, domain tccpft iProductFeature, ref domain tcitem oItem, ref domain tccpft oProductFeature, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface starts session Product Features by Configurable Item (tipcf1101m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used. iSessionIndex Specifies the sort order of the records being displayed. Possible values are: 1 - Session is sorted by Item, Product feature. 2 - Session is sorted by Product feature, Item. Not supported when iStartMode = MODAL. 3 - Session is sorted by Item, Sequence number. iQueryExtend A specific query to be used when zooming to this session. iItem The Configurable Item for which the session will be started. Mandatory if iSessionIndex is 1 or 3. iProductFeature The Product Feature for which the session will be started. Mandatory if iSessionIndex is 2. Output: Variables below contain the values of the selected record. They are only filled if iStartMode is MODAL and 1 record has been selected. - oItem - Configurable Item. - oProductFeature - Product Feature. - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant

	Return: 0 <> 0	information. - Session started. - Otherwise.

Public Interfaces for ConfigurableItemSettingsForDataGeneration

The following functions are available:

[ConfigurableItemSettingsForDataGeneration.StartOverview](#)

ConfigurableItemSettingsForDataGeneration.StartOverview

DLL:	tiextpcfapi
Syntax:	<pre> long ConfigurableItemSettingsForDataGeneration.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, domain tckoid iItemField, ref domain tcitem oItem, ref domain tckoid oItemField, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface starts session Generic Item - Settings for Data Generation (tipcf3101m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used. iSessionIndex Specifies the sort order of the records being displayed. Possible values are: 1 - Session is sorted by Item, Item Field,

	Sequence Number. 3 - Session is sorted by Item, Item Field, Sequence Number, Text Block. iQueryExtend A specific query to be used when zooming to this session. iItem The Configurable Item for which the session will be started. Mandatory. iItemField The Item Field for which the session will be started. Mandatory if iSessionIndex is 1. Output: Variables below contain the values of the selected record. They are only filled if iStartMmode is MODAL and 1 record has been selected. - oItem - Configurable Item. - oItemField - Item Field. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started. <> 0 - Otherwise.

Public Interfaces for StandardItemConfigurationsOptions

The following functions are available:

[StandardItemConfigurationsOptions.StartOverview](#)

StandardItemConfigurationsOptions.StartOverview

DLL:	tiextpcfapi
	This function is available from KB2247487 .
Syntax:	<pre>long StandardItemConfigurationsOptions.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter,</pre>

	<pre> const long iSessionIndex, string iQueryExtend(), domain tcitem iItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface starts session Standard Item - Configurations Options (tipcf5160m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used. iSessionIndex Not used. iQueryExtend Not used. iItem The Standard Item for which the session will be started. Mandatory. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started. <> 0 - Otherwise.

Public Interfaces for ProductVariantOptions

The following functions are available:

[ProductVariantOptions.StartOverview](#)

ProductVariantOptions.StartOverview

DLL:	tiextpcfapi This function is available from KB2283720 .
Syntax:	<pre> long ProductVariantOptions.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccpva iProductVariant, domain tcopts iOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Product Variant Options in overview mode (tipcf5520m000). Input: <i>iStartMode</i> Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. <i>iStartFilter</i> Not Used. <i>iSessionIndex</i> Not Used. <i>iQueryExtend</i> A specific query to be used when zooming to this session. Optional <i>iProductVariant</i> The Product Variant for which the session will be started. Mandatory. <i>iOptionSet</i> The Option Set for which the session will be started. Output: <i>oExceptionMessage</i> - The last message if the return value is not equal to 0. If more than one message is given, these are present in the <i>oExceptionID</i> <i>oExceptionID</i> - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Otherwise.

Public Interfaces for ProductionOrder

The following functions are available:

[ProductionOrder.Cancel](#)
[ProductionOrder.CheckShortages](#)
[ProductionOrder.CreateAsBuilt](#)
[ProductionOrder.FinishOperationAddition](#)
[ProductionOrder.InitiateInventoryIssue](#)
[ProductionOrder.InitiateInventoryIssueV2](#)
[ProductionOrder.InitiateMaterialInventoryIssue](#)
[ProductionOrder.MoveRejectedMaterialToQuarantine](#)
[ProductionOrder.NewVersion](#)
[ProductionOrder.Release](#)
[ProductionOrder.ReportProduct](#)
[ProductionOrder.ReportProductV2](#)
[ProductionOrder.ReportProductWithSerials](#)
[ProductionOrder.ReportProductWithSerialsV2](#)
[ProductionOrder.ResetCompletedToActive](#)
[ProductionOrder.ScrapRejectedMaterial](#)
[ProductionOrder.StartClose](#)
[ProductionOrder.StartDetail](#)
[ProductionOrder.StartInventoryOverview](#)
[ProductionOrder.StartMultiMain](#)
[ProductionOrder.StartOperationAddition](#)
[ProductionOrder.StartPrintDocuments](#)
[ProductionOrder.StartReportOperationsCompleted](#)
[ProductionOrder.StartReportOrdersCompleted](#)
[ProductionOrder.StartResetStatus](#)
[ProductionOrder.UpdateMaterialToIssueQuantity](#)
[ProductionOrders.StartOverview](#)

ProductionOrder.Cancel

DLL:	tiextsfapi
Syntax:	<pre> long ProductionOrder.Cancel(domain tcsite iSite, domain tcpdno iProductionOrder, boolean iAutoComplete, boolean iReturnMaterials, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will cancel a Production Order in status "Released" or "Active". Transaction management is handled by this public interface. Pre: None. Post: None. Input: iSite Site (mandatory if active). iProductionOrder Production Order (mandatory). iAutoComplete Option if Active Production Orders have to be completed automatically. iReturnMaterials Option if already issued Materials have to be returned. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Production Order has been cancelled. <> 0 - Production Order has not been cancelled.

ProductionOrder.CheckShortages

DLL:	tiextsfapi
Syntax:	<pre> long ProductionOrder.CheckShortages(domain tcsite iSite, domain tcpdno iProductionOrder, domain tisfc.stck iShortageCheckType, domain tisfc.scin iShortageCheckScope, domain tcyesno iShortageCheckSkipBlockedInventory, domain tcyesno iShortageCheckCriticalItemsOnly, ref boolean oShortageDetected, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface can be used to check for material shortages prior to releasing the production order. Pre: None Post: None

	Input: iSite Site (mandatory when the Site concept is active). iProductionOrder Production Order (mandatory and must be in iSite). iShortageCheckType Type of Shortage Check (mandatory) iShortageCheckScope Scope of Shortage Check (mandatory) iShortageCheckSkipBlockedInventory Exclude blocked inventory from Shortage Check (mandatory) iShortageCheckCriticalItemsOnly Check Critical Items only during Shortage Check (mandatory) Output: oShortageDetected Indicates if a shortage was detected. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Return: 0 Check Shortages executed successfully. <> 0 An error encountered during check for shortages.

ProductionOrder.CreateAsBuilt

DLL:	tiextsfapi
Syntax:	<pre>long ProductionOrder.CreateAsBuilt(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcitem iItem, domain tcibd.sern iSerialNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this public interface to add a created Serial Number to the As-Built Header of the Production Order. The Serial Number can be created using ProductionOrder.CreateSerialNumber. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iSite Site (mandatory if active). iProductionOrder Production Order (mandatory). iItem Item (mandatory). iSerialNumber Serial Number (mandatory) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information.

	Return: 0 <> 0	- Serial Number has been added - Serial Number has not been added

ProductionOrder.FinishOperationAddition

DLL:	tiextsfapi
	This function is available from KB3511148 .
Syntax:	<pre>long ProductionOrder.FinishOperationAddition(domain tcsite iSite, domain tcpdno iProductionOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: With this Public Interface, the production order that was prepared for having operation(s) added (via ProductionOrder.StartOperationAdditions), is reset to standard behavior. Input: iSite Site (mandatory when the Site concept is active). iProductionOrder Production Order (mandatory and must be in iSite). Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Operation(s) added via Public Interface ProductionOrder.StartOperationAdditions has reset successfully. <> 0 Errors occurred.

ProductionOrder.InitiateInventoryIssue

DLL:	tiextsfapi
Syntax:	<pre>long ProductionOrder.InitiateInventoryIssue(domain tcsite iSite, domain tcpdno iProductionOrder, domain tisfc.wybf iOperationSelectionMethod, domain tcopno iOperation, domain tcutcs iActualDate, domain tcyesno iProcessSerialUsingAsBuilt, domain tcyesno iProcessSerialNotInAsBuilt, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>

Usage:	Expl: *** Warning *** This public interface is deprecated. use: ProductionOrder.InitiateInventoryIssueV2 This function executes the Initiate Inventory Issue functionality, as in session tisfc0207m000, for 1 production order. NB. - retrypoint and commit / rollback is done in the function - no reports are printed Input: iSite Site (mandatory when the Site concept is active) iProductionOrder Production Order (mandatory and must be in iSite) iOperationSelectionMethod Operation(s) to handle: all upto one empty (= all) iOperation Operation (mandatory in case iOperationSelectionMethod = tisfc.wybf.one) iActualDate Actual Date (0 = current date): date used on warehouse transactions iProcessSerialUsingAsBuilt Process Serialized Materials In Inventory according to the As Built structure iProcessSerialNotInAsBuilt Process Serialized Materials In Inventory which are not in the As Built structure Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Return: 0 Initiate Inventory Issue completed succesfully. <> 0 Initiate Inventory Issue was not completed succesfully.

ProductionOrder.InitiateInventoryIssueV2

DLL:	tiextsfcap
Syntax:	<pre> long ProductionOrder.InitiateInventoryIssueV2 (domain tcsite iSite, domain tcpdno iProductionOrder, domain tisfc.wybf iOperationSelectionMethod, domain tcopno iOperation, domain tcdate iUpToDate, domain tcutcs iActualDate, </pre>

	<pre> domain tcyesno iProcessSerialUsingAsBuilt, ref domain tcyesno iProcessSerialNotInAsBuilt, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function executes the Initiate Inventory Issue functionality, as in session tisfc0207m000, for 1 production order. NB. - retrypoint and commit / rollback is done in the function - no reports are printed Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: - iSite Site (mandatory when the Site concept is active). - iProductionOrder Production Order (mandatory and must be in iSite). - iOperationSelectionMethod Operation(s) to handle: - all - upto - one - empty (= all) - iOperation Operation (mandatory in case iOperationSelectionMethod = tisfc.wybf.one). - iUpToDate Select materials with requirement date up to the specified date (if 0, all materials will be selected). - iActualDate Actual Date (0 = current date): date used on warehouse transactions. - iProcessSerialUsingAsBuilt Process Serialized Materials In Inventory according to the As Built structure. - iProcessSerialNotInAsBuilt Process Serialized Materials In Inventory which are not in the As Built structure. Output: - oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. - oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Return: 0 Initiate Inventory Issue completed succesfully. <> 0 Initiate Inventory Issue was not completed succesfully.

ProductionOrder.InitiateMaterialInventoryIssue

DLL:	tiextsfapi
Syntax:	<pre> long ProductionOrder.InitiateMaterialInventoryIssue(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcpono iMaterialPosition, domain tcutcs iIssueDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface will initiate Material Inventory issue for a Production Order. Transaction management is handled by this Public Interface. Pre: - Post: - Input: iSite Site (mandatory if active). iProductionOrder Production Order (mandatory). iMaterialPosition Material Position (mandatory). iIssueDate The Issue Date (optional). If 0, then the Planned Dates will not be updated. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Material Inventory Issue has been initiated. <> 0 - Otherwise.

ProductionOrder.MoveRejectedMaterialToQuarantine

DLL:	tiextsfapi
Syntax:	<pre> long ProductionOrder.MoveRejectedMaterialToQuarantine(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcpono iMaterialLine, domain tcitem iItem, domain tiqepl iQuantity, domain tcibd.sern iSerial, domain tcuef.effn iUnit, domain tcclot iLot, domain tccwar iQuarantineWarehouse, domain tiloca iQuarantineLocation, string iQuarantineText(1), boolean iDirectProcessInbound, ref domain tcmcs.s999m oExceptionMessage mb, </pre>

	ref	long	oExceptionID)
Usage:	Expl:	Use this Public Interface to move rejected Material to Quarantine for a given Production Order and Material Line. Transaction management is handled by this Public Interface.	
	Pre:	-	
	Post:	-	
	Input:	iSite	Site (mandatory when the Site concept is active).
		iProductionOrder	Production Order (mandatory, must be in iSite).
		iMaterialLine	Material Position Line (Mandatory).
		iItem	Item (Mandatory).
		iQuantity	Quantity to be moved to Quarantine. If iSerialToQuarantine is not empty, iQuantityToQuarantine must be 1.0. (Mandatory).
		iSerial	The serial number of the Material to be moved to Quarantine (Optional).
		iUnit	Effectivity Unit (Optional).
		iLot	Lot Code (Optional).
		iQuarantineWarehouse	Warehouse for the Quarantine (Mandatory).
		iQuarantineLocation	Quarantine location (Optional).
		iQuarantineText	Quarantine text (Optional).
		iDirectProcessInbound	Direct process Warehouse Order (Optional).
	Output:		
		oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.
		oExceptionID	An ID that refers to all error information.
			Use the functions in Exception handling to get all relevant information.
	Return:	0	Material has been moved to Quarantine.
		<> 0	Material has not been moved to Quarantine.

ProductionOrder.NewVersion

DLL:	tiextsfapi
	This function is available from KB2274584 .
Syntax:	<pre> long ProductionOrder.NewVersion(domain tcsite iSite, domain tcpdno iProductionOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this Public Interface to update the status of the given Production Order to Modify. The update also increases the version number of the Production Order, following the logic of session Production Order (tisfc0101m100).

	<pre> Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iSite Site (mandatory when the Site concept is active). iProductionOrder Production Order (mandatory, must be in iSite). Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Return: 0 The status has been updated. <> 0 The status has not been updated. </pre>

ProductionOrder.Release

DLL:	tiextsfscapi
Syntax:	<pre> long ProductionOrder.Release(domain tcsite iSite, domain tcpdno iProductionOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl: This public interface releases the given Production Order. In case Electronic Signature is required this function cannot be used. In case a check on shortages must be done before releasing the production order, public interface function ProductionOrder.CheckShortages can be called first. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iSite Site (mandatory when the Site concept is active). iProductionOrder Production Order (mandatory and must be in iSite). Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Return: 0 Production Order released successfully <> 0 Otherwise </pre>

ProductionOrder.ReportProduct

DLL:	tiextsfscapi						
Syntax:	<pre> long ProductionOrder.ReportProduct (domain tcsite iSite, domain tcpdno iProductionOrder, domain tcclot iLotCode, domain tiqep2 iProducedQuantity, domain tiqep2 iRejectedQuantity, domain tccdis iRejectReason, domain tcutcs iCompletionDate, boolean iSetStatusToCompleted, ref domain tcclot oLotCode, ref boolean oOrderIsCompleted, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>						
Usage:	Expl: With this Public Interface, a quantity can be reported complete and or rejected for a specific production order, as in session tisfc0120s000. If the main item of the production order requires backflushing, then Backflushing will be executed, but only when the company or site setting for the backflushing method is 'Automatic'. When the setting of the backflushing method is 'Manual' or 'Interactive', then no backflushing will be done. If iSetStatusToCompleted is true, then the production order status will be set to 'To be Completed' and the required warehouse procedures will be executed, but only if they have the setting 'automatic'. If all the warehouse procedures are executed, then the status of the production order will be changed from 'To be Completed' to 'Completed'. The output argument oOrderIsCompleted will indicate if the status is 'Completed'. When false, then additional processing of the warehouse orders is required. iSetStatusToCompleted is not allowed when: - the production order requires backflushing and - backflushing is not automatic and - actual costing is used. Printing labels is not supported in this Public Interface. An error will be returned when the production order requires label printing. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: <table> <tr> <td>iSite</td> <td>Site (mandatory when the Site concept is active).</td> </tr> <tr> <td>iProductionOrder</td> <td>Production Order (mandatory and must be in iSite).</td> </tr> <tr> <td>iLotCode</td> <td>Code of existing or new lot or empty. When empty and a lot is required, then a new lot will be created.</td> </tr> </table>	iSite	Site (mandatory when the Site concept is active).	iProductionOrder	Production Order (mandatory and must be in iSite).	iLotCode	Code of existing or new lot or empty. When empty and a lot is required, then a new lot will be created.
iSite	Site (mandatory when the Site concept is active).						
iProductionOrder	Production Order (mandatory and must be in iSite).						
iLotCode	Code of existing or new lot or empty. When empty and a lot is required, then a new lot will be created.						

		When the specified lot does not exist, then it will be created. The quantity that is reported as completed. The quantity that is reported as rejected. Optional and can only be filled when iProductionOrder does not have operations. The reason for rejection. Date of completion If true then the status of the production order will be set to 'To be Completed' or 'Completed'; else the status will not be changed.
	iProducedQuantity iRejectedQuantity iRejectReason iCompletionDate iSetStatusToCompleted	
	Output:	
	oLotCode	Code of the lot that was actually used.
	oOrderIsCompleted	If true when the production order status was changed to Completed, else false.
	oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.
	oExceptionID	An ID that refers to all error information.
	Return: 0	Use the functions in Exception handling to get all relevant information.
	<> 0	The reported quantities and / or status change is succssfully processed. Errors occurred.

ProductionOrder.ReportProductV2

DLL:	tiextsfapi
Syntax:	<pre> long ProductionOrder.ReportProductV2 (domain tcsite iSite, domain tcpdno iProductionOrder, domain tcclot iLotCode, domain tiqep2 iProducedQuantity, domain tiqep2 iRejectedQuantity, domain tcdis iRejectReason, domain tcuef.effn iEffectivityUnit, domain tcutcs iCompletionDate, boolean iSetStatusToCompleted, boolean iAllowReportMoreThanOrdered, boolean iDirectReceipt, domain tcyesno iDoBackflush, ref domain tcclot oLotCode, ref boolean oOrderIsCompleted, ref boolean oBackflushCompleted, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: With this Public Interface, a quantity can be reported complete and or rejected for a specific production order, as in session tisfc0120s000. If the main item of the production order requires backflushing,

	then Backflushing will be executed when the company or site setting for the backflushing method is 'Automatic', or when iDoBackflush is 'Yes'. If iSetStatusToCompleted is true, then the production order status will be set to 'To be Completed' and the required warehouse procedures will be executed, but only if they have the setting 'automatic'. If all the warehouse procedures are executed, then the status of the production order will be changed from 'To be Completed' to 'Completed'. The output argument oOrderIsCompleted will indicate if the status is 'Completed'. When false, then additional processing of the warehouse orders is required. iSetStatusToCompleted is not allowed when: - the production order requires backflushing and - backflushing is not automatic and - actual costing is used. Printing labels is not supported in this Public Interface. An error will be returned when the production order requires label printing. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: <table> <tr> <td>iSite</td><td>Site (mandatory when the Site concept is active).</td></tr> <tr> <td>iProductionOrder</td><td>Production Order (mandatory and must be in iSite).</td></tr> <tr> <td>iLotCode</td><td>Code of existing or new lot or empty. When empty and a lot is required, then a new lot will be created. When the specified lot does not exist, then it will be created.</td></tr> <tr> <td>iProducedQuantity</td><td>The quantity that is reported as completed.</td></tr> <tr> <td>iRejectedQuantity</td><td>The quantity that is reported as rejected. Optional and can only be filled when iProductionOrder does not have operations.</td></tr> <tr> <td>iRejectReason</td><td>The reason for rejection.</td></tr> <tr> <td>iEffectivityUnit</td><td>The Effectivity Unit. Not used when set to 0.</td></tr> <tr> <td>iCompletionDate</td><td>Date of completion.</td></tr> <tr> <td>iSetStatusToCompleted</td><td>If true then the status of the production order will be set to 'To be Completed' or 'Completed'; else the status will not be changed.</td></tr> <tr> <td>iAllowReportMoreThanOrdered</td><td>If true then more can be reported as complete than ordered.</td></tr> <tr> <td>iDirectReceipt</td><td>Process inbound of the Item as soon as the Order is reported complete.</td></tr> <tr> <td>iDoBackflush</td><td>Execute backflushing even when backflushing is not done automatically.</td></tr> </table> Output: <table> <tr> <td>oLotCode</td><td>Code of the lot that was actually used.</td></tr> <tr> <td>oOrderIsCompleted</td><td>If true when the production order status was changed to Completed,</td></tr> </table>	iSite	Site (mandatory when the Site concept is active).	iProductionOrder	Production Order (mandatory and must be in iSite).	iLotCode	Code of existing or new lot or empty. When empty and a lot is required, then a new lot will be created. When the specified lot does not exist, then it will be created.	iProducedQuantity	The quantity that is reported as completed.	iRejectedQuantity	The quantity that is reported as rejected. Optional and can only be filled when iProductionOrder does not have operations.	iRejectReason	The reason for rejection.	iEffectivityUnit	The Effectivity Unit. Not used when set to 0.	iCompletionDate	Date of completion.	iSetStatusToCompleted	If true then the status of the production order will be set to 'To be Completed' or 'Completed'; else the status will not be changed.	iAllowReportMoreThanOrdered	If true then more can be reported as complete than ordered.	iDirectReceipt	Process inbound of the Item as soon as the Order is reported complete.	iDoBackflush	Execute backflushing even when backflushing is not done automatically.	oLotCode	Code of the lot that was actually used.	oOrderIsCompleted	If true when the production order status was changed to Completed,
iSite	Site (mandatory when the Site concept is active).																												
iProductionOrder	Production Order (mandatory and must be in iSite).																												
iLotCode	Code of existing or new lot or empty. When empty and a lot is required, then a new lot will be created. When the specified lot does not exist, then it will be created.																												
iProducedQuantity	The quantity that is reported as completed.																												
iRejectedQuantity	The quantity that is reported as rejected. Optional and can only be filled when iProductionOrder does not have operations.																												
iRejectReason	The reason for rejection.																												
iEffectivityUnit	The Effectivity Unit. Not used when set to 0.																												
iCompletionDate	Date of completion.																												
iSetStatusToCompleted	If true then the status of the production order will be set to 'To be Completed' or 'Completed'; else the status will not be changed.																												
iAllowReportMoreThanOrdered	If true then more can be reported as complete than ordered.																												
iDirectReceipt	Process inbound of the Item as soon as the Order is reported complete.																												
iDoBackflush	Execute backflushing even when backflushing is not done automatically.																												
oLotCode	Code of the lot that was actually used.																												
oOrderIsCompleted	If true when the production order status was changed to Completed,																												

	oBackflushCompleted else false. If true when the backflush could be executed, else false. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Return: 0 The reported quantities and / or status change is succssfully processed. <> 0 Errors occurred.

ProductionOrder.ReportProductWithSerials

DLL:	tiextsfapi
Syntax:	<pre> long ProductionOrder.ReportProductWithSerials(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcclot iLotCode, domain tiqep2 iProducedQuantity, domain tiqep2 iRejectedQuantity, domain tccdis iRejectReason, ref domain tcibd.sern iSerialArrayProduced() fixed, ref domain tcibd.sern iSerialArrayRejected() fixed, domain tcutcs iCompletionDate, boolean iSetStatusToCompleted, ref domain tcclot oLotCode, ref boolean oOrderIsCompleted, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: With this Public Interface, a quantity can be reported complete and/or rejected for a specific production order with serials, as in session tisfc0120s000. If the main item of the production order requires Backflushing, then Backflushing will be executed, but only when the company or site setting for the backflushing method is 'Automatic'. When the setting of the backflushing method is 'Manual' or 'Interactive', then no Backflushing will be done. If iSetStatusToCompleted is true, the production order status will be set to 'To be Completed' and the required warehouse procedures will be executed, but only if they have the setting 'Automatic'. If all the warehouse procedures are executed, the status of the production order will be changed from 'To be Completed' to 'Completed'. The output argument oOrderIsCompleted will indicate if the status is 'Completed'. When false, additional processing of the warehouse orders is required.

	iSetStatusToCompleted is not allowed when: - the production order requires backflushing and - backflushing is not automatic and - actual costing is used. Printing labels is not supported in this Public Interface. An error will be returned when the production order requires label printing. There should be no pending logical transactions before calling this function. Commit or abort transactions are handled within the function. Pre: - Post: - Input: <table> <tr> <td>iSite</td><td>Site (mandatory when the Site concept is active).</td></tr> <tr> <td>iProductionOrder</td><td>Production Order (mandatory and must exist within iSite).</td></tr> <tr> <td>iLotCode</td><td>Code of existing or new lot or empty. When empty and a lot is required, then a new lot will be created. When the specified lot does not exist, then it will be created.</td></tr> <tr> <td>iProducedQuantity</td><td>The quantity that is reported as completed.</td></tr> <tr> <td>iRejectedQuantity</td><td>The quantity that is reported as rejected. (Optional and can only be filled when iProductionOrder does not have operations.)</td></tr> <tr> <td>iRejectReason</td><td>The reason for rejection.</td></tr> <tr> <td>iSerialArrayProduced</td><td>Array of Serial Numbers for the quantity produced.</td></tr> <tr> <td>iSerialArrayRejected</td><td>Array of Serial Numbers for the quantity rejected.</td></tr> <tr> <td>iCompletionDate</td><td>Date of completion.</td></tr> <tr> <td>iSetStatusToCompleted</td><td>If true then the status of the production order will be set to 'To be Completed' or 'Completed'; else the status will not be changed.</td></tr> </table> Output: <table> <tr> <td>oLotCode</td><td>Code of the lot that was actually used.</td></tr> <tr> <td>oOrderIsCompleted</td><td>If true when the production order status was changed to Completed, else false.</td></tr> <tr> <td>oExceptionMessage</td><td>The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.</td></tr> <tr> <td>oExceptionID</td><td>An ID that refers to all error information. Use the functions in Exception handling to get all relevant information.</td></tr> </table> Return: 0 The reported quantities and / or status change is successfully processed. <> 0 Errors occurred.	iSite	Site (mandatory when the Site concept is active).	iProductionOrder	Production Order (mandatory and must exist within iSite).	iLotCode	Code of existing or new lot or empty. When empty and a lot is required, then a new lot will be created. When the specified lot does not exist, then it will be created.	iProducedQuantity	The quantity that is reported as completed.	iRejectedQuantity	The quantity that is reported as rejected. (Optional and can only be filled when iProductionOrder does not have operations.)	iRejectReason	The reason for rejection.	iSerialArrayProduced	Array of Serial Numbers for the quantity produced.	iSerialArrayRejected	Array of Serial Numbers for the quantity rejected.	iCompletionDate	Date of completion.	iSetStatusToCompleted	If true then the status of the production order will be set to 'To be Completed' or 'Completed'; else the status will not be changed.	oLotCode	Code of the lot that was actually used.	oOrderIsCompleted	If true when the production order status was changed to Completed, else false.	oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.	oExceptionID	An ID that refers to all error information. Use the functions in Exception handling to get all relevant information.
iSite	Site (mandatory when the Site concept is active).																												
iProductionOrder	Production Order (mandatory and must exist within iSite).																												
iLotCode	Code of existing or new lot or empty. When empty and a lot is required, then a new lot will be created. When the specified lot does not exist, then it will be created.																												
iProducedQuantity	The quantity that is reported as completed.																												
iRejectedQuantity	The quantity that is reported as rejected. (Optional and can only be filled when iProductionOrder does not have operations.)																												
iRejectReason	The reason for rejection.																												
iSerialArrayProduced	Array of Serial Numbers for the quantity produced.																												
iSerialArrayRejected	Array of Serial Numbers for the quantity rejected.																												
iCompletionDate	Date of completion.																												
iSetStatusToCompleted	If true then the status of the production order will be set to 'To be Completed' or 'Completed'; else the status will not be changed.																												
oLotCode	Code of the lot that was actually used.																												
oOrderIsCompleted	If true when the production order status was changed to Completed, else false.																												
oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.																												
oExceptionID	An ID that refers to all error information. Use the functions in Exception handling to get all relevant information.																												

ProductionOrder.ReportProductWithSerialsV2

DLL:	tiextsfscapi
Syntax:	<pre> long ProductionOrder.ReportProductWithSerialsV2 (domain tcsite iSite, domain tcpdno iProductionOrder, domain tcclot iLotCode, domain tiqep2 iProducedQuantity, domain tiqep2 iRejectedQuantity, domain tccdis iRejectReason, domain tcuef.effn iEffectivityUnit, ref domain tcibd.sern iSerialArrayProduced() fixed, ref domain tcibd.sern iSerialArrayRejected() fixed, domain tcutcs iCompletionDate, boolean iSetStatusToCompleted, boolean iAllowReportMoreThanOrdered, boolean iDirectReceipt, domain tcyesno iDoBackflush, ref domain tcclot oLotCode, ref boolean oOrderIsCompleted, ref boolean oBackflushCompleted, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: With this Public Interface, a quantity can be reported complete and/or rejected for a specific production order with serials, as in session tisfc0120s000. If the main item of the production order requires backflushing, then Backflushing will be executed when the company or site setting for the backflushing method is 'Automatic', or when iDoBackflush is 'Yes'. If iSetStatusToCompleted is true, the production order status will be set to 'To be Completed' and the required warehouse procedures will be executed, but only if they have the setting 'Automatic'. If all the warehouse procedures are executed, the status of the production order will be changed from 'To be Completed' to 'Completed'. The output argument oOrderIsCompleted will indicate if the status is 'Completed'. When false, additional processing of the warehouse orders is required. iSetStatusToCompleted is not allowed when: - the production order requires backflushing and - backflushing is not automatic and - actual costing is used. Printing labels is not supported in this Public Interface. An error will be returned when the production order requires label printing. There should be no pending logical transactions before calling this function. Commit or abort transactions are handled within the function. Pre: - Post: -

	Input: iSite Site (mandatory when the Site concept is active). iProductionOrder Production Order (mandatory and must exist within iSite). iLotCode Code of existing or new lot or empty. When empty and a lot is required, then a new lot will be created. When the specified lot does not exist, then it will be created. iProducedQuantity The quantity that is reported as completed. iRejectedQuantity The quantity that is reported as rejected. (Optional and can only be filled when iProductionOrder does not have operations.) iRejectReason The reason for rejection. iEffectivityUnit The Effectivity Unit. Not used when set to 0. iSerialArrayProduced Array of Serial Numbers for the quantity produced. iSerialArrayRejected Array of Serial Numbers for the quantity rejected. iCompletionDate Date of completion. iSetStatusToCompleted If true then the status of the production order will be set to 'To be Completed' or 'Completed'; else the status will not be changed. iAllowReportMoreThanOrdered If true then more can be reported as complete than ordered. iDirectReceipt Process inbound of the Item as soon as the Order is reported complete. iDoBackflush Execute backflushing even when backflushing is not done automatically. Output: oLotCode Code of the lot that was actually used. oOrderIsCompleted If true when the production order status was changed to Completed, else false. oBackflushCompleted If true when the backflush could be executed, else false. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Return: 0 The reported quantities and / or status change is successfully processed. <> 0 Errors occurred.

ProductionOrder.ResetCompletedToActive

DLL:	tiextsfapi
Syntax:	<pre> long ProductionOrder.ResetCompletedToActive(domain tcsite iSite, domain tcpdno iProductionOrder, boolean iResetOperations, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this public interface to reset the status of a completed Production Order to active, as in session ticst0203m000. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iSite Site (mandatory when the Site concept is active). iProductionOrder Production Order (mandatory, must be in iSite and its status must be Completed or To be Completed). iResetOperations If true: also reset the status of all Operations of the specified production order from Completed to Active. Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Return: 0 Rejected quantity is scrapped <> 0 Rejected quantity could not be scrapped

ProductionOrder.ScrapRejectedMaterial

DLL:	tiextsfapi
Syntax:	<pre> long ProductionOrder.ScrapRejectedMaterial(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcopno iMaterialPosition, domain tiqep2 iQuantityToScrap, domain tcibd.sern iSerialNumber, domain tcuef.effn iEffectivityUnit, domain tcclot iLotCode, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this Public Interface to scrap a quantity of rejected material of an operation, as in session tisfc0213m000. Pre: Retry point must be set.

	Post: Commit or abort the transaction. Input: iSite Site (Mandatory when the Site concept is active). iProductionOrder Production Order (Mandatory and must be in iSite). iMaterialPosition MaterialPosition (Mandatory). iQuantityToScrap Quantity to Scrap (Mandatory). When the Serial Number is not empty, this value must be 1.0. iSerialNumber The Serial Number (Optional). Should exist if given. iEffectivityUnit Effectivity Unit (Optional). iLotCode Lot Code (Optional). Should exist if given. Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Return: 0 Rejected quantity is scrapped. <> 0 Rejected quantity was not scrapped.

ProductionOrder.StartClose

DLL:	tiextsfapi This function is available from KB2274114.
Syntax:	<pre> long ProductionOrder.StartClose(long iStartMode, boolean iIgnoreSelectionFields, domain tcsite iFromSite, domain tcpdno iFromProductionOrder, domain tcpdno iFromOrderGroup, domain tcsite iToSite, domain tcpdno iToProductionOrder, domain tcpdno iToOrderGroup, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Close Production Orders (ticst0201m000). Input: iStartMode Not Used. iIgnoreSelectionFields This argument determines whether user defaults are used. User defaults should take precedence when iIgnoreSelectionFields is set to false. iFromSite Site : if specified Site concept must be active. From Site field selection field is filled with this value. (used when iIgnoreSelectionFields

	is false and NCRS is not applicable) iFromProductionOrder From Production Order field selection field is filled with this value. (used when iIgnoreSelectionFields is false and NCRS is not applicable) iFromOrderGroup From Order Group selection field is filled with this value. (used when iIgnoreSelectionFields is false and NCRS is not applicable) iToSite Site : if specified Site concept must be active. To Site field selection field is filled with this value. (used when iIgnoreSelectionFields is false and NCRS is not applicable) iToProductionOrder To Production Order field selection field is filled with this value. (used when iIgnoreSelectionFields is false and NCRS is not applicable) iToOrderGroup To Order Group selection field is filled with this value. (used when iIgnoreSelectionFields is false and NCRS is not applicable) Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 Error Occurred

ProductionOrder.StartDetail

DLL:	tiextsfapi This function is available from KB2280731.
Syntax:	<pre>long ProductionOrder.StartDetail(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcsite iSite, domain tcpdno iProductionOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts Details session Production Order (tisfc0101s000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the

	child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used iSessionIndex Not used iQueryExtend Not used iSite Optional - Mandatory when the Sites concept is active iProductionOrder Production Order - Mandatory Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 Error Occurred

ProductionOrder.StartInventoryOverview

DLL:	tiextsfapi
Syntax:	<pre> long ProductionOrder.StartInventoryOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcsite iSite, domain tcpdno iProductionOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Inventory by Production Order (tisfc0516m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used. iSessionIndex Not used. iQueryExtend Not used. iSite Site (Mandatory if active). iProductionOrder Production Order.

	Output: oExceptionMessage oExceptionID Return: 0 <> 0	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Session started. Otherwise.

ProductionOrder.StartMultiMain

DLL:	tiextsfapi This function is available from KB2280731 .	
Syntax:	<pre>long ProductionOrder.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcsite iSite, domain tcpdno iProductionOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: This function starts Multi Main session Production Order (tisfc0101m100). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used iSessionIndex Not used iQueryExtend Not used iSite Optional - Mandatory when the Sites concept is active iProductionOrder Production Order - Mandatory Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID	

	<code>oExceptionID</code> An ID that refers to all error information. Use the functions in <code>Exception</code> to get all relevant information. Return: 0 Session started <> 0 Error Occurred

ProductionOrder.StartOperationAddition

DLL:	<code>tiextsfapi</code> This function is available from KB3511148.
Syntax:	<pre>long ProductionOrder.StartOperationAddition(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcopno iPrecedingOperation, domain tcopno iNextOperation, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: With this Public Interface, a production order is prepared for having operation(s) added to the production order's operation network at a given location, as identified by its preceding and its next operation. To ensure Data integrity it is required that operations are only inserted with this preparation. After finishing the insert of a new operation, the function <code>ProductionOrder.FinishOperationAddition()</code> must be called to reset the system to normal. Start Example Usage: <pre>domain tcsite site domain tcpdno production.order domain tcopno operation.to.add domain tcopno preceding.operation domain tcopno subsequent.operation operation.to.add = 15 preceding.operation = 10 subsequent.operation = 20 ret = ProductionOrder.StartOperationAddition(site, production.order, preceding.operation, subsequent.operation, exception.message, exception.id) db.retry.point() /* insert new operation via DAL operations. dal.new.object("tisfc010") dal.set.field("tisfc010.pdno", production.order) dal.set.field("tisfc010.opno", operation.to.add) dal.set.field("tisfc010.nopr", subsequent.operation)</pre>

	<pre> /* set other required fields dal.set.field ("tisfc010.... dal.save.object("tisfc010") <> 0 then /* something went wrong endif commit.transaction() /* Finish operation addition ret = ProductionOrder.FinishOperationAddition(site, production.order, exception.message, exception.id) End Example Usage. </pre>
Input:	iSite Site (mandatory when the Site concept is active). iProductionOrder Production Order (mandatory and must be in iSite). iPrecedingOperation The operation of the current operation network that must precede the operation to be added. If 0 is given, the operation to be added will have no predecessor operation. iNextOperation The operation of the current operation network that must succeed the operation to be added. If 0 is given, the added operation must be the last in order.
Output:	oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information.
Return:	0 Able to add operation(s) to the production order. <> 0 Errors occurred.

ProductionOrder.StartPrintDocuments

DLL:	tiextsfapi This function is available from KB2274307.
Syntax:	<pre> long ProductionOrder.StartPrintDocuments(long iStartMode, boolean iIgnoreSelectionFields, domain tcsite iSite, domain tcpdno iFromProductionOrder, </pre>

	domain tcpdno iFromOrderGroup, domain tcpdno iToProductionOrder, domain tcpdno iToOrderGroup, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function starts the session Print Production Order Documents(tisfc0408m000). Input: iStartMode Not Used. iIgnoreSelectionFields This argument determines whether user defaults are used. User defaults should take precedence when iIgnoreSelectionFields is set to false. iSite Site (mandatory) if Site concept is active. iFromProductionOrder From Production Order field selection field is filled with this value. (used when iIgnoreSelectionFields is false and NCRS is not applicable) iFromOrderGroup From Order Group selection field is filled with this value. (used when iIgnoreSelectionFields is false and NCRS is not applicable) iToProductionOrder To Production Order field selection field is filled with this value. (used when iIgnoreSelectionFields is false and NCRS is not applicable) iToOrderGroup To Order Group selection field is filled with this value. (used when iIgnoreSelectionFields is false and NCRS is not applicable) Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 Error Occurred

ProductionOrder.StartReportOperationsCompleted

DLL:	tiextsfapi
	This function is available from KB2285288 .
Syntax:	<pre>long ProductionOrder.StartReportOperationsCompleted(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcsite iSite, domain tcpdno iProductionOrder, ref domain tcmcs.s999m oExceptionMessage mb,</pre>

	ref	long	oExceptionID)
Usage:	Expl: This function starts the session Report Operations Completed (tisfc0130m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used iSessionIndex Not used iQueryExtend Not used iSite Optional - Mandatory when the Sites concept is active iProductionOrder Production Order - Mandatory Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 Error Occurred		

ProductionOrder.StartReportOrdersCompleted

DLL:	tiextsfapi
	This function is available from KB2288027 .
Syntax:	<pre>long ProductionOrder.StartReportOrdersCompleted(long iStartMode, domain tcsite iSite, domain tcpdno iProductionOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the session Report Orders Completed (tisfc0120s000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session.

	MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iSite Optional - Mandatory when the Sites concept is active iProductionOrder Production Order - Mandatory Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 Error Occurred

ProductionOrder.StartResetStatus

DLL:	tiextsfapi
	This function is available from KB2298685 .
Syntax:	<pre>long ProductionOrder.StartResetStatus(long iStartMode, boolean iIgnoreSelectionFields, domain tcsite iSite, domain tcpdno iFromProductionOrder, domain tcpdno iToProductionOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the session Reset Production Orders Status (ticst0203m000). Input: iStartMode Not Used. iIgnoreSelectionFields This argument determines whether user defaults are used. User defaults should take precedence when iIgnoreSelectionFields is set to false. iSite Site selection field is filled with this value. Optional (used when iIgnoreSelectionFields is false and NCRS is not applicable) iFromProductionOrder From Production Order selection field is filled with this value. (used when iIgnoreSelectionFields is false and NCRS is not applicable) iToProductionOrder To Production Order selection field is filled with this value. (used when iIgnoreSelectionFields is false and NCRS is not applicable) Output: oExceptionMessage

	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 Error Occurred

ProductionOrder.UpdateMaterialToIssueQuantity

DLL:	tiextsfapi
	This function is available from KB2244728 .
Syntax:	<pre>long ProductionOrder.UpdateMaterialToIssueQuantity(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcpono iMaterialPosition, domain tcqcpl iQuantityToIssue, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this Public Interface to update the To Issue field for a Production Order Material Position. The update can also affect the value of the Subsequent Delivery field, following the logic of session Material to Issue for Production Orders (ticst0101m100). Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iSite Site (mandatory when the Site concept is active). iProductionOrder Production Order (mandatory, must be in iSite). iMaterialPosition Material Position (Mandatory). iQuantityToIssue The quantity to set the To Issue field to (Mandatory). Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Return: 0 The To Issue field has been updated. <> 0 The To Issue field has not been updated.

ProductionOrders.StartOverview

DLL:	tiextsfapi
Syntax:	<pre> long ProductionOrders.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcsite iSite, domain tcpdno iProductionOrder, domain tccprj iProject, domain tcitem iItem, domain tcutcs iActualDeliveryDate, domain tisfc.prcd iPriority, domain tcpdno iGroup, ref domain tcpdno oProductionOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Production Orders (tisfc0501m000) in overview mode. Input: <code>iStartMode</code> Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. <code>iStartFilter</code> Optional. Allowed values: showAll showInUse (default) <code>iSessionIndex</code> Optional. The index that will be used. Supported values: 1: sort by Production Order (default) 2: sort by Site, Production Order 3: sort by Project, Item 4: sort by Item, Delivery Date 5: sort by Priority 6: sort by Production Order Group <code>iQueryExtend</code> A specific query to be used when zooming to this session. <code>iSite</code> Site. Mandatory if <code>iStartMode = MODELESS</code> and <code>iSessionIndex = 2</code>. <code>iProductionOrder</code> Production Order Number <code>iProject</code> Project <code>iItem</code> Main Item <code>iActualDeliveryDate</code> Actual Delivery Date <code>iPriority</code> Priority <code>iGroup</code> Production Order Group Output: Variables below contain the values of the selected record. They are only filled if <code>iStartMode</code> is MODAL and 1 record has been selected. <code>oProductionOrder</code> Production Order Number <code>oExceptionMessage</code> - The last message if the return value is not equal to 0. If more than one message is given, these are present in the

	oExceptionID Return: 0 <> 0	oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Session started Otherwise.

Public Interfaces for ProductionOrderGroup

The following functions are available:

[ProductionOrderGroup.StartMultiMain](#)

ProductionOrderGroup.StartMultiMain

DLL:	tiextsfscapi
	This function is available from KB2280731 .
Syntax:	<pre>long ProductionOrderGroup.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcpdno iProductionOrderGroup, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the Multi Main session Production Order Groups - Production Orders (tisfc0501m100). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used iSessionIndex Not used iQueryExtend Not used

	<pre> iProductionOrderGroup Production Order Group - Mandatory Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 Error Occurred </pre>

Public Interfaces for ProductionOrderOperation

The following functions are available:

[ProductionOrderOperation.BackflushMaterialAndHours](#)

[ProductionOrderOperation.Block](#)

[ProductionOrderOperation.PlanOperation](#)

[ProductionOrderOperation.ReportProduct](#)

[ProductionOrderOperation.ReportProductV2](#)

[ProductionOrderOperation.ResetStatus](#)

[ProductionOrderOperation.SetRemainderStartDate](#)

[ProductionOrderOperation.SetStatusCompleted](#)

[ProductionOrderOperation.StartReportCompleted](#)

ProductionOrderOperation.BackflushMaterialAndHours

DLL:	tiextsfscapi
Syntax:	<pre> long ProductionOrderOperation.BackflushMaterialAndHours(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcopno iOperation, domain tisfc.wybf iSelectionMethod, boolean iCompletedOperationOnly, domain tcdate iBackflushDate, </pre>

	ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: Use this Public Interface to perform the backflushing for one or more Production Order Operations of a given Production Order. Transaction management is handled by this Public Interface. Pre: - Post: - Input: iSite Site (mandatory when the Site concept is active). iProductionOrder Production Order (mandatory, must be in iSite). iOperation Production Order Operation (mandatory if iSelectionMethod is not all). iSelectionMethod Operation(s) to handle: all upto one iCompletedOperationOnly Backflush Completed Operations only. iBackflushDate Backflush Date, cannot be in the future (0 = current date). Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Operation(s) have been backflushed. Backflushing could not be completed. Return: 0 <> 0

ProductionOrderOperation.Block

DLL:	tiextsfapi
	This function is available from KB2299065 .
Syntax:	<pre>long ProductionOrderOperation.Block(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcopno iOperation, domain tisfc.blcd iBlockingReason, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: With this Public Interface, the Production Order Operation will be blocked with the given blocking reason. ProductionOrderOperation.ResetStatus can be used to unblock an Operation. Pre: Retry point must be set. Post: Commit or abort the transaction.

	Input: iSite	Site (mandatory when the Site concept is active).
	iProductionOrder	Production Order (mandatory and must be in iSite).
	iOperation	Operation (mandatory)
	iBlockingReason	Blocking Reason (mandatory)
Output:	oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.
	oExceptionID	An ID that refers to all error information. Use the functions in Exception handling to get all relevant information.
Return: 0	<> 0	The operation was blocked.
		The operation was not blocked.

ProductionOrderOperation.PlanOperation

DLL:	tiextsfapi
	This function is available from KB2323309 .
Syntax:	<pre> long ProductionOrderOperation.PlanOperation(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcopno iOperation, domain tcutcs iStartOrEndDate, domain tcsetm iSetupTime, domain tisfc.prtm iProductionTime, domain tisfc.prtm iRemainingTime, domain tcwtm iQueueTime, domain tcwtm iWaitTime, domain tcwtm iMoveTime, ref domain tcutcs oTransportDateTo, ref domain tcutcs oStartDateQueue, ref domain tcutcs oProductionStartDate, ref domain tcutcs oRunStartDate, ref domain tcutcs oRemainingStartDate, ref domain tcutcs oStartDateWait, ref domain tcutcs oStartDateMove, ref domain tcutcs oTransportDateFrom, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function can be used to plan a Production Order Operation. Based on the given input times, the resulting Operation dates will be returned. Whether the Operation is planned forwards or backwards is defined on the Production Order by the Planning Method. Pre: NA Post: NA Input: iSite Site (Mandatory when the Site concept is active). iProductionOrder Production Order (Mandatory. Must be in iSite). iOperation Production Order Operation (Mandatory).

	iStartOrEndDate	The desired Start Date for the Operation if the planning method is forwards, or the End Date if the planning method is backwards.
	iSetupTime	The Setup Time for the Operation to be planned. Mandatory, use tisfc010.sutm, unless this is one of the times required to be changed.
	iProductionTime	The Production Time for the Operation to be planned. Mandatory, use tisfc010.prtm, unless this is one of the times required to be changed.
	iRemainingTime	The Remaining Time for the Operation to be planned. Mandatory, use tisfc010.retm, unless this is one of the times required to be changed.
	iQueueTime	The Queue Time for the Operation to be planned. Mandatory, use tisfc010.qutm, unless this is one of the times required to be changed.
	iWaitTime	The Wait Time for the Operation to be planned. Mandatory, use tisfc010.trdl, unless this is one of the times required to be changed.
	iMoveTime	The Move Time for the Operation to be planned. Mandatory, use tisfc010.mvtm, unless this is one of the times required to be changed.
Operation.	Output:	
	oTransportDateTo	The Transport Date To for the planned Operation.
	oStartDateQueue	The Start of the Queue Time for the planned
	oProductionStartDate	The Start Date of the Production for the planned Operation.
	oRunStartDate	The Run Start Date for the planned Operation.
	oRemainingStartDate	The Remaining Start Date for the planned
	oStartDateWait	The Start of the Wait Time for the planned
	oStartDateMove	The Start of the Move Time for the planned
	oTransportDateFrom	The Transport Date From for the planned Operation.
	oExceptionMessage	The last message if any message is found. If more than one message is given, these are present in the oExceptionID.
	oExceptionID	An ID that refers to the exception information. Use the functions in Exception to get all relevant information.
	Return: 0 / <> 0	

ProductionOrderOperation.ReportProduct

DLL:	tiextsfapi
Syntax:	long ProductionOrderOperation.ReportProduct(domain tcsite iSite, domain tcpdno iProductionOrder,

	domain tcopno iOperation, domain tcclot iLotCode, domain tiqep2 iProducedQuantity, domain tiqep2 iQuantityToInspect, domain tiqep2 iRejectedQuantity, domain tccdis iRejectReason, ref domain tcibd.sern iSerialArrayProduced() fixed, ref domain tcibd.sern iSerialArrayToInspect() fixed, ref domain tcibd.sern iSerialArrayRejected() fixed, domain tcutcs iCompletionDate, boolean iSetStatusToCompleted, ref domain tcclot oLotCode, ref boolean oOperationIsCompleted, ref boolean oErrorInReportProduct, ref boolean oErrorInWarehousing, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: With this Public Interface, a quantity on the specified production order operation can be reported as complete, rejected or sent to inspection, similar to session tisfc0130m000. This Public Interface consists of 2 transactions: 1. Report Product (including changing of status if requested) 2. Process the automatic activities on generated warehousing orders (if needed). This transaction will only be executed if: - the first action is successfully completed and committed(!), and - serveral other detailed functional conditions are met (e.g. the operation is the last operation, the end item has the direct receipt setting, etc). If the main item of the production order requires backflushing, then Backflushing will be executed, but only when the company or site setting for the backflushing method is 'Automatic'. When the setting of the backflushing method is 'Manual' or 'Interactive', then no backflushing will be done. If iSetStatusToCompleted is true, then the operation status will be set to 'Completed' (if possible). The output argument oOperationIsCompleted will indicate if it was possible change the status of the operation. Use the serial array arguments to pass information about serial numbers into this Public API. There is a separate serial array for each of the 3 reported quantity types (produced, to inspection and rejected). If filled, then the length of the array must equal the reported quantity. The array can be static or dynamic. Pass an empty array variable when no serial information must be used for the quantity type. Example of empty array variable dummy.serials.produced, which is defined as: domain tcibd.sern dummy.serials.produced(1). It is possible to use 1 and the same empty array variable for all 3 array arguments. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled inside

	the public interface.
Input:	iSite Site (mandatory when the Site concept is active). iProductionOrder Production Order (mandatory and must be in iSite). iOperation Operation (mandatory) iLotCode Code of existing or new lot or empty. When empty and a lot is required, then a new lot will be created. When the specified lot does not exist, then it will be created. iProducedQuantity The quantity that is reported as completed. iQuantityToInspect The quantity that is reported as send to Inspection. iRejectedQuantity The quantity that is reported as rejected. iRejectReason The reason for rejection. iSerialArrayProduced Array of Serial Numbers for the quantity produced. iSerialArrayToInspect Array of Serial Numbers for the quantity to inspect. iSerialArrayRejected Array of Serial Numbers for the quantity rejected. iCompletionDate Date of completion. Use zero to specify the current date. iSetStatusToCompleted If true then the status of the production order must be set to 'Completed'.
Output:	oLotCode Code of the lot that was actually used. oOperationIsCompleted True when the status of the operation was changed to Completed, else false. oErrorInReportProduct If true, then an error occurred in the Report Product transaction (see above). oErrorInWarehousing If true, then an error occurred in the Process Warehousing Orders transaction (see above). oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information.
Return: 0	The 2 transactions (see above) were successfully completed.
<> 0	Errors occurred in one of the two transactions. Use oErrorInReportProduct and oErrorInWarehousing to find out in which transaction the error occurred. If these are both false, then the error occurred during checking the input arguments.

ProductionOrderOperation.ReportProductV2

DLL:	tiextsfscapi
Syntax:	<pre> long ProductionOrderOperation.ReportProductV2(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcopno iOperation, domain tcclot iLotCode, domain tiqep2 iProducedQuantity, domain tiqep2 iQuantityToInspect, domain tiqep2 iRejectedQuantity, domain tccdis iRejectReason, domain tcuef.effn iEffectivityUnit, ref domain tcibd.sern iSerialArrayProduced() fixed, ref domain tcibd.sern iSerialArrayToInspect() fixed, ref domain tcibd.sern iSerialArrayRejected() fixed, domain tcutcs iCompletionDate, boolean iSetStatusToCompleted, ref domain tcclot oLotCode, ref boolean oOperationIsCompleted, ref boolean oErrorInReportProduct, ref boolean oErrorInWarehousing, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: With this Public Interface, a quantity on the specified production order operation can be reported as complete, rejected or sent to inspection, similar to session tisfc0130m000. This Public Interface consists of 2 transactions: 1. Report Product (including changing of status if requested) 2. Process the automatic activities on generated warehousing orders (if needed). This transaction will only be executed if: - the first action is successfully completed and committed(!), and - serveral other detailed functional conditions are met (e.g. the operation is the last operation, the end item has the direct receipt setting, etc). If the main item of the production order requires backflushing, then Backflushing will be executed, but only when the company or site setting for the backflushing method is 'Automatic'. When the setting of the backflushing method is 'Manual' or 'Interactive', then no backflushing will be done. If iSetStatusToCompleted is true, then the operation status will be set to 'Completed' (if possible). The output argument oOperationIsCompleted will indicate if it was possible change the status of the operation. Use the serial array arguments to pass information about serial numbers into this Public API. There is a separate serial array for each of the 3 reported quantity types (produced, to inspection and rejected). If filled, then the length of the array must equal the reported quantity. The array can be static or dynamic. Pass an empty array variable when no serial information must be used for the quantity type. Example of empty array variable

	dummy.serials.produced, which is defined as: domain tcibd.sern dummy.serials.produced(1). It is possible to use 1 and the same empty array variable for all 3 array arguments.
Pre:	There should be no pending logical transaction before calling this function.
Post:	No need to commit or abort the process, that is handled inside the public interface.
Input:	iSite Site (mandatory when the Site concept is active). iProductionOrder Production Order (mandatory and must be in iSite). iOperation Operation (mandatory) iLotCode Code of existing or new lot or empty. When empty and a lot is required, then a new lot will be created. When the specified lot does not exist, then it will be created. iProducedQuantity The quantity that is reported as completed. iQuantityToInspect The quantity that is reported as send to Inspection. iRejectedQuantity The quantity that is reported as rejected. iRejectReason The reason for rejection. iEffectivityUnit The Effectivity Unit. Not used when set to 0. iSerialArrayProduced Array of Serial Numbers for the quantity produced. iSerialArrayToInspect Array of Serial Numbers for the quantity to inspect. iSerialArrayRejected Array of Serial Numbers for the quantity rejected. iCompletionDate Date of completion. Use zero to specify the current date. iSetStatusToCompleted If true then the status of the production order must be set to 'Completed'.
Output:	oLotCode Code of the lot that was actually used. oOperationIsCompleted True when the status of the operation was changed to Completed, else false. oErrorInReportProduct If true, then an error occurred in the Report Product transaction (see above). oErrorInWarehousing If true, then an error occurred in the Process Warehousing Orders transaction (see above). oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information.
Return: 0	The 2 transactions (see above) were successfully completed.
<> 0	Errors occurred in one of the two transactions.

	Use oErrorInReportProduct and oErrorInWarehousing to find out in which transaction the error occurred. If these are both false, then the error occurred during checking the input arguments.

ProductionOrderOperation.ResetStatus

DLL:	tiextsfapi												
Syntax:	<pre> long ProductionOrderOperation.ResetStatus (domain tcsite iSite, domain tcpdno iProductionOrder, domain tcopno iOperation, boolean iResetPlannedQuantities, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>												
Usage:	Expl: Use this public interface to reset the status of a completed Production Order Operation, as in session tisfc0130m000. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: <table> <tr> <td>iSite</td><td>Site (mandatory when the Site concept is active).</td></tr> <tr> <td>iProductionOrder</td><td>Production Order (mandatory, must be in iSite).</td></tr> <tr> <td>iOperation</td><td>Production Order Operation (mandatory, its status must be blocked completed or closed).</td></tr> <tr> <td>iResetPlannedQuantities</td><td>Relevant if the reported quantities are zero. If true, resets the planned quantities and material quantities to the estimated values. If false, the planned quantities are not reset.</td></tr> </table> Output: <table> <tr> <td>oExceptionMessage</td><td>The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.</td></tr> <tr> <td>oExceptionID</td><td>An ID that refers to all error information. Use the functions in Exception handling to get all relevant information.</td></tr> </table> Return: 0 <> 0 Operation is reset. Operation could not be reset.	iSite	Site (mandatory when the Site concept is active).	iProductionOrder	Production Order (mandatory, must be in iSite).	iOperation	Production Order Operation (mandatory, its status must be blocked completed or closed).	iResetPlannedQuantities	Relevant if the reported quantities are zero. If true, resets the planned quantities and material quantities to the estimated values. If false, the planned quantities are not reset.	oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.	oExceptionID	An ID that refers to all error information. Use the functions in Exception handling to get all relevant information.
iSite	Site (mandatory when the Site concept is active).												
iProductionOrder	Production Order (mandatory, must be in iSite).												
iOperation	Production Order Operation (mandatory, its status must be blocked completed or closed).												
iResetPlannedQuantities	Relevant if the reported quantities are zero. If true, resets the planned quantities and material quantities to the estimated values. If false, the planned quantities are not reset.												
oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.												
oExceptionID	An ID that refers to all error information. Use the functions in Exception handling to get all relevant information.												

ProductionOrderOperation.SetRemainderStartDate

DLL:	tiextsfscapi
Syntax:	<pre> long ProductionOrderOperation.SetRemainderStartDate(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcopno iOperation, domain tcutcs iRemainderStartDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface will update the Remainder Start Date for a Production Order Operation. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iSite Site (mandatory if active). iProductionOrder Production Order (mandatory). iOperation Production Order Operation (mandatory). iRemainderStartDate The Remainder Start Date for the Production Order Operation (mandatory). Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Remainder Start Date has been updated. <> 0 - Remainder Start Date has not been updated.

ProductionOrderOperation.SetStatusCompleted

DLL:	tiextsfscapi
Syntax:	<pre> long ProductionOrderOperation.SetStatusCompleted(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcopno iOperation, domain tcutcs iCompletionDate, boolean iCheckFullReportedQuantity, ref boolean oOperationIsCompleted, ref boolean oErrorInChangeStatus, ref boolean oErrorInWarehousing, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: With this Public Interface, the Production Order Operation status of the specified Operation will be set to >¼À~Completed>¼À™. This Public Interface consists of 2 transactions: 1. Change status.

	2. Process the automatic activities on the generated warehousing orders. This transaction will only be executed when the first action is successfully completed and committed and only when handling the last operation and backflusing was executed in the first transaction. The output argument oOperationIsCompleted indicates if the status of the operation was changed. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iSite Site (mandatory when the Site concept is active). iProductionOrder Production Order (mandatory and must be in iSite). iOperation Operation (mandatory) iCompletionDate Date of completion. Use zero to specify the current date. iCheckFullReportedQuantity If true, then the status change is only done when the planned quantity is produced. Output: oOperationIsCompleted If true when the status of the operation was successfully changed to Completed else false. oErrorInChangeStatus If true, then an error occurred in the Change Status transaction (see above). oErrorInWarehousing If true, then an error occurred in the Process Warehousing Orders transaction (see above). oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Return: 0 The 2 transactions (see above) were successfully completed. <> 0 Errors occurred in one of the two transactions. Use oErrorInChangeStatus and oErrorInWarehousing to find out in which transaction the error occurred. If these are both false, then the error occurred during checking the input arguments.

ProductionOrderOperation.StartReportCompleted

DLL:	tiextsfapi
	This function is available from KB3564305 .
Syntax:	<pre> long ProductionOrderOperation.StartReportCompleted(long iStartMode, domain tcsite iSite, domain tcpdno iProductionOrder, domain tcopno iOperation, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Report Operations Completed (tisfc0130m000) for the given Production Order and Operation. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iSite Site (mandatory when the Sites concept is active). iProductionOrder Production Order (Mandatory). iOperation Operation (Mandatory). Output: oExceptionMessage The last message, if any message is found. If more than one message is found, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started. <> 0 Otherwise.

Public Interfaces for ProductionOrderReject

The following functions are available:

[ProductionOrderReject.MoveToQuarantine](#)

[ProductionOrderReject.Scrap](#)

ProductionOrderReject.MoveToQuarantine

DLL:	tiextsfscapi																										
Syntax:	<pre> long ProductionOrderReject.MoveToQuarantine(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcopno iOperation, domain tiqep2 iQuantity, ref domain tcibd.sern iSerialArray() fixed, domain tccwar iQuarantineWarehouse, domain tiloca iWarehouseLocation, domain tcuef.effn iEffectivityUnit, domain tcclot iLotCode, domain tcorno iNCMReport, domain tctxtn iTextNumber, const string iTextString(), domain tcyesno iDirectProcessInbound, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>																										
Usage:	Expl: This function moves a rejected end item quantity of a production order operation to a quarantine warehouse, as in session tisfc0209m000. Retry point and commit / abort transaction are executed within this Public Interface. Input: <table> <tr> <td>iSite</td><td>Site (mandatory when the Site concept is active).</td></tr> <tr> <td>iProductionOrder</td><td>Production Order (mandatory and must be in iSite).</td></tr> <tr> <td>iOperation</td><td>Operation (mandatory).</td></tr> <tr> <td>iQuantity</td><td>Quantity (mandatory).</td></tr> <tr> <td>iSerialArray</td><td>Array of Serial Numbers. Pass a filled array (static or dynamic) when the main item is serialized and serial numbers are required; the length of the array must equal i.quantity. Pass an empty array variable when the main item is not serialized. E.g. pass the variable dummy.serials, which is defined as: domain tcibd.sern dummy.serials(1)</td></tr> <tr> <td>iQuarantineWarehouse</td><td>The warehouse to which the quarantined end item quantity is sent (mandatory).</td></tr> <tr> <td>iWarehouseLocation</td><td>Location in iQuarantineWarehouse (optional).</td></tr> <tr> <td>iEffectivityUnit</td><td>Effectivity Unit (optional).</td></tr> <tr> <td>iLotCode</td><td>An existing Lot code (mandatory for lot controlled items)</td></tr> <tr> <td>iNCMReport</td><td>The non-conformance material report to be updated (optional)</td></tr> <tr> <td>iTextNumber</td><td>Text number (optional) which will be copied into a new text and linked to the Production Warehouse Order.</td></tr> <tr> <td>iTextString</td><td>Text string (optional) that will stored as Text Number and linked to the Production Warehouse Order. Must be empty when iText is filled and vice versa.</td></tr> <tr> <td>iDirectProcessInbound</td><td>If Yes, then the inbound procedure</td></tr> </table>	iSite	Site (mandatory when the Site concept is active).	iProductionOrder	Production Order (mandatory and must be in iSite).	iOperation	Operation (mandatory).	iQuantity	Quantity (mandatory).	iSerialArray	Array of Serial Numbers. Pass a filled array (static or dynamic) when the main item is serialized and serial numbers are required; the length of the array must equal i.quantity. Pass an empty array variable when the main item is not serialized. E.g. pass the variable dummy.serials, which is defined as: domain tcibd.sern dummy.serials(1)	iQuarantineWarehouse	The warehouse to which the quarantined end item quantity is sent (mandatory).	iWarehouseLocation	Location in iQuarantineWarehouse (optional).	iEffectivityUnit	Effectivity Unit (optional).	iLotCode	An existing Lot code (mandatory for lot controlled items)	iNCMReport	The non-conformance material report to be updated (optional)	iTextNumber	Text number (optional) which will be copied into a new text and linked to the Production Warehouse Order.	iTextString	Text string (optional) that will stored as Text Number and linked to the Production Warehouse Order. Must be empty when iText is filled and vice versa.	iDirectProcessInbound	If Yes, then the inbound procedure
iSite	Site (mandatory when the Site concept is active).																										
iProductionOrder	Production Order (mandatory and must be in iSite).																										
iOperation	Operation (mandatory).																										
iQuantity	Quantity (mandatory).																										
iSerialArray	Array of Serial Numbers. Pass a filled array (static or dynamic) when the main item is serialized and serial numbers are required; the length of the array must equal i.quantity. Pass an empty array variable when the main item is not serialized. E.g. pass the variable dummy.serials, which is defined as: domain tcibd.sern dummy.serials(1)																										
iQuarantineWarehouse	The warehouse to which the quarantined end item quantity is sent (mandatory).																										
iWarehouseLocation	Location in iQuarantineWarehouse (optional).																										
iEffectivityUnit	Effectivity Unit (optional).																										
iLotCode	An existing Lot code (mandatory for lot controlled items)																										
iNCMReport	The non-conformance material report to be updated (optional)																										
iTextNumber	Text number (optional) which will be copied into a new text and linked to the Production Warehouse Order.																										
iTextString	Text string (optional) that will stored as Text Number and linked to the Production Warehouse Order. Must be empty when iText is filled and vice versa.																										
iDirectProcessInbound	If Yes, then the inbound procedure																										

	Output: oExceptionMessage oExceptionID Return: 0 <> 0	will be executed automatically. The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Reject is moved to quarantine Reject could not be moved to quarentine.

ProductionOrderReject.Scrap

DLL:	tiextsfscapi
Syntax:	<pre>long ProductionOrderReject.Scrap(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcopno iOperation, domain tiqep2 iQuantity, ref domain tcibd.sern iSerialArray() fixed, domain tcuef.effn iEffectivityUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this public interface to scrap a quantity of rejected end item of an operation, as in session tisfc0211m000. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iSite Site (mandatory when the Site concept is active). iProductionOrder Production Order (mandatory and must be in iSite). iOperation Operation (mandatory). iQuantity Quantity (mandatory). For serialized items, this double argument must have an integer value. iSerialArray Array of Serial Numbers. Pass a filled array (static or dynamic) when the main item is serialized and serial numbers are required; the length of the array must equal i.Quantity. Pass an empty array variable when the main item is not serialized. E.g. pass the variable dummy.serials, which is defined as: domain tcibd.sern dummy.serials(1) iEffectivityUnit Effectivity Unit (optional). Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.

	<code>oExceptionID</code> Return: 0 <> 0	An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Rejected quantity is scrapped Rejected quantity could not be scrapped

Public Interfaces for WorkCenter

The following functions are available:

[WorkCenters.StartOverview](#)

WorkCenters.StartOverview

DLL:	tiextrouapi
	This function is available from KB2274105 .
Syntax:	<pre> long WorkCenters.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcsite iSite, domain tccwoc iWorkCenter, ref domain tcsite oSite, ref domain tccwoc oWorkCenter, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Work Centers in overview mode (tirou010lm000). Input: <code>iStartMode</code> Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. <code>iStartFilter</code> Not Used. <code>iSessionIndex</code> 1 - The session will start with the first session index: Work Center 2 - The session will start with the second session index:

	Site, Work Center Second session index is only possible when Job Shop by Site is active. iQueryExtend A specific query to be used when zooming to this session. Primary Key fields: iSite Site (Mandatory if iSessionIndex = 2) iWorkCenter Work Center Output: Variables below contain the values of the selected record. They are only filled if iStartMode is MODAL and 1 record has been selected. oSite Site oWorkCenter Work Center oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 Otherwise.

Public Interfaces for JobShopTask

The following functions are available:

[JobShopTasks.StartOverview](#)

JobShopTasks.StartOverview

DLL:	tiextrouapi
	This function is available from KB2274111 .
Syntax:	<pre>long JobShopTasks.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tctano iTask, domain tcseak iSearchKey mb,</pre>

	<pre> ref domain tctano oTask, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Tasks(Reference Operations) in overview mode (tirou0103m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex 1 - The session will start with the first session index: Task 2 - The session will start with the second session index: Search Key, Task. iQueryExtend A specific query to be used when zooming to this session. Primary Key fields: iTask Task iSearchKey Search Key (Mandatory if iSessionIndex = 2) Output: Variables below contain the values of the selected record. They are only filled if iStartMode is MODAL and 1 record has been selected. oTask Task oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 Otherwise.

Public Interfaces for UtilizationByWeek

The following functions are available:

[UtilizationByWeek.StartOverview](#)

UtilizationByWeek.StartOverview

DLL:	tiextsfscapi This function is available from KB2300431 .
Syntax:	<pre> long UtilizationByWeek.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccwoc iWorkCenter, domain tcyrno iProductionYear, domain tcweek iProductionWeek, domain tcutcs iSetupStartDate, domain tirou.mcno iMachine, domain tcpdno iProductionOrder, domain tcopno iOperation, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Utilization by Week (tisfc1502m000) in overview mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not used. iSessionIndex Optional. The index that will be used. Supported values: 2: sort by Production Year, Production Week, Setup Start Date, Work Center, Machine, Production Order and Operation. 5: sort by Production Year, Production Week, Setup Start Date, Machine, Production Order and Operation. iQueryExtend A specific query to be used when zooming to this session. Optional. iWorkCenter Work Center - Optional iProductionYear Production Year - Optional iProductionWeek Production Week - Optional iSetupStartDate Setup Start Date - Optional iMachine Machine - Optional iProductionOrder Production Order - Optional iOperation Operation - Optional Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started. <> 0 Otherwise.

Public Interfaces for SubcontractedOperation

The following functions are available:

[SubcontractedOperations.StartDetail](#)

[SubcontractedOperations.StartOverview](#)

SubcontractedOperations.StartDetail

DLL:	tiextsfapi This function is available from KB2323105 .
Syntax:	<pre> long SubcontractedOperations.StartDetail(long iStartMode, domain tcpdno iProductionOrder, domain tcopno iOperation, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Subcontracted Operations (tisfc2110m000) in deatil mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Primary Key Fields: iProductionOrder Production Order - Mandatory. iOperation Operation - Mandatory. Output: oExceptionMessage The last message if any message is found. If more than one message is found, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started. <> 0 Otherwise.

SubcontractedOperations.StartOverview

DLL:	tiextsfapi
	This function is available from KB2323105 .
Syntax:	<pre> long SubcontractedOperations.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcpdno iProductionOrder, domain tcopno iOperation, domain tccom.bpid iSubcontractor, domain tcsite iSubcontractorSite, ref domain tcpdno oProductionOrder, ref domain tcopno oOperation, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Subcontracted Operations (tisfc2110m000) in overview mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Specifies the table-index that is to be used. Optional. Standard supported values: 1: sort by Production Order, Operation (default). 2: sort by Subcontractor, Subcontractor Site, Production Order, Operation. iQueryExtend A specific query to be used when zooming to this session. Optional. iProductionOrder Production Order - Optional. iOperation Operation - Optional. iSubcontractor Subcontractor - Optional. iSubcontractorSite Subcontractor Site - Optional. Output: Variables below contain the values of the selected record. They are only filled if iStartMode is MODAL and 1 record has been selected. oProductionOrder Production Order. oOperation Operation. oExceptionMessage The last message if any message is found. If more than one message is found, these are present in the oExceptionID. oExceptionID An ID that refers to the exception

	<pre>Return: 0 <> 0</pre>	<pre>information. Use the functions in Exception to get all relevant information. Session started. Otherwise.</pre>

Chapter 20 Public Interfaces for Subcontracting

Public Interfaces for SubcontractingModel

The following functions are available:

[SubcontractingModel.ApproveRevision](#)

[SubcontractingModel.SetUseForCosting](#)

[SubcontractingModel.SetUseForPlanning](#)

[SubcontractingModel.ValidateRevision](#)

SubcontractingModel.ApproveRevision

DLL:	tiextsubapi
	This function is available from KB2282924 .
Syntax:	<pre> long SubcontractingModel.ApproveRevision(domain tcsite iProductionSite, domain tcitem iProduct, domain tccom.bpid iSubcontractor, domain tcsite iSubcontractorSite, domain tccom.bpid iShipFromBusinessPartner, domain tirpt.revi iRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this Public Interface to Approve a Subcontracting Model Revision. The functionality is the same as the Form Command "Approve Revision" in session in session Product Subcontracting List (tisub1100m000). This Public Interface can only be used if the Sites concept is active. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iProductionSite Site (mandatory).

	iProduct Product (mandatory). iSubcontractor Subcontractor (mandatory). iSubcontractorSite Subcontractor Site (mandatory if Resource by Site is active). iShipFromBusinessPartner Ship From Business Partner (mandatory if Resource by Site is not active, derived from iSubcontractorSite otherwise). iRevision Subcontracting Model Revision (mandatory). Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Return: 0 The Subcontracting Model Revision has been approved. <> 0 The Subcontracting Model Revision has not been approved.

SubcontractingModel.SetUseForCosting

DLL:	tiextsubapi This function is available from KB2282924 .
Syntax:	<pre> long SubcontractingModel.SetUseForCosting(domain tcsite iProductionSite, domain tcitem iProduct, domain tccom.bpid iSubcontractor, domain tcsite iSubcontractorSite, domain tccom.bpid iShipFromBusinessPartner, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this Public Interface to set the Use for Costing flag for a Subcontracting Model. For all Subcontracting Model revisions for this Product and Site, the use for Costing flag will be set. For all other Subcontracting Models related to this Product and Site, the Use for Costing flag will be cleared. This function follows the same logic as the form command "Use for Costing" in session Product Subcontracting List (tisub1100m000). This Public Interface can only be used if the Sites concept is active. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iProductionSite Site (mandatory).

	iProduct Product (mandatory). iSubcontractor Subcontractor (mandatory). iSubcontractorSite Subcontractor Site (mandatory if Resource by Site is active). iShipFromBusinessPartner Ship From Business Partner (mandatory if Resource by Site is not active, derived from iSubcontractorSite otherwise). Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Return: 0 The Use for Costing flag has been set. <> 0 The Use for Costing flag has not been set.

SubcontractingModel.SetUseForPlanning

DLL:	tiextsubapi
	This function is available from KB2282924 .
Syntax:	<pre>long SubcontractingModel.SetUseForPlanning(domain tcsite iProductionSite, domain tcitem iProduct, domain tccom.bpid iSubcontractor, domain tcsite iSubcontractorSite, domain tccom.bpid iShipFromBusinessPartner, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this Public Interface to set the Use for Planning flag for a Subcontracting Model. For all Subcontracting Model revisions for this Product and Site, the use for Planning flag will be set. For all other Subcontracting Models related to this Product and Site, the Use for Planning flag will be cleared. If the resource by site concept is active, the model needs to be have the Available for Planning flag turned on. This function follows the same logic as the form command "Use for Planning" in session Product Subcontracting List (tisubl100m000). This Public Interface can only be used if the Sites concept is active. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iProductionSite Site (mandatory). iProduct Product (mandatory).

	<pre> iSubcontractor Subcontractor (mandatory). iSubcontractorSite Subcontractor Site (mandatory if Resource by Site is active). iShipFromBusinessPartner Ship From Business Partner (mandatory if Resource by Site is not active, derived from iSubcontractorSite otherwise). Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Return: 0 The Use for Planning flag has been set. <> 0 The Use for Planning flag has not been set. </pre>

SubcontractingModel.ValidateRevision

DLL:	tiextsubapi This function is available from KB2282924 .
Syntax:	<pre> long SubcontractingModel.ValidateRevision(domain tcsite iProductionSite, domain tcitem iProduct, domain tccom.bpid iSubcontractor, domain tcsite iSubcontractorSite, domain tccom.bpid iShipFromBusinessPartner, domain tirpt.revi iRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this Public Interface to Validate a Subcontracting Model Revision. All validation messages will be returned in oExceptionID. It can be filled with non-blocking information or warning messages while the revision is valid. Messages can also be returned if the return value is 0. The validation is identical to that of session Validate Product Subcontracting Models (tisub1200m000). This Public Interface can only be used if the Sites concept is active. Pre: - Post: - Input: iProductionSite Site (mandatory). iProduct Product (mandatory). iSubcontractor Subcontractor (mandatory). iSubcontractorSite Subcontractor Site (mandatory if Resource by Site is active).

	<code>iShipFromBusinessPartner</code> <code>iRevision</code> Output: <code>oExceptionMessage</code> <code>oExceptionID</code> Return: 0 <code><> 0</code>	Ship From Business Partner (mandatory if Resource by Site is not active, derived from <code>iSubcontractorSite</code> otherwise). Subcontracting Model Revision (mandatory). The last message if the return value is not equal to 0. If more than one message is given, these are present in the <code>oExceptionID</code>. An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. The Subcontracting Model Revision is valid. Non-blocking messages can still be returned. The Subcontracting Model Revision is not valid.

Chapter 21 Public Interfaces for Repetitive Manufacturing

Public Interfaces for Repetitive

The following functions are available:

[Repetitive.GetSettings](#)

Repetitive.GetSettings

DLL:	tiextrptapi
Syntax:	<pre> long Repetitive.GetSettings(domain tcncmp iLogisticCompany, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves master data settings for job shop (table tirpt000 / tirpt080). One or more setting fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tirpt000), or site level (settings table tirpt080), depending on: - implementation phase Sites, - input argument iSite, - the specified input field(s) (mnemonic). Note: This function uses variable arguments, for input and output. Data is retrieved via name-value pairs: specify the field (input) and a variable for the field value (output). Example: when the settings 'allow offloading' (alof) and 'series for EP' (ssep) are needed,

	the function must be called with the following arguments: <pre> Repetitive.GetSettings(/* Fixed arguments: company, --> input site, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output /* Variable arguments: "alof", --> input allow.offlooding, --> output "ssep", --> input series.for.ep) --> output </pre> Pre: None Post: None Input: <table> <tr> <td>iLogisticCompany</td><td>- Logistic Company: Mandatory</td></tr> <tr> <td>iSite</td><td>- Site: Not Mandatory</td></tr> <tr> <td>iForceRead</td><td>- Option to force reading new data in stead of using cached information</td></tr> <tr> <td>iContextIsMasterData</td><td>- True/False (not used)</td></tr> <tr> <td>...</td><td>- The field mnemonic of the required field(s)</td></tr> </table> Output: <table> <tr> <td>oExceptionMessage</td><td>- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID</td></tr> <tr> <td>oExceptionID</td><td>- An ID that refers to all error information. Use the functions in Exception to get all relevant information.</td></tr> <tr> <td>...</td><td>- The value of the required field(s).</td></tr> </table> Return: 0 <> 0 <table> <tr> <td>0</td><td>- Data read</td></tr> <tr> <td><> 0</td><td>- Otherwise.</td></tr> </table>	iLogisticCompany	- Logistic Company: Mandatory	iSite	- Site: Not Mandatory	iForceRead	- Option to force reading new data in stead of using cached information	iContextIsMasterData	- True/False (not used)	...	- The field mnemonic of the required field(s)	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.	...	- The value of the required field(s).	0	- Data read	<> 0	- Otherwise.
iLogisticCompany	- Logistic Company: Mandatory																				
iSite	- Site: Not Mandatory																				
iForceRead	- Option to force reading new data in stead of using cached information																				
iContextIsMasterData	- True/False (not used)																				
...	- The field mnemonic of the required field(s)																				
oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID																				
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.																				
...	- The value of the required field(s).																				
0	- Data read																				
<> 0	- Otherwise.																				

Chapter 22 Public Interfaces for Assembly

Public Interfaces for Assembly

The following functions are available:

[Assembly.CalculatePartRequirements](#)

[Assembly.GetSettings](#)

[Assembly.UpdateSegmentSchedules](#)

Assembly.CalculatePartRequirements

DLL:	tiextascapi														
Syntax:	<pre> long Assembly.CalculatePartRequirements (domain tcncmp iCompany, domain cpitem iPlanItem, domain tiutcd iDateFrom, domain tiutcd iDateTo, domain tcyesno iUpdateSegmentSchedules, domain tcyesno iUpdatePlanItems, domain tiasc.ardt.typ iReferenceDateType, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>														
Usage:	Expl: Use this Public Interface to calculate the Assembly Part Requirements for a Product Variant whose Assembly Parts are not yet allocated. Transaction management is handled in this function. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process; that is handled within the function. Input: <table> <tr> <td>iCompany</td><td>- Company: Mandatory.</td></tr> <tr> <td>iPlanItem</td><td>- Plan Item: Mandatory.</td></tr> <tr> <td>iDateFrom</td><td>- Date From.</td></tr> <tr> <td>iDateTo</td><td>- Date To.</td></tr> <tr> <td>iUpdateSegmentSchedules</td><td>- Update Segment Schedules: Mandatory.</td></tr> <tr> <td>iUpdatePlanItems</td><td>- Update Plan Items: Mandatory.</td></tr> <tr> <td>iReferenceDateType</td><td>- Reference Date Type: Mandatory.</td></tr> </table>	iCompany	- Company: Mandatory.	iPlanItem	- Plan Item: Mandatory.	iDateFrom	- Date From.	iDateTo	- Date To.	iUpdateSegmentSchedules	- Update Segment Schedules: Mandatory.	iUpdatePlanItems	- Update Plan Items: Mandatory.	iReferenceDateType	- Reference Date Type: Mandatory.
iCompany	- Company: Mandatory.														
iPlanItem	- Plan Item: Mandatory.														
iDateFrom	- Date From.														
iDateTo	- Date To.														
iUpdateSegmentSchedules	- Update Segment Schedules: Mandatory.														
iUpdatePlanItems	- Update Plan Items: Mandatory.														
iReferenceDateType	- Reference Date Type: Mandatory.														

	Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Assembly Part Requirement Calculated. <> 0 - Otherwise.

Assembly.GetSettings

DLL:	tiextascapi
Syntax:	<pre>long Assembly.GetSettings(domain tcncmp iLogisticCompany, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...)</pre>
Usage:	Expl: This function retrieves master data settings for tool requirement planning (table tiasc000 / tiasc080). One or more setting fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tiasc000), or site level (settings table tiasc080), depending on: - implementation phase of Sites, - input argument iSite, - the specified input field(s) (mnemonic). Note: This function uses variable arguments, for input and output. Data is retrieved via name-value pairs: specify the field (input) and a variable for the field value (output). Example: when the settings 'allaction horizon' (altf) and 'assembly order number group' (along) are needed, the function must be called with the following arguments: <pre>Assembly.GetSettings(/* Fixed arguments: company, --> input site, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output /* Variable arguments:</pre>

	<pre> "alft", --> input alloc.horizon, --> output "aong", --> input number.group) --> output </pre>
Pre:	None
Post:	None
Input:	<pre> iLogisticCompany - Logistic Company: Mandatory iSite - Site: Not Mandatory iForceRead - Option to force reading new data in stead of using cached information iContextIsMasterData - True/False (not used) ... - The field mnemonic of the required field(s) </pre>
Output:	<pre> oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field(s). Return: 0 - Data read <> 0 - Otherwise. </pre>

Assembly.UpdateSegmentSchedules

DLL:	tiextascapi
Syntax:	<pre> long Assembly.UpdateSegmentSchedules(domain cpcom.plnc iScenario, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl: Update the Assembly Segment Schedules in a Planning Scenario. Transaction management is handled within the Public Interface. Pre: Scenario must be initialized. There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process; that is handled within the function. Input: iScenario - Planning Scenario (Mandatory). Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. </pre>

	Return: 0 <> 0	- If the Segment Schedules were properly updated or no updates were required. - Otherwise.

Public Interfaces for AssemblyOrder

The following functions are available:

[AssemblyOrder.Create](#)

[AssemblyOrder.CreateV2](#)

AssemblyOrder.Create

DLL:	tiextascapi
Syntax:	<pre> long AssemblyOrder.Create(domain tcitem iAssemblyItem, domain tccpva iProductVariant, domain tiutcd iPlannedOfflineDate, domain tiasc.ardt.typ iReferenceDateType, domain tcseri iAssemblyOrderSeries, ref domain tcorno oAssemblyOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: Use this Public interface to create 1 Assembly Order for the specified Product Variant. The Product Variant must exist for the specified Assembly Item. The specified Planned Offline Date is only used for a sell-multiple Product Variant. The Assembly Order Series is optional. If not specified, the standard logic for determination of order series is used. This PI is not suitable for sell-multiple Product Variants. PI 'AssemblyOrder.CreateV2' can be used for such variants. Transaction management is handled within the Public Interface. Errors which occurred during the actual assembly order creation process are logged in the Assembly Messages table (session tiasc050lm000), but also included in oExceptionID. oExceptionMessage and oExceptionID will also indicate if messages are logged. *** Warning *** This public interface is deprecated. use: AssemblyOrder.CreateV2

	Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process; that is handled within the function. Input: iAssemblyItem - configurable assembly item mandatory; must exist in session tiapl2500m000 iProductVariant - product variant mandatory; must have Assembly Item iPlannedOfflineDate - planned offline date mandatory for sell-multiple variants iReferenceDateType - assembly reference date type mandatory; it must have one of the values as can be specified in session tiapl3201m000 iAssemblyOrderSeries - assembly order series not mandatory; but must be valid if filled: must exist in session tcmcs0150m000 as series for the Assembly Order Number Group parameter Output: oAssemblyOrder - order number of the created assembly order oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - assembly order has been generated <> 0 - order could not be generated Use session Assembly Messages (tiasc0501m000) to view the messages.

AssemblyOrder.CreateV2

DLL:	tiextascapi
Syntax:	<pre> long AssemblyOrder.CreateV2 (domain tcitem iAssemblyItem, domain tccpva iProductVariant, domain tiutcd iPlannedOfflineDate, domain tiasc.ardt.typ iReferenceDateType, domain tcseri iAssemblyOrderSeries, domain tcorno iSalesOrder, domain tcpono iSalesOrderLine, ref domain tcorno oAssemblyOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: Use this Public interface to create 1 Assembly Order for the specified Product Variant. The Product Variant must exist for the specified Assembly Item.

	The specified Planned Offline Date is only used for a sell-multiple Product Variant. The Assembly Order Series is optional. If not specified, the standard logic for determination of order series is used. For sell-multiple variants, the Sales Order Number and Sales Order Position are necessary to retrieve the correct specification for the assembly order that must be created. Transaction management is handled within the Public Interface. Errors which occurred during the actual assembly order creation process are logged in the Assembly Messages table (session tiasc0501m000), but also included in oExceptionID. oExceptionMessage and oExceptionID will also indicate if messages are logged. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process; that is handled within the function. Input: iAssemblyItem - configurable assembly item mandatory; must exist in session tiapl2500m000 iProductVariant - product variant mandatory; must have Assembly Item iPlannedOfflineDate - planned offline date mandatory for sell-multiple variants iReferenceDateType - assembly reference date type mandatory; it must have one of the values as can be specified in session tiapl3201m000 iAssemblyOrderSeries - assembly order series not mandatory; but must be valid if filled: must exist in session tcmcs0150m000 as series for the Assembly Order Number Group parameter iSalesOrder - sales order number mandatory for sell-multiple variants iSalesOrderLine - sales order position mandatory for sell-multiple variants Output: oAssemblyOrder - order number of the created assembly order oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - assembly order has been generated <> 0 - order could not be generated Use session Assembly Messages (tiasc0501m000) to view the messages.

Public Interfaces for AssemblyLineStationOrder

The following functions are available:

[AssemblyLineStationOrder.AllocateParts](#)

[AssemblyLineStationOrder.ReportFinished](#)

[AssemblyLineStationOrder.ReportFinishedV2](#)

AssemblyLineStationOrder.AllocateParts

DLL:	tiextascapi
Syntax:	<pre> long AssemblyLineStationOrder.AllocateParts(domain tcsite iSite, domain tcorno iAssemblyOrder, domain tccwoc iLineStation, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this public interface to allocate parts (= build or rebuild the Assembly Part Requirements) for the specified Assembly Order and Line Station. Errors that occur during the actual allocation of the parts are logged in the Assembly Messages table (session tiasc0501m000), but also included in oExceptionID. oExceptionMessage and oExceptionID will also indicate if messages are logged. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iSite Not used iAssemblyOrder Assembly Order (mandatory). iLineStation Line Station (mandatory). Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception handling to get all relevant information. Return: 0 Allocate Parts was completed without any messages in the Assembly Messages table. <> 0 Allocate Parts could not be completed, or was completed with messages logged in the Assembly Messages table. Use session Assembly Messages

	(tiasc0501m000) to view the messages.

AssemblyLineStationOrder.ReportFinished

DLL:	tiextascapi												
Syntax:	<pre> long AssemblyLineStationOrder.ReportFinished(domain tcsite iSite, domain tcorno iAssemblyOrder, domain tccwoc iLineStation, domain tiutcs iActualCompletionTime, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>												
Usage:	Expl: Use this public interface to report a Line Station Order as Finished, as in session tiasl6510m000. Transaction management (retry point, commit/rollback) is handled within in this function, because also the created inbound warehouse orders are processed in this function. Inbound warehouse orders are only created when the last line station of the order is finished. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process; that is handled within the function. Input: <table> <tr> <td>iSite</td><td>Not used</td></tr> <tr> <td>iAssemblyOrder</td><td>Assembly Order (mandatory).</td></tr> <tr> <td>iLineStation</td><td>Line Station (mandatory).</td></tr> <tr> <td>iActualCompletionTime</td><td>Actual Completion Time (mandatory).</td></tr> </table> Output: <table> <tr> <td>oExceptionMessage</td><td>The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.</td></tr> <tr> <td>oExceptionID</td><td>An ID that refers to all error information. Use the functions in Exception handling to get all relevant information.</td></tr> </table> Return: 0 Report as Finished is successfully completed <> 0 Report as Finished is not completed	iSite	Not used	iAssemblyOrder	Assembly Order (mandatory).	iLineStation	Line Station (mandatory).	iActualCompletionTime	Actual Completion Time (mandatory).	oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.	oExceptionID	An ID that refers to all error information. Use the functions in Exception handling to get all relevant information.
iSite	Not used												
iAssemblyOrder	Assembly Order (mandatory).												
iLineStation	Line Station (mandatory).												
iActualCompletionTime	Actual Completion Time (mandatory).												
oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.												
oExceptionID	An ID that refers to all error information. Use the functions in Exception handling to get all relevant information.												

AssemblyLineStationOrder.ReportFinishedV2

DLL:	tiextascapi
Syntax:	long AssemblyLineStationOrder.ReportFinishedV2 (

	<pre> domain tcsite iSite, domain tcorno iAssemblyOrder, domain tccwoc iLineStation, domain tiutcs iActualStartTime, domain tiutcs iActualCompletionTime, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>														
Usage:	Expl: Use this public interface to report a Line Station Order as Finished, as in session tiasl6510m000. With ...V2() extra argument iActualStartTime is added, to allow reporting an exact date/time for this via this PI. (note: this is not possible via LN sessions). Transaction management (retry point, commit/rollback) is handled within in this function, because also the created inbound warehouse orders are processed in this function. Inbound warehouse orders are only created when the last line station of the order is finished. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process; that is handled within the function. Input: <table> <tr><td>iSite</td><td>Not used</td></tr> <tr><td>iAssemblyOrder</td><td>Assembly Order (mandatory).</td></tr> <tr><td>iLineStation</td><td>Line Station (mandatory).</td></tr> <tr><td>iActualStartTime</td><td>Actual Start Time (not mandatory).</td></tr> <tr><td>iActualCompletionTime</td><td>Actual Completion Time (mandatory).</td></tr> </table> Output: <table> <tr><td>oExceptionMessage</td><td>The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.</td></tr> <tr><td>oExceptionID</td><td>An ID that refers to all error information. Use the functions in Exception handling to get all relevant information.</td></tr> </table> Return: 0 Report as Finished is successfully completed <> 0 Report as Finished is not completed	iSite	Not used	iAssemblyOrder	Assembly Order (mandatory).	iLineStation	Line Station (mandatory).	iActualStartTime	Actual Start Time (not mandatory).	iActualCompletionTime	Actual Completion Time (mandatory).	oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.	oExceptionID	An ID that refers to all error information. Use the functions in Exception handling to get all relevant information.
iSite	Not used														
iAssemblyOrder	Assembly Order (mandatory).														
iLineStation	Line Station (mandatory).														
iActualStartTime	Actual Start Time (not mandatory).														
iActualCompletionTime	Actual Completion Time (mandatory).														
oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.														
oExceptionID	An ID that refers to all error information. Use the functions in Exception handling to get all relevant information.														

Public Interfaces for AssemblyPart

The following functions are available:

[AssemblyPart.StartProcessReturn](#)

AssemblyPart.StartProcessReturn

DLL:	tiextascapi This function is available from KB2259064 .
Syntax:	<pre> long AssemblyPart.StartProcessReturn(long iStartMode, domain tccwoc iLineStationFrom, domain tccwoc iLineStationTo, domain tiutcd iEffectiveDateFrom, domain tiutcd iEffectiveDateTo, domain tcitem iAssemblyPartFrom, domain tcitem iAssemblyPartTo, domain tcyesno iDirectProcessInbound, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the session Return Assembly Parts (tiasc7245m000). Pre: - Post: - Input: iStartMode (Mandatory) Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS_ALWAYS - Parent and child are parallel sessions that can be manipulated simultaneously. iLineStationFrom (Optional) Lower bound of the Line Station selection range. iLineStationTo (Optional) Upper bound of the Line Station selection range. If empty, the maximum value is used. iEffectiveDateFrom (Optional) Lower bound of the Effective Date selection range. iEffectiveDateTo (Optional) Lower bound of the Effective Date selection range. If empty, the current date is used. iAssemblyPartFrom (Optional) Lower bound of the Assembly Part selection range. iAssemblyPartTo (Optional) Lower bound of the Assembly Part selection range. If empty, the maximum value is used. iDirectProcessInbound If yes, the quantity that is completed is posted to inventory. The inbound procedure is processed automatically. Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - If the Return Assembly Parts session was started

	succesfully. <> 0 - Otherwise.

Public Interfaces for AssemblyProductVariant

The following functions are available:

[AssemblyProductVariant.CalculateStandardCost](#)

AssemblyProductVariant.CalculateStandardCost

DLL:	tiextascapi
	This function is available from KB2253167 .
Syntax:	<pre> long AssemblyProductVariant.CalculateStandardCost (domain tccpva iProductVariant, domain tccpcc iCalculationCode, domain tiutcs iEffectiveDate, domain tcyesno iActualizeCostForProductVariant, domain tcyesno iUpdateNotStartedFrozenOrders, domain tcorno iAssemblyOrderToUpdateFrom, domain tcorno iAssemblyOrderToUpdateTo, domain tiutcs iAssemblyOrderStartDateFrom, domain tiutcs iAssemblyOrderStartDateTo, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this Public Interface to calculate the Standard Cost for a specific Product Variant in Assembly. Calculation will be performed after updating the product variant structures. The result of the calculation may be used to update a specific set of Assembly Orders based on the new calculations. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process; that is handled within the function. Input: - iProductVariant - (Mandatory) Product Variant - iCalculationCode - (Mandatory) Calculation Code - iEffectiveDate - (Mandatory) Effective Date of Calculated Standard Cost. - iActualizeCostForProductVariant - If set to YES, the Current Standard Cost field in the

	Product Variant (Assembly) is updated with the total current value of the roll-off line taken from the Product Variant Standard Costs by Calculation Code (tiapl3540m000) session. If set to NO, only the new effective date and new cost component details are updated in the Product Variant Standard Costs by Calculation Code (tiapl3540m000). iUpdateNotStartedFrozenOrders - If set to YES, all assembly orders on which production has not yet started are updated. On assembly orders with WIP estimates already available and the status Sequenced, the WIP estimates are updated with the new product variant standard cost. iAssemblyOrderToUpdateFrom - Applies when iUpdateNotStartedFrozenOrders is YES. Defines the FROM assembly order in the range of assembly orders for which cost is actualized and not started frozen orders are updated. iAssemblyOrderToUpdateTo - Applies when iUpdateNotStartedFrozenOrders is YES. Defines the TO assembly order in the range of assembly orders for which cost is actualized and not started frozen orders are updated. iAssemblyOrderStartDateFrom - Applies when iUpdateNotStartedFrozenOrders is YES. Defines the FROM START date to select planned assembly orders for which cost is actualized and not started frozen orders are updated. iAssemblyOrderStartDateTo - Applies when iUpdateNotStartedFrozenOrders is YES. Defines the TO start date to select planned assembly orders for which cost is actualized and not started frozen orders are updated. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Calculation was successful. <> 0 - Calculation was not successful.

Chapter 23 Public Interfaces for Manufacturing Tools

Public Interfaces for Tool

The following functions are available:

[Tool.Request](#)

[Tool.Return](#)

[Tool.ReturnToOrder](#)

[Tools.GetSettings](#)

Tool.Request

DLL:	tiexttrpapi										
Syntax:	<pre> long Tool.Request (domain titrp.otyp iOrderType, domain tcsite iOrderSite, domain tcpdno iOrderNumber, domain tcopno iOperation, domain tcsern iOperationStep, domain tcponl iLineNumber, domain tcitem iTool, domain tcibd.sern iToolNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>										
Usage:	Expl: This Public Interface requests a Tool like session titrp0215m000. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: <table> <tr> <td>iOrderType</td><td>- Order Type</td></tr> <tr> <td>iOrderSite</td><td>- Order Site (Mandatory when Active)</td></tr> <tr> <td>iOrderNumber</td><td>- Order Number (Mandatory)</td></tr> <tr> <td>iOperation</td><td>- Operation (Mandatory if iOrderType is Production Order or Service Order)</td></tr> <tr> <td>iOperationStep</td><td>- Operation Step (Optional. Only used</td></tr> </table>	iOrderType	- Order Type	iOrderSite	- Order Site (Mandatory when Active)	iOrderNumber	- Order Number (Mandatory)	iOperation	- Operation (Mandatory if iOrderType is Production Order or Service Order)	iOperationStep	- Operation Step (Optional. Only used
iOrderType	- Order Type										
iOrderSite	- Order Site (Mandatory when Active)										
iOrderNumber	- Order Number (Mandatory)										
iOperation	- Operation (Mandatory if iOrderType is Production Order or Service Order)										
iOperationStep	- Operation Step (Optional. Only used										

	if iOrderType is Production Order) iLineNumber - Line Number (Mandatory if iOrderType is Service Order, Maintenance Work Order or Production Schedule) iTool - Tool (Mandatory) iToolNumber - Tool Number (Mandatory) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Tool requested. <> 0 - Otherwise.

Tool.Return

DLL:	tiexttrpapi
Syntax:	<pre>long Tool.Return(domain titrp.otyp iOrderType, domain tcsite iOrderSite, domain tcpdno iOrderNumber, domain tcopno iOperation, domain tcsern iOperationStep, domain tcponl iLineNumber, domain tcitem iTool, domain tcibd.sern iToolNumber, domain tccwoc iWorkCenter, domain tcyesno iReturnToWarehouse, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This Public Interface returns a Tool like session titrp0215m000. If iWorkCenter is not empty, the Tool will be returned to the Work Center. If iWarehouse is 'Yes', the Tool will be returned to the warehouse. If iWorkcenter is empty and iWarehouse is on 'No', the Tool is planned for the next request line. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: iOrderType - Order Type iOrderSite - Order Site (Mandatory when Active) iOrderNumber - Order Number (Mandatory) iOperation - Operation (Mandatory if iOrderType is Production Order or Service Order) iOperationStep - Operation Step (Optional. Only used if iOrderType is Production Order) iLineNumber - Line Number (Mandatory if iOrderType is Service Order, Maintenance Work Order or Production Schedule) iTool - Tool (Mandatory) iToolNumber - Tool Number (Mandatory)

	iWorkCenter iReturnToWarehouse Output: oExceptionMessage oExceptionID Return: 0 <> 0	- Work Center to return the Tool to (Optional, should be empty if iWarehouse is 'Yes') - If the Tool should be returned to the Warehouse (Should be 'No' if iWorkCenter is not empty) - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. - Tool returned. - Otherwise.

Tool.ReturnToOrder

DLL:	tiexttrpapi																		
Syntax:	<pre>long Tool.ReturnToOrder(domain titrp.otyp iOrderType, domain tcsite iOrderSite, domain tcpdno iOrderNumber, domain tcopno iOperation, domain tcsern iOperationStep, domain tcponl iLineNumber, domain tcitem iTool, domain tcibd.sern iToolNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>																		
Usage:	Expl: This Public Interface returns a Tool to Order like session titrp0215m000. Pre: Retry point must be set. Post: Commit or abort the transaction. Input: <table><tr><td>iOrderType</td><td>- Order Type</td></tr><tr><td>iOrderSite</td><td>- Order Site (Mandatory when Active)</td></tr><tr><td>iOrderNumber</td><td>- Order Number (Mandatory)</td></tr><tr><td>iOperation</td><td>- Operation (Mandatory)</td></tr><tr><td>iOperationStep</td><td>- Operation Step (Mandatory)</td></tr><tr><td>iLineNumber</td><td>- Line Number (Mandatory)</td></tr><tr><td>iTool</td><td>- Tool (Mandatory)</td></tr><tr><td>iToolNumber</td><td>- Tool Number (Mandatory)</td></tr></table> Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant			iOrderType	- Order Type	iOrderSite	- Order Site (Mandatory when Active)	iOrderNumber	- Order Number (Mandatory)	iOperation	- Operation (Mandatory)	iOperationStep	- Operation Step (Mandatory)	iLineNumber	- Line Number (Mandatory)	iTool	- Tool (Mandatory)	iToolNumber	- Tool Number (Mandatory)
iOrderType	- Order Type																		
iOrderSite	- Order Site (Mandatory when Active)																		
iOrderNumber	- Order Number (Mandatory)																		
iOperation	- Operation (Mandatory)																		
iOperationStep	- Operation Step (Mandatory)																		
iLineNumber	- Line Number (Mandatory)																		
iTool	- Tool (Mandatory)																		
iToolNumber	- Tool Number (Mandatory)																		

	Return: 0 <> 0	information. - Tool returned to Order. - Otherwise.

Tools.GetSettings

DLL:	tiexttrpapi	
Syntax:	<pre>long Tools.GetSettings(domain tcncmp iLogisticCompany, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...)</pre>	
Usage:	Expl: This function retrieves master data settings for tool requirement planning (table titrp000 / titrp080). One or more setting fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table titrp000), or site level (settings table titrp080), depending on: - implementation phase of Sites, - input argument iSite, - the specified input field(s) (mnemonic). Note: This function uses variable arguments, for input and output. Data is retrieved via name-value pairs: specify the field (input) and a variable for the field value (output). Example: when the settings 'default refurbish by' (drfb) and 'tool rate time unit' (untr) are needed, the function must be called with the following arguments: <pre>Tools.GetSettings(* Fixed arguments: company, --> input site, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output * Variable arguments: "drfb", --> input default.refurb.by, --> output "untr", --> input tool.rate.time.unit) --> output</pre> Pre: None Post: None Input: iLogisticCompany - Logistic Company: Mandatory	

	iSite - Site: Not Mandatory iForceRead - Option to force reading new data in stead of using cached information iContextIsMasterData - True/False (not used) ... - The field mnemonic of the required field(s) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field(s). Return: 0 - Data read <> 0 - Otherwise.

Chapter 24 Public Interfaces for Manufacturing Project

Public Interfaces for ProjectPCS

The following functions are available:

[ProjectPCS.CalculateCost](#)

[ProjectPCS.CopyCustomizedProductStructureToCustomizedStructure](#)

[ProjectPCS.CopyCustomizedProductStructureToStandardStructure](#)

[ProjectPCS.CopyProject](#)

[ProjectPCS.CopyStandardProductStructureToCustomizedStructure](#)

[ProjectPCS.Start360](#)

ProjectPCS.CalculateCost

DLL:	tiextpcsapi
Syntax:	<pre> long ProjectPCS.CalculateCost(domain tccprj iProject, boolean iCalculateEstimatedCost, boolean iCalculateActualCost, domain tcdate iEffectiveDate, boolean iNetChangeOnly, boolean iSimulatedCalculation, domain tccpcc iSimulationCalculationCode, boolean iUpdateCOSDistribution, boolean iRecalculateStandardParts, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will Calculate the PCS Project Cost for one project, according to the logic of session Calculate Standard Costs by Project (tipcs3250m000). Note that even when the function returns 0 (indicating success), in various parts of the calculation process warning messages may have been set.

	These messages can be found in the returned ExceptionID structure. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within. Input: iProject The PCS project (Mandatory). iCalculateEstimatedCost When true: Calculate Estimated Cost. iCalculateActualCost When true: Calculate Actual Cost. iEffectiveDate Use this date as the Effective Date of Standard Cost and Valuation Prices. When set to 0, the current date is used. iNetChangeOnly When true: Only calculate actual cost when transactions were logged after the previous actual cost calculation. iSimulatedCalculation When true: perform simulated calculation of Standard Cost of all project items. Simulation cannot be combined with Estimated or Actual Cost calculation. iSimulationCalculationCode (Optional), if provided, the calculation is done using the given code. Otherwise the Calculation Code of the Project is used. This affects the Simulated Cost calculation. iUpdateCOSDistribution When true: the COS (Cost of Sales) distribution is updated. This option is independent of other calculation options. iRecalculateStandardParts When true: Standard Parts in the customized bill of material are recalculated. This option affects the Estimated Cost calculation. Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error/warning information. Use the functions in Exception handling to get all relevant information. Return: 0 Successful completion <> 0 Failure

ProjectPCS.CopyCustomizedProductStructureToCustomizedStructure

DLL:	tiextpcsapi
	This function is available from KB3555140 .
Syntax:	<pre>long ProjectPCS.CopyCustomizedProductStructureToCustomizedStructure(domain tcsite iSite, domain tcitem iSourceCustomizedItem, domain tcitem iTargetCustomizedItem,</pre>

	domain tyesno iUseProjectReferenceDate, domain tcdater iReferenceDate, domain tyesno iCopyAllComponentsAndEffectiveUnits, domain tceff.effn iUnit, domain ticpst iCopyMethod, domain tyesno iCopyEitemRelationships, domain tyesno iCopyProductVariantStructure, domain tyesno iIncludeAlternatives, domain tyesno iIncludeUseUpMaterials, domain tyesno iGenerateNewItemCodes, domain tyesno iApproveConversionFactors, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: Copy a customized product structure to a customized structure linked to a PCS project. The project segment of given Customized Item determines which PCS Project the customized structure is created for. Pre: Retry point must be set. Post: Abort or commit the transaction. Input: iSite Site (Mandatory when the Site concept is active). iSourceCustomizedItem Source Customized Item to copy (Mandatory). The project segment determines the Project. iTargetCustomizedItem Target Customized Item to copy to (Mandatory). The project segment determines the Project. iUseProjectReferenceDate Indicates if the reference date of the target PCS project must be used (Mandatory). If the project does not have a reference date, the current date is used. iReferenceDate Reference date to determine which components must be included. Not applicable when iUseProjectReferenceDate is set to YES. iCopyAllComponentsAndEffectiveUnits To specify whether or not effectivity unit in the product structure must be copied. Mandatory if Unit Effectivity is YES in common parameters. iUnit Effectivity Unit (Optional). When provided, only the effective components are copied. Not applicable when iCopyAllComponentsAndEffectiveUnits is set to YES. iCopyMethod Copy method (Mandatory). Possible options are: Single Level: Only the first level of the product structure is copied Multilevel: All levels of the product structure are copied. Interactive is not applicable. iCopyEitemRelationships To specify whether the E-item relationships are copied to the customized structure. Mandatory when Site is not active. iCopyProductVariantStructure To specify whether the product variant structure of the customized source item is copied to customized target item. (Mandatory). iIncludeAlternatives To specify whether or not alternative items in the product structure must be

	copied. Mandatory if Alternative materials is YES in common parameters. iIncludeUseUpMaterials To specify whether or not use up materials in the product structure must be copied. Mandatory if Alternative materials is YES in common parameters. iGenerateNewItemCodes To specify whether or not new item codes must be generated when the customized item already exist. iApproveConversionFactors To specify whether or not for the created customized items, conversion factors need to be approved automatically (Mandatory). Output: oExceptionMessage The last message if any message is found.If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Successfully copied. <> 0 Failed to copy.

ProjectPCS.CopyCustomizedProductStructureToStandard Structure

DLL:	tiextpcsapi
	This function is available from KB3555140 .
Syntax:	<pre>long ProjectPCS.CopyCustomizedProductStructureToStandardStructure(domain tcsite iSite, domain tcitem iStandardItem, domain tcitem iCustomizedItem, domain tcyesno iCopyAllComponentsAndEffectiveUnits, domain tcuef.effn iUnit, domain tcdate iReferenceDate, domain ticpst iCopyMethod, domain tcyesno iCopyEitemRelationships, domain tcyesno iIncludeAlternatives, domain tcyesno iIncludeUseUpMaterials, domain tcyesno iGenerateNewItemCodes, domain tcyesno iApproveConversionFactors, domain tcyesno iCopyToStandardItemConfiguration, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Copy a customized product structure to a standard structure. Pre: Retry point must be set. Post: Abort or commit the transaction.

	Input: iSite Site (Mandatory when the Site concept is active). iStandardItem Standard Item to copy (Mandatory). iCustomizedItem Customized Item to copy to (Mandatory). The project segment determines the Project. iCopyAllComponentsAndEffectiveUnits To specify whether or not effectivity unit in the product structure must be copied. Mandatory if Unit Effectivity is YES in common parameters. iUnit Effectivity Unit (Optional). When provided, only the effective components are copied. Not applicable when iCopyAllComponentsAndEffectiveUnits is set to YES. iReferenceDate Reference date to determine which components must be included. (Mandatory). iCopyMethod Copy method (Mandatory). Possible options are: Single Level: Only the first level of the product structure is copied If standard item and Customized item are equal then Single level is not applicable Multilevel: All levels of the product structure are copied. Interactive is not applicable. iCopyEitemRelationships To specify whether the E-item relationships of a customized structure is copied to the standard structure. Mandatory when Site is not active. iIncludeAlternatives To specify whether or not alternative items in the product structure must be copied. Mandatory if Alternative materials is YES in common parameters. iIncludeUseUpMaterials To specify whether or not use up materials in the product structure must be copied. Mandatory if Alternative materials is YES in common parameters. iGenerateNewItemCodes To specify whether or not new item codes must be generated when the target customized item already exist. iApproveConversionFactors To specify whether or not for the created customized items, conversion factors need to be approved automatically (Mandatory). iCopyToStandardItemConfiguration To specify whether product structure of a customized structure is copied to standard structure. Output: oExceptionMessage The last message if any message is found.If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant

	information. Return: 0 Successfully copied. <> 0 Failed to copy.

ProjectPCS.CopyProject

DLL:	tiextpcsapi																								
Syntax:	<pre> long ProjectPCS.CopyProject(domain tccprj iSourceProject, domain tccprj iTargetProjectCode, domain tckopr iTargetProjectType, domain tipcs.ccgr iTargetCalculationGroup, domain tcsite iSourceSiteItemRelatedData, boolean iCopyItemData, boolean iCopyBillsOfMaterial, boolean iCopyRoutings, boolean iCopyCostingData, boolean iCopyPlanning, boolean iCopyProductVariantStructure, domain tcynna iApproveConversionFactors, ref domain tccprj oGeneratedProject, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>																								
Usage:	Expl: This Public Interface will copy a PCS Project. If iTargetProjectCode is empty, the created Project Code will be generated using the First Free Number of the default Project Series. If iTargetProjectCode is only filled with a Project Series, then the created Project Code will be generated using the First Free Number of the given Project Series. Pre: Retry point has been set. Post: Abort or commit the transaction. Input: <table> <tr> <td>iSourceProject</td><td>Source Project (Mandatory).</td></tr> <tr> <td>iTargetProjectCode</td><td>Target Project Code (Optional).</td></tr> <tr> <td>iTargetProjectType</td><td>Target Project Type (Mandatory).</td></tr> <tr> <td>iTargetCalculationGroup</td><td>Target Calculation Group (Optional).</td></tr> <tr> <td>iSourceSiteItemRelatedData</td><td>The source Site for the BOM, Routing, Subcontracting Model to be copied with the Project (Optional).</td></tr> <tr> <td>iCopyItemData</td><td>Copy Item Data.</td></tr> <tr> <td>iCopyBillsOfMaterial</td><td>Copy Bill of Material and Subcontracting models.</td></tr> <tr> <td>iCopyRoutings</td><td>Copy Routings.</td></tr> <tr> <td>iCopyCostingData</td><td>Copy Costing Data.</td></tr> <tr> <td>iCopyPlanning</td><td>Copy Planning.</td></tr> <tr> <td>iCopyProductVariantStructure</td><td>Copy Product Variant Structure.</td></tr> <tr> <td>iApproveConversionFactors</td><td>Approve the Conversion Factors of the copied Project.</td></tr> </table>	iSourceProject	Source Project (Mandatory).	iTargetProjectCode	Target Project Code (Optional).	iTargetProjectType	Target Project Type (Mandatory).	iTargetCalculationGroup	Target Calculation Group (Optional).	iSourceSiteItemRelatedData	The source Site for the BOM, Routing, Subcontracting Model to be copied with the Project (Optional).	iCopyItemData	Copy Item Data.	iCopyBillsOfMaterial	Copy Bill of Material and Subcontracting models.	iCopyRoutings	Copy Routings.	iCopyCostingData	Copy Costing Data.	iCopyPlanning	Copy Planning.	iCopyProductVariantStructure	Copy Product Variant Structure.	iApproveConversionFactors	Approve the Conversion Factors of the copied Project.
iSourceProject	Source Project (Mandatory).																								
iTargetProjectCode	Target Project Code (Optional).																								
iTargetProjectType	Target Project Type (Mandatory).																								
iTargetCalculationGroup	Target Calculation Group (Optional).																								
iSourceSiteItemRelatedData	The source Site for the BOM, Routing, Subcontracting Model to be copied with the Project (Optional).																								
iCopyItemData	Copy Item Data.																								
iCopyBillsOfMaterial	Copy Bill of Material and Subcontracting models.																								
iCopyRoutings	Copy Routings.																								
iCopyCostingData	Copy Costing Data.																								
iCopyPlanning	Copy Planning.																								
iCopyProductVariantStructure	Copy Product Variant Structure.																								
iApproveConversionFactors	Approve the Conversion Factors of the copied Project.																								

	Output: oGeneratedProject The created Project code. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error/warning information. Return: 0 Use the functions in Exception handling to get all relevant information. <> 0 Project Copied succesfully. Project has not been copied.

ProjectPCS.CopyStandardProductStructureToCustomized Structure

DLL:	tiextpcsapi
	This function is available from KB3555140 .
Syntax:	<pre>long ProjectPCS.CopyStandardProductStructureToCustomizedStructure(domain tcsite iSite, domain tcitem iStandardItem, domain tcitem iCustomizedItem, domain tcyesno iUseProjectReferenceDate, domain tcddate iReferenceDate, domain tcyesno iCopyAllComponentsAndEffectiveUnits, domain tcuef.effn iUnit, domain ticpst iCopyMethod, domain ticmpm iCopyModels, domain tcyesno iApproveModels, domain ticcit iCopyMethodItems, domain tcyesno iCopyEitemRelationships, domain tcyesno iIncludeAlternatives, domain tcyesno iIncludeUseUpMaterials, domain tcyesno iGenerateNewItemCodes, domain tcyesno iApproveConversionFactors, domain tcyesno iCopyTestingCombinations, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Copy a standard product structure to a customized structure linked to a PCS project. The project segment of given Customized Item determines which PCS Project the customized product structure is created for. Pre: Retry point must be set. Post: Abort or commit the transaction. Input: iSite Site (Mandatory when the Site concept is active). iStandardItem Standard Item to copy (Mandatory). iCustomizedItem Customized Item to copy to (Mandatory). The project segment determines the PCS Project. iUseProjectReferenceDate Indicates if the reference date of the target

	PCS project must be used (Mandatory). If the project does not have a reference date, the current date is used. iReferenceDate Date to determine which Job Shop BOMs and Job Shop Routings must be included Not applicable when iUseProjectReferenceDate is set to YES. iCopyAllComponentsAndEffectiveUnits To specify whether or not effectivity units in the product structure must be copied. Mandatory if Unit Effectivity is YES in common parameters. iUnit Effectivity Unit (Optional). When provided, only the effective components are copied. Not applicable when iCopyAllComponentsAndEffectiveUnits is set to YES. iCopyMethod Copy method (Mandatory). Possible options are: Single Level: Only the first level of the product structure is copied Multilevel: All levels of the product structure are copied. Interactive is not applicable. iCopyModels Specify scope of copying production models (Mandatory) and applicable only if sites is implemented. Possible copy options are: Only Use for Planning models: Models marked as Use for Planning All Models : All models. iApproveModels To specify whether or not copied models must be approved (Mandatory) and applicable only if sites is implemented. iCopyMethodItems To specify which class of items must be included (Mandatory). Possible values are Customizable Items: Only customizable items in the product structure are copied. All: All items are copied. iCopyEitemRelationships To specify whether the E-item relationships are copied to the customized structure. Mandatory when Site is not active. iIncludeAlternatives To specify whether or not alternative items in the product structure must be copied. Mandatory if Alternative materials is YES in common parameters. iIncludeUseUpMaterials To specify whether or not use up materials in the product structure must be copied. Mandatory if Alternative materials is YES in common parameters. iGenerateNewItemCodes To specify whether or not new item codes must be generated when the customized item already exist.

	iApproveConversionFactors To specify whether or not for the created customized items, conversion factors need to be approved automatically (Mandatory). iCopyTestingCombinations To specify whether or not combinations used for testing (in Quality Management) need to be copied along with the item. Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Successfully copied. <> 0 Failed to copy.

ProjectPCS.Start360

DLL:	tiextpcsapi
Syntax:	<pre>long ProjectPCS.Start360(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccprj iProject, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the session Project 360 (tipcs0320m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Optional. Possible values are: "ShowAllProjects": All Projects are displayed. "ShowBudgets": Only Budgets are shown. "ShowOpenProjects": Only open Projects are shown. "ShowActiveProjects": Only active Projects are shown. "ShowSimulatedProjects": Only simulated Projects are shown.

	"ShowFreeProjects": Only free Projects are shown. "ShowCanceledProjects": Only canceled Projects are shown. "ShowFinishedProjects": Only finished Projects are shown. "ShowToBeClosedProjects": Only to be closed Projects are shown. "ShowClosedProjects": Only closed Projects are shown. "ShowArchivedProjects": Only archived Projects are shown. "ShowOneProject": Only iProject is shown. iSessionIndex The index that will be used. Supported values: 1: sort by Project. 2: sort by Search Key/Project. 3: sort by Kind Of Project/Project. 4: sort by Calculation Group/Project. 5: sort by Sold-to BP/Project. iQueryExtend Not used. iProject Project. Only used if iStartFilter "ShowOneProject" is used. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started. <> 0 Otherwise.

Chapter 25 Public Interfaces for Warehousing

Public Interfaces for AdjustmentOrder

The following functions are available:

[AdjustmentOrder.Process](#)

[AdjustmentOrder.StartDetail](#)

[AdjustmentOrder.StartMultiMain](#)

[AdjustmentOrder.StartOverview](#)

AdjustmentOrder.Process

DLL:	whextinhapi
Syntax:	<pre>long AdjustmentOrder.Process(domain tcorno iAdjustmentOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This Public Interface will process a given adjustment order. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iAdjustmentOrder - The Adjustment Order which needs to be processed; Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Adjustment Order has been processed successfully <> 0 - Error.

AdjustmentOrder.StartDetail

DLL:	whextinhapi
Syntax:	<pre> long AdjustmentOrder.StartDetail(long iStartMode, domain tcorno iOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the detail session Adjustment Orders (whinh5100m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iOrder Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

AdjustmentOrder.StartMultiMain

DLL:	whextinhapi
Syntax:	<pre> long AdjustmentOrder.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iOrder, domain tccwar iWarehouse, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the multi main table session Adjustment Order (whinh5600m000). Input: iStartMode

	Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byWarehouse": adjustment orders are displayed by warehouse. session will be started on index 2 view fields: Warehouse iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iOrder iWarehouse Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

AdjustmentOrder.StartOverview

DLL:	whextinhapi
Syntax:	<pre>long AdjustmentOrder.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iOrder, domain tccwar iWarehouse, ref domain tcorno oOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl This function starts the overview session Adjustment Orders (whinh5100m000). Input: iStartMode Specifies the start mode for the session. Possible values are:

	MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byWarehouse": adjustment orders are displayed by warehouse. session will be started on index 2 view fields: Warehouse iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iOrder iWarehouse Mandatory when iStartFilter is "byWarehouse", or when iSessionIndex is 2 and start mode is MODELESS Output: for iStartMode MODAL: oOrder - order of selected adjustment oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

Public Interfaces for AdjustmentOrderLine

The following functions are available:

[AdjustmentOrderLine.Process](#)

[AdjustmentOrderLine.Submit](#)

AdjustmentOrderLine.Process

DLL:	whextinhapi
Syntax:	<pre> long AdjustmentOrderLine.Process(domain tcorno iAdjustmentOrder, domain tcpono iAdjustmentOrderLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface will process a given adjustment order line. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iAdjustmentOrder - The Adjustment Order which needs to be processed; Mandatory iAdjustmentOrderLine - The Adjustment Order Line which needs to be processed; Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Adjustment Order Line has been processed successfully <> 0 - Error.

AdjustmentOrderLine.Submit

DLL:	whextinhapi
Syntax:	<pre> long AdjustmentOrderLine.Submit(domain tcorno iAdjustmentOrder, domain tcpono iAdjustmentOrderLine, domain tcmcs.str30 iAction, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function will submit a draft version of the adjustment order line which is being controlled by Object Configuration Management. The workflow will be started for the adjustment order line with this function. Pre: db.retry.point() must have been set. This function will read the adjustment order line, so that is not required from the calling process. Post: Commit or abort the transaction Input: iAdjustmentOrder - Adjustment Order; Mandatory iAdjustmentOrderLine - Adjustment Order Line; Mandatory iAction - The action to be performed; Mandatory This action can be one of the default actions: Create, Change, Delete. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID

	<pre> oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Success <> 0 - Error </pre>

Public Interfaces for CycleCountingOrder

The following functions are available:

[CycleCountingOrder.Generate](#)

[CycleCountingOrder.Process](#)

[CycleCountingOrder.StartDetail](#)

[CycleCountingOrder.StartMultiMain](#)

[CycleCountingOrder.StartOverview](#)

CycleCountingOrder.Generate

DLL:	whextinhapi
Syntax:	<pre> long CycleCountingOrder.Generate(domain tcsite iSiteFrom, domain tcsite iSiteTo, domain tccwar iWarehouseFrom, domain tccwar iWarehouseTo, domain whloca iLocationFrom, domain whloca iLocationTo, domain tccom.bpid iLocationOwnerFrom, domain tccom.bpid iLocationOwnerTo, domain tcdate iInventoryDateFrom, domain tcdate iInventoryDateTo, domain whwmd.zone iZoneFrom, domain whwmd.zone iZoneTo, domain whwmd.strt iRowFrom, domain whwmd.strt iRowTo, domain whwmd.coln iLevelFrom, domain whwmd.coln iLevelTo, domain whwmd.rack iBinFrom, domain whwmd.rack iBinTo, domain tccitg iItemGroupFrom, domain tccitg iItemGroupTo, domain tcclot iLotFrom, </pre>

	domain tcclot iLotTo, domain whhuid iHandlingUnitFrom, domain whhuid iHandlingUnitTo, domain tcdate iLastCountingDateFrom, domain tcdate iLastCountingDateTo, domain tcabcc iABCCCodeFrom, domain tcabcc iABCCCodeTo, domain tcamnt iInventoryValueFrom, domain tcamnt iInventoryValueTo, domain tccur iCurrency, domain tcrtyp iRateType, domain tcitem iItemFrom, domain tcitem iItemTo, boolean iReconciliation, boolean iForceCycleCount, boolean iExcludeLocationAllocatedItems, boolean iCountBasedOnInventoryValue, boolean iCountNewItem, boolean iCountNonHandlingUnitInventory, boolean iCountHandlingUnitInventory, domain whinh.cchu iHandlingUnitLevel, boolean iOnlyCompleteHandlingUnitTopLevels, domain whinh.ccmihu iMultiItemHandlingUnits, domain tcseri iSeries, domain tcqcia iMaximumNumberOfLinesPerOrder, domain tcqcia iMaximumNumberOfStockPoints, ref long oNumberOfCreatedLines, ref domain tcorno oOrderArray() fixed, ref domain tcsern oCountNumberArray(), ref domain tcpono oOrderLineArray(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function generates cycle counting orders. Be aware that transaction management is handled within this function. Pre: oOrderArray, oCycleCountNumberArray and oOrderLineArray must be declared as based variables. This function will allocate the memory. Post: N.a. Input: iSiteFrom - Not Mandatory iSiteTo - Mandatory iWarehouseFrom - Not Mandatory iWarehouseTo - Mandatory iLocationFrom - Not Mandatory iLocationTo - Mandatory iLocationOwnerFrom - Not Mandatory iLocationOwnerTo - Mandatory iInventoryDateFrom - Not Mandatory iInventoryDateTo - Mandatory iZoneFrom - Not Mandatory iZoneTo - Mandatory iRowFrom - Not Mandatory iRowTo - Mandatory iLevelFrom - Not Mandatory iLevelTo - Mandatory iBinFrom - Not Mandatory iBinTo - Mandatory iItemGroupFrom - Not Mandatory iItemGroupTo - Mandatory iLotFrom - Not Mandatory iLotTo - Mandatory iHandlingUnitFrom - Not Mandatory

	iHandlingUnitTo - Mandatory iLastCountingDateFrom - Not Mandatory iLastCountingDateTo - Mandatory iAbcCodeFrom - Not Mandatory iAbcCodeTo - Mandatory iInventoryValueFrom - Not Mandatory iInventoryValueTo - Mandatory iCurrency - Not Mandatory iRateType - Not Mandatory iItemFrom - Not Mandatory iItemTo - Mandatory iReconciliation - When this is set to True, the discrepancies will be reconciled between ERP LN and the WMS system. iForceCycleCount - When this is set to True, all stock points found in the selection are added to a cycle count order. iExcludeLocationAllocatedItems - When this is set to True, ERP LN does not generate a cycle count order line for stock points with location allocated inventory. iCountBasedOnInventoryValue - When this is set to True, the InventoryValue range will be used and applied. iCountNewItem - When this is set to True, cycle counting order lines are generated for new items that were earlier never handled in the logistic process iCountNonHandlingUnitInventory - When this is set to True, inventory that is not stored in handling units is counted. iCountHandlingUnitInventory - When this is set to True, inventory that is stored in handling units is counted iHandlingUnitLevel - Mandatory Allowed Values: - Count Handling Unit Top Levels (whinh.cchu.top.level) - Count Handling Unit Bottom Levels (whinh.cchu.bottom.level) iOnlyCompleteHandlingUnitTopLevels - When this is set to True, cycle counting orders are generated for top-level handling units for which the Complete and Labeled check boxes are selected in the Handling Units session. When this is set to False, cycle counting orders are generated for top-level handling units regardless of the settings of the Complete and Labeled check boxes in the Handling Units session. iMultiItemHandlingUnits - Mandatory Allowed Values: - Include (whinh.ccmihu.include) - Exclude (whinh.ccmihu.exclude) - Only (whinh.ccmihu.only) iSeries - Not Mandatory iMaximumNumberOfLinesPerOrder - Mandatory iMaximumNumberOfStockPoints - Mandatory Output: oNumberOfCreatedLines - Number of Created Cycle Counting Order Lines oOrderArray - Array with created Orders oCycleCountNumberArray - Array with created Cycle Count Numbers oOrderLineArray - Array with created Cycle Counting Order Lines oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use

	the functions in Exception to get all relevant information. Return: 0 - Success <> 0 - Error

CycleCountingOrder.Process

DLL:	whextinhapi
Syntax:	<pre>long CycleCountingOrder.Process(domain tcorno iOrder, domain tcsern iCountNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This Public Interface will process a given cycle counting order. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iOrder - The Cycle Counting Order which needs to be processed; Mandatory iCountNumber - The Count Number of the Cycle Counting Order which needs to be processed; Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Cycle Counting Order has been processed successfully <> 0 - Error.

CycleCountingOrder.StartDetail

DLL:	whextinhapi
Syntax:	<pre>long CycleCountingOrder.StartDetail(long iStartMode, domain tcorno iOrder, domain tcsern iCountNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the detail session Cycle Counting Orders (whinh5100m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel

	sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iOrder iCountNumber Output: oExceptionMessage oExceptionID Return: 0 <> 0 - The iOrder and iCountNumber must refer to an existing cycle count order - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Session started - Error.

CycleCountingOrder.StartMultiMain

DLL:	whextinhapi
Syntax:	<pre>long CycleCountingOrder.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iOrder, domain tcsern iCountNumber, domain tccwar iWarehouse, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the multi main table session Cycle Counting Order (whinh5600m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byWarehouse": cycle count orders are displayed by warehouse. session will be started on index 2 view fields: Warehouse iSessionIndex

	Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iOrder iCountNumber iWarehouse Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

CycleCountingOrder.StartOverview

DLL:	whextinhapi
Syntax:	<pre> long CycleCountingOrder.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iOrder, domain tcsern iCountNumber, domain tccwar iWarehouse, ref domain tcorno oOrder, ref domain tcsern oCountNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl This function starts the overview session Cycle Counting Orders (whinh5100m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byWarehouse": cycle count orders are displayed by warehouse. session will be started on index 2 view fields: Warehouse

	iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iOrder iCountNumber iWarehouse Mandatory when iStartFilter is "byWarehouse", or when iSessionIndex is 2 and when the start mode is MODELESS Output: for iStartMode MODAL: oOrder - order of selected cycle count oCountNumber - count number of selected cycle count oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

Public Interfaces for ItemWarehouseFixedLocation

The following functions are available:

[ItemWarehouseFixedLocation.StartOverview](#)

ItemWarehouseFixedLocation.StartOverview

DLL:	whextwmdapi
Syntax:	<pre>long ItemWarehouseFixedLocation.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, domain tccwar iWarehouse, ref domain tcitem oItem, ref domain tccwar oWarehouse, ref domain whloca oLocation,</pre>

	<pre> ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl: This function starts the overview session Fixed Locations (whwmd3502m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byItemWarehouse": data is displayed by Item and Warehouse session will be started on index 1 view fields: Item, Warehouse "byWarehouseLocation": data is displayed by Warehouse session will be started on index 2 view field: Warehouse "byItemWarehousePriority": data is displayed by Item and Warehouse session will be started on index 3 view field: Item, Warehouse iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a iStartFilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iItem Mandatory when iStartFilter "byItemWarehouse" or "byItemWarehousePriority" is used or iSessionIndex is 1 or 3, and iStartMode is MODELESS. iWarehouse Mandatory when iStartFilter "byItemWarehouse" or "byItemWarehousePriority" or "byWarehouseLocation" is used or iSessionIndex 1 or 3 and iStartMode is MODELESS. Output: for iStartMode MODAL: oWarehouse - Warehouse of selected fixed location. oItem - Item of selected fixed location. oLocation - Location of selected fixed location. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error </pre>

Public Interfaces for ItemWarehouseInventoryTransaction

The following functions are available:

[ItemWarehouseInventoryTransaction.StartDetail](#)

[ItemWarehouseInventoryTransaction.StartOverview](#)

ItemWarehouseInventoryTransaction.StartDetail

DLL:	whextinrapi
Syntax:	<pre> long ItemWarehouseInventoryTransaction.StartDetail(long iStartMode, domain tcitem iItem, domain tccwar iWarehouse, domain tcddate iTransactionDate, domain tcmcs.long iSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session Item - Warehouse - Inventory Transactions (whinrl510s000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iItem iWarehouse iTransactionDate iSequence Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started

	<> 0	- Error.

ItemWarehouseInventoryTransaction.StartOverview

DLL:	whextinrapi
Syntax:	<pre> long ItemWarehouseInventoryTransaction.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, domain tccwar iWarehouse, domain tcdate iTransactionDate, domain tcmcs.long iSequence, domain tcncmp iOrderCompany, domain tckoor iTypeOfOrder, domain tcorno iOrder, domain tccprj iProject, domain tcpdm.cspa iElement, domain tcpdm.cact iActivity, domain tcorno iInventoryTransactionId, domain tcmcs.long iInventoryTransactionIdSequence, domain tccom.bpid iOwner, domain tccom.bpid iBuyFromBusinessPartner, ref domain tcitem oItem, ref domain tccwar oWarehouse, ref domain tcdate oTransactionDate, ref domain tcmcs.long oSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref domain long oExceptionID) </pre>
Usage:	Expl: This function starts session Item - Warehouse - Inventory Transactions (whinrl510m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byItemWarehouse": data is displayed by item and warehouse session will be started on index 1 view fields: Item and Warehouse "byItem": data is displayed by item session will be started on session index 1 view fields: Item "byOrder": data is displayed by Order Company, and

	Type of Order session will be started on session index 2 view fields: Order Company, Type of Order "byProjectOrder": data is displayed by Project, Element, Activity, Order Company and Type of Order session will be started on session index 3 view fields: Project, Element, Activity "byTransactionId" data is displayed by Inventory Transaction ID session will be started on session index 4 view fields: Inventory Transaction ID "byOwner" data is displayed by Owner session will be started on session index 5 view fields: Owner "byBuyFromBusinessPartner" data is displayed by Buy-From Business Partner session will be started on session index 6 view fields: Buy-From Business Partner iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iItem Mandatory when iStartFilter "byItemWarehouse" or "byItem" is used or iSessionIndex = 1 and iStartMode is MODELESS iWarehouse Mandatory when iStartFilter "byItemWarehouse" is used or iSessionIndex = 1 and iStartMode is MODELESS iTransactionDate Not mandatory. iSequence Not mandatory. iOrderCompany Mandatory when iStartFilter "byOrder" is used or iSessionIndex = 2 and iStartMode is MODELESS iTypeOfOrder Mandatory when iStartFilter "byOrder" is used or iSessionIndex = 2 is used and iStartMode is MODELESS iOrder Not mandatory. iProject Not mandatory. iElement Not Mandatory. iActivity Not Mandatory. iInventoryTransactionId Mandatory when iStartFilter "byTransactionId" is used or iSessionIndex = 4 and iStartMode is MODELESS iInventoryTransactionIdSequence Not Mandatory. iOwner Not Mandatory. iBuyFromBusinessPartner Not Mandatory. Output: for iStartMode MODAL:

	oItem	- item of selected transaction
	oWarehouse	- warehouse of selected transaction
	oTransactionDate	- transaction date of selected transaction
	oSequence	- sequence of selected transaction
	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
	Return: 0	- Session started
	<> 0	- Error

Public Interfaces for OrderInventoryTransaction

The following functions are available:

[OrderInventoryTransaction.StartOverview](#)

OrderInventoryTransaction.StartOverview

DLL:	whextinrapi		
	This function is available from KB2274419 .		
Syntax:	<pre>long OrderInventoryTransaction.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcncmp iOrderCompany, domain tckoor iTypeOfOrder, domain tcorno iOrder, domain tcpono iOrderLine, domain tcdate iTransactionDate, domain tcmcs.long iSequence, domain tcitem iItem, domain tccwar iWarehouse, ref domain tcncmp oOrderCompany, ref domain tckoor oTypeOfOrder, ref domain tcorno oOrder, ref domain tcpono oOrderLine, ref domain tcdate oTransactionDate,</pre>		

	<pre> ref domain tcmcs.long oSequence, ref domain tcitem oItem, ref domain tccwar oWarehouse, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session Order - Inventory Transactions (whinrl51lm000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend A specific query to be used when zooming to this session. iOrderCompany Mandatory when iStartMode is MODELESS iTypeOfOrder Mandatory when iStartMode is MODELESS iOrder Mandatory when iStartMode is MODELESS iOrderLine Not mandatory. iTransactionDate Not mandatory. iSequence Not mandatory. iItem Not mandatory. iWarehouse Not mandatory. Output: for iStartMode MODAL: oOrderCompany - order company of selected transaction oTypeOfOrder - type of orderd of selected transaction oOrder - order of selected transaction oOrderLine - order line of selected transaction oTransactionDate - transaction date of selected transaction oSequence - sequence of selected transaction oItem - item of selected transaction oWarehouse - warehouse of selected transaction oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

Public Interfaces for ItemIssueByPeriod

The following functions are available:

[ItemIssueByPeriod.StartDetail](#)

[ItemIssueByPeriod.StartOverview](#)

ItemIssueByPeriod.StartDetail

DLL:	whextinrapi	
	This function is available from KB3625698 .	
Syntax:	<pre>long ItemIssueByPeriod.StartDetail(long iStartMode, domain tcitem iItem, domain tcyrno iYear, domain tcccp.peri iPeriod, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: This function starts the detail session Item Issue by Period (whinr1520m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variable form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iItem iYear iPeriod Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.	

ItemIssueByPeriod.StartOverview

DLL:	whextinrapi
	This function is available from KB3625698 .
Syntax:	<pre> long ItemIssueByPeriod.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, domain tcyrno iYear, domain tcccp.peri iPeriod, ref domain tcitem oItem, ref domain tcyrno oYear, ref domain tcccp.peri oPeriod, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl This function starts the overview session Item Issue by Period (whinrl520m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used iSessionIndex Not Used iQueryExtend Not Used iItem Mandatory iYear Mandatory iPeriod Mandatory Output: for iStartMode MODAL: oItem - Item of selected record oYear - Year of selected record oPeriod - Period of selected record oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

Public Interfaces for PlannedInventoryTransaction

The following functions are available:

[PlannedInventoryTransaction.StartDetail](#)

[PlannedInventoryTransaction.StartOverview](#)

PlannedInventoryTransaction.StartDetail

DLL:	whextinpapi
Syntax:	<pre> long PlannedInventoryTransaction.StartDetail(long iStartMode, domain tckoor iOriginatingTypeOfOrder, domain tcorno iOrder, domain tckotr iTransactionType, domain tcpono iOrderLine, domain tcpono iOrderLineSequence, domain tcpono iBillOfMaterialLine, domain tcpono iDistributionLine, domain tcuef.effn iEffectivityUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the detail session Planned Inventory Transactions (whinp1500m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iOriginatingTypeOfOrder iOrder iTransactionType iOrderLine iOrderLineSequence iBillOfMaterialLine iDistributionLine iEffectivityUnit Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the

	oExceptionID Return: 0 <> 0	oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Session started - Error

PlannedInventoryTransaction.StartOverview

DLL:	whextinpapi
Syntax:	<pre> long PlannedInventoryTransaction.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tckoor iOriginatingTypeOfOrder, domain tcorno iOrder, domain tckotr iTransactionType, domain tcpono iOrderLine, domain tcpono iOrderLineSequence, domain tcpono iBillofMaterialLine, domain tcpono iDistributionLine, domain tcuef.effn iEffectivityUnit, domain tcitem iItem, domain tctrns.date iTransactionDate, domain tcmcs.long iOrderPriority, domain tccwar iWarehouse, domain tccprj iProject, domain tcpdm.cspa iElement, domain tcpdm.cact iActivity, domain tcptc.cstl iExtension, domain tccpcp iCostComponent, ref domain tckoor oOriginatingTypeOfOrder, ref domain tcorno oOrder, ref domain tckotr oTransactionType, ref domain tcpono oOrderLine, ref domain tcpono oOrderLineSequence, ref domain tcpono oBillofMaterialLine, ref domain tcpono oDistributionLine, ref domain tcuef.effn oEffectivityUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the overview session Planned Inventory Transactions (whinp1500m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter

	Specifies the start filter that is to be applied to the started session. Possible values are: "byOriginatingTypeOfOrder": data is displayed by Originating Type of Order session will be started on index 1 view field: Originating Type of Order "byItemTransactionDate": data is displayed by Item and Transaction Date session will be started on index 2 view field: Item "byItemEffectivityUnit": data is displayed by Item and Effectivity Unit session will be started on index 3 view field: Item "byItemWarehouse": data is displayed by Item and Warehouse session will be started on index 4 view fields: Item and Warehouse "byProjectOrder": data is displayed by Project, Element, Activity, and Order fields session will be started on session index 5 view field: Project "byItemEffectivityUnitProject": data is displayed by Item and Effectivity Unit and Project related fields session will be started on index 6 view field: Item "byProjectPeggedOrderLine": data is displayed by Order fields session will be started on session index 7 view fields: Originating Type of Order, Order, Line, Sequence and Transaction Type "byWarehouseItem": data is displayed by Warehouse and Item session will be started on index 8 view field: Warehouse iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iOriginatingTypeOfOrder Mandatory when iStartFilter "byOriginatingTypeOfOrder" or "byProjectPeggedOrderLine" is used or iSessionIndex is 1 or 7 and iStartMode is MODELESS iOrder Mandatory when iStartFilter "byProjectPeggedOrderLine" is used or iSessionIndex = 7, and iStartMode is MODELESS iTransactionType Mandatory when iStartFilter "byProjectPeggedOrderLine" is used or iSessionIndex = 7, and iStartMode is MODELESS iOrderLine Not mandatory. iOrderLineSequence Not mandatory. iBillOfMaterialLine Not mandatory iDistributionLine

	Not mandatory iEffectivityUnit Not mandatory. iItem Mandatory when iStartFilter "byItemTransactionDate", "byItemEffectivityUnit", "byItemWarehouse" or "byItemEffectivityUnitProject" is used or when iSessionIndex = 2, 3, 4 or 6 and iStartMode is MODELESS iTransactionDate Not mandatory. iOrderPriority Not mandatory. iWarehouse Not mandatory. iProject Not mandatory. iElement Not mandatory. iActivity Not mandatory. iExtension Not mandatory. iCostComponent Not mandatory. Output: for iStartMode MODAL: oOriginatingTypeOfOrder - Originating Order Type of selected transaction oOrder - Order of selected transaction oTransactionType - Transaction type of selected transaction oOrderLine - Order Line of selected transaction oOrderLineSequence - Order Line Sequence of selected transaction oBillofMaterialLine - BOM Line of selected transaction oDistributionLine - Distribution Line of selected transaction oEffectivityUnit - Effectivity unit of selected transaction oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

Public Interfaces for Inventory

The following functions are available:

[Inventory.CalculateTimePhasedAvailability](#)[Inventory.ChangeMAUC](#)[Inventory.ChangeMAUCCostDetails](#)[Inventory.DetermineCompanyOwnedValueDependingOnValuationBasis](#)[Inventory.DetermineQuantity](#)[Inventory.DetermineQuantityAndValue](#)[Inventory.DetermineQuantityAndValueV2](#)[Inventory.GetAnonymousInventoryQuantities](#)[Inventory.GetTimePhasedAvailability](#)[Inventory.GetTimePhasedAvailabilityV2](#)[Inventory.Start360](#)

Inventory.CalculateTimePhasedAvailability

DLL:	whextinpapi															
	This function is available from KB2300987 .															
Syntax:	<pre>long Inventory.CalculateTimePhasedAvailability(domain tcitem iItem, domain tcuef.effn iEffectivityUnit, domain tccwar iWarehouse, domain tceмм.clus iCluster, domain tcguid iSpecification, domain tctrns.date iDate, domain tcyesno iSkipPlannedReceipts, ref domain tcqivl oAvailableQuantity, ref domain tcqivl oCurrentlyBlockedQuantity, ref domain tccuni oInventoryUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>															
Usage:	Expl: This function will read the available quantity and the currently blocked quantity from table Warehouse - Item Inventory (whwmd215). The available quantity will be calculated up to and including the given date (whwmd215 + whinp100). Depending on the values of warehouse and cluster the following inventory is calculated: <table><tr><th>Warehouse</th><th>Cluster</th><th>Result</th></tr><tr><td>-----</td><td>-----</td><td>-----</td></tr><tr><td>""</td><td>Filled</td><td>1</td></tr><tr><td>""</td><td>""</td><td>2</td></tr><tr><td>Filled</td><td>Any value</td><td>3</td></tr></table>	Warehouse	Cluster	Result	-----	-----	-----	""	Filled	1	""	""	2	Filled	Any value	3
Warehouse	Cluster	Result														
-----	-----	-----														
""	Filled	1														
""	""	2														
Filled	Any value	3														

	1 = Aggregated inventory in the specified cluster 2 = Aggregated inventory in the empty cluster. 3 = Inventory in the specified warehouse Only warehouses of type Normal, Shop Floor and Consignment are taken into account. If specification is filled, then the inventory levels will not be read from table whwmd215 but from table whwmd219 (Inventory by Specification). Input: iItem - Item (Mandatory) iEffectivityUnit - Effectivity Unit (Optional) iWarehouse - Warehouse (Optional) iCluster - Cluster (Optional) iSpecification - Specification (Optional) iDate - Date (Mandatory) iSkipPlannedReceipts - Skip planned receipts (Yes/No) Output: oAvailableQuantity - Available quantity expressed in the inventory unit. oCurrentlyBlockedQuantity - Currently blocked quantity expressed in the inventory unit. oInventoryUnit - Inventory Unit Return: 0 - Inventory availability could not be calculated. <> 0 - Error

Inventory.ChangeMAUC

DLL:	whextinaapi
	This function is available from KB2327731 .
Syntax:	<pre>long Inventory.ChangeMAUC (domain whina.mcby iMAUCCorrectionBy, domain tccwar iWarehouse, domain whina.valg iWarehouseValuationGroup, domain tcitem iItem, domain tcemm.grid iEnterpriseUnit, domain tccprj iProject, domain tccdis iReason, domain tcccur iReportCurrency, domain tcrtyp iExchangeRateType, domain whina.uort iUsageOfRates, boolean iSetNewMAUC, domain tccopr iNewMAUCInReportCurrency, boolean iSetNewMAUH, domain tcphrs iNewMAUH, domain tcyesno iOnlyByWarehouseValuationGroup, domain tcyesno iOnlyWithMethodMAUC, domain tcyesno iIncludeServiceRejectWarehouses, ref oDataProcessed, ref domain tcccur oReportCurrency, ref domain tcccur oHomeCurrencyArray() fixed, ref domain tccopr oOldMAUC, ref domain tccopr oOldHomeMAUCArray(),</pre>

	ref domain tcphrs oOldMAUH, ref domain tccopr oNewMAUC, ref domain tccopr oNewHomeMAUCArray(), ref domain tcphrs oNewMAUH, ref long oNumberOfCostComponents, ref domain tccpcp oCostCompArray() fixed, ref domain tccopr oOldCostCompMAUCReportCurrencyArray(), ref domain tccopr oOldCostCompMAUCHomeCurrency1Array(), ref domain tccopr oOldCostCompMAUCHomeCurrency2Array(), ref domain tccopr oOldCostCompMAUCHomeCurrency3Array(), ref domain tcphrs oOldCostCompMAUHArray(), ref domain tccopr oNewCostCompMAUCReportCurrencyArray(), ref domain tccopr oNewCostCompMAUCHomeCurrency1Array(), ref domain tccopr oNewCostCompMAUCHomeCurrency2Array(), ref domain tccopr oNewCostCompMAUCHomeCurrency3Array(), ref domain tcphrs oNewCostCompMAUHArray(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This Public Interface will change the MAUC for the given selection. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Output arrays will be allocated, so a free.mem() must be used to free the memory which is used by these arrays. Input: iMAUCCorrectionBy Mandatory iWarehouse Mandatory, if iMAUCCorrectionBy is by Warehouse else ignored. iWarehouseValuationGroup Mandatory, if iMAUCCorrectionBy is by Warehouse Valuation Group else ignored. iItem Mandatory iEnterpriseUnit Mandatory if standard cost per enterprise unit for item is set. Otherwise it should be empty. iProject Optional iReason Mandatory iReportCurrency Mandatory iExchangeRateType Mandatory iUsageOfRates Mandatory iSetNewMAUC Mandatory iNewMAUCInReportCurrency Used when iSetNewMAUC is True iSetNewMAUC Mandatory iNewMAUH Used when iSetNewMAUH is True iOnlyByWarehouseValuationGroup Mandatory iOnlyWithMethodMAUC Mandatory iIncludeServiceRejectWarehouses Mandatory Output: oDataProcessed - true: MAUC has changed. false: MAUC is not changed. oReportCurrency oHomeCurrencyArray oOldMAUC - MAUC value before correction in home or option set currency. oOldHomeMAUCArray - MAUC values per home currency before correction. oOldMAUH - MAUH value before correction. oNewMAUC - MAUC value after correction in home or option set currency. oNewHomeMAUCArray - MAUC values per home currency after

	correction. - MAUH value after correction. oNewMAUH oNumberOfCostComponents oCostCompArray oOldCostCompMAUCReportCurrencyArray oOldCostCompMAUCHomeCurrency1Array oOldCostCompMAUCHomeCurrency2Array oOldCostCompMAUCHomeCurrency3Array oOldCostCompMAUHArray oNewCostCompMAUCReportCurrencyArray oNewCostCompMAUCHomeCurrency1Array oNewCostCompMAUCHomeCurrency2Array oNewCostCompMAUCHomeCurrency3Array oNewCostCompMAUHArray oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Inventory.ChangeMAUCCostDetails

DLL:	whextinaapi
	This function is available from KB3501684 .
Syntax:	<pre> long Inventory.ChangeMAUCCostDetails(domain whina.mcby iMAUCCorrectionBy, domain tccwar iWarehouse, domain whina.valg iWarehouseValuationGroup, domain tcitem iItem, domain tcemm.grid iEnterpriseUnit, domain tccprj iProject, domain tccdis iReason, domain tcccur iReportCurrency, domain tertyp iExchangeRateType, domain whina.uort iUsageOfRates, boolean iSetNewMAUC, long iNewMAUCNumberOfCostComponents, ref domain tccpcp iNewMAUCCostCompArray() fixed, ref domain tccopr iNewMAUCReportCurrencyArray(), boolean iSetNewMAUH, long iNewMAUHNumberOfCostComponents, ref domain tccpcp iNewMAUHCostCompArray() fixed, ref domain tcphrs iNewMAUHArray(), domain tcyesno iOnlyByWarehouseValuationGroup, domain tcyesno iOnlyWithMethodMAUC, domain tcyesno iIncludeServiceRejectWarehouses, ref boolean oDataProcessed, ref domain tcccur oReportCurrency, ref domain tcccur oHomeCurrencyArray() fixed, ref domain tccopr oOldMAUC, </pre>

	ref domain tccopr oOldHomeMAUCArray(), ref domain tcphrs oOldMAUH, ref domain tccopr oNewMAUC, ref domain tccopr oNewHomeMAUCArray(), ref domain tcphrs oNewMAUH, ref long oNumberOfCostComponents, ref domain tccpcp oCostCompArray() fixed, ref domain tccopr oOldCostCompMAUCReportCurrencyArray(), ref domain tccopr oOldCostCompMAUCHomeCurrency1Array(), ref domain tccopr oOldCostCompMAUCHomeCurrency2Array(), ref domain tccopr oOldCostCompMAUCHomeCurrency3Array(), ref domain tcphrs oOldCostCompMAUHArray(), ref domain tccopr oNewCostCompMAUCReportCurrencyArray(), ref domain tccopr oNewCostCompMAUCHomeCurrency1Array(), ref domain tccopr oNewCostCompMAUCHomeCurrency2Array(), ref domain tccopr oNewCostCompMAUCHomeCurrency3Array(), ref domain tcphrs oNewCostCompMAUHArray(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)																																																		
Usage:	Expl: This Public Interface will change the MAUC for the given selection on cost component level. Arrays must only be filled for the cost components which need to be changed. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Output arrays will be allocated, so a free.mem() must be used to free the memory which is used by these arrays. Input: <table> <tr> <td>iMAUCCorrectionBy</td><td>Mandatory</td></tr> <tr> <td>iWarehouse</td><td>Mandatory, if iMAUCCorrectionBy is by Warehouse else ignored.</td></tr> <tr> <td>iWarehouseValuationGroup</td><td>Mandatory, if iMAUCCorrectionBy is by Warehouse Valuation Group else ignored.</td></tr> <tr> <td>iItem</td><td>Mandatory</td></tr> <tr> <td>iEnterpriseUnit</td><td>Mandatory if standard cost per enterprise unit for item is set. Otherwise it should be empty.</td></tr> <tr> <td>iProject</td><td>Optional</td></tr> <tr> <td>iReason</td><td>Mandatory</td></tr> <tr> <td>iReportCurrency</td><td>Mandatory when iSetNewMAUC is True</td></tr> <tr> <td>iExchangeRateType</td><td>Mandatory</td></tr> <tr> <td>iUsageOfRates</td><td>Mandatory</td></tr> <tr> <td>iSetNewMAUC</td><td>Mandatory</td></tr> <tr> <td>iNewMAUCNumberOfCostComponents</td><td>Used when iSetNewMAUC is True</td></tr> <tr> <td>iNewMAUCCostCompArray()</td><td>Used when iSetNewMAUC is True</td></tr> <tr> <td>iNewMAUCReportCurrencyArray()</td><td>Used when iSetNewMAUC is True</td></tr> <tr> <td>iSetNewMAUH</td><td>Mandatory</td></tr> <tr> <td>iNewMAUHNumberOfCostComponents</td><td>Used when iSetNewMAUH is True</td></tr> <tr> <td>iNewMAUHCostCompArray()</td><td>Used when iSetNewMAUH is True</td></tr> <tr> <td>iNewMAUHArray()</td><td>Used when iSetNewMAUH is True</td></tr> <tr> <td>iOnlyByWarehouseValuationGroup</td><td>Mandatory</td></tr> <tr> <td>iOnlyWithMethodMAUC</td><td>Mandatory</td></tr> <tr> <td>iIncludeServiceRejectWarehouses</td><td>Mandatory</td></tr> </table> Output: <table> <tr> <td>oDataProcessed</td><td>- true: MAUC has changed. false: MAUC is not changed.</td></tr> <tr> <td>oReportCurrency</td><td></td></tr> <tr> <td>oHomeCurrencyArray</td><td></td></tr> <tr> <td>oOldMAUC</td><td>- MAUC value before correction in</td></tr> </table>	iMAUCCorrectionBy	Mandatory	iWarehouse	Mandatory, if iMAUCCorrectionBy is by Warehouse else ignored.	iWarehouseValuationGroup	Mandatory, if iMAUCCorrectionBy is by Warehouse Valuation Group else ignored.	iItem	Mandatory	iEnterpriseUnit	Mandatory if standard cost per enterprise unit for item is set. Otherwise it should be empty.	iProject	Optional	iReason	Mandatory	iReportCurrency	Mandatory when iSetNewMAUC is True	iExchangeRateType	Mandatory	iUsageOfRates	Mandatory	iSetNewMAUC	Mandatory	iNewMAUCNumberOfCostComponents	Used when iSetNewMAUC is True	iNewMAUCCostCompArray()	Used when iSetNewMAUC is True	iNewMAUCReportCurrencyArray()	Used when iSetNewMAUC is True	iSetNewMAUH	Mandatory	iNewMAUHNumberOfCostComponents	Used when iSetNewMAUH is True	iNewMAUHCostCompArray()	Used when iSetNewMAUH is True	iNewMAUHArray()	Used when iSetNewMAUH is True	iOnlyByWarehouseValuationGroup	Mandatory	iOnlyWithMethodMAUC	Mandatory	iIncludeServiceRejectWarehouses	Mandatory	oDataProcessed	- true: MAUC has changed. false: MAUC is not changed.	oReportCurrency		oHomeCurrencyArray		oOldMAUC	- MAUC value before correction in
iMAUCCorrectionBy	Mandatory																																																		
iWarehouse	Mandatory, if iMAUCCorrectionBy is by Warehouse else ignored.																																																		
iWarehouseValuationGroup	Mandatory, if iMAUCCorrectionBy is by Warehouse Valuation Group else ignored.																																																		
iItem	Mandatory																																																		
iEnterpriseUnit	Mandatory if standard cost per enterprise unit for item is set. Otherwise it should be empty.																																																		
iProject	Optional																																																		
iReason	Mandatory																																																		
iReportCurrency	Mandatory when iSetNewMAUC is True																																																		
iExchangeRateType	Mandatory																																																		
iUsageOfRates	Mandatory																																																		
iSetNewMAUC	Mandatory																																																		
iNewMAUCNumberOfCostComponents	Used when iSetNewMAUC is True																																																		
iNewMAUCCostCompArray()	Used when iSetNewMAUC is True																																																		
iNewMAUCReportCurrencyArray()	Used when iSetNewMAUC is True																																																		
iSetNewMAUH	Mandatory																																																		
iNewMAUHNumberOfCostComponents	Used when iSetNewMAUH is True																																																		
iNewMAUHCostCompArray()	Used when iSetNewMAUH is True																																																		
iNewMAUHArray()	Used when iSetNewMAUH is True																																																		
iOnlyByWarehouseValuationGroup	Mandatory																																																		
iOnlyWithMethodMAUC	Mandatory																																																		
iIncludeServiceRejectWarehouses	Mandatory																																																		
oDataProcessed	- true: MAUC has changed. false: MAUC is not changed.																																																		
oReportCurrency																																																			
oHomeCurrencyArray																																																			
oOldMAUC	- MAUC value before correction in																																																		

	home or option set currency. oOldHomeMAUCArray - MAUC values per home currency before correction. oOldMAUH - MAUH value before correction. oNewMAUC - MAUC value after correction in home or option set currency. oNewHomeMAUCArray - MAUC values per home currency after correction. oNewMAUH - MAUH value after correction. oNumberOfCostComponents oCostCompArray oOldCostCompMAUCReportCurrencyArray oOldCostCompMAUCHomeCurrency1Array oOldCostCompMAUCHomeCurrency2Array oOldCostCompMAUCHomeCurrency3Array oOldCostCompMAUHArray oNewCostCompMAUCReportCurrencyArray oNewCostCompMAUCHomeCurrency1Array oNewCostCompMAUCHomeCurrency2Array oNewCostCompMAUCHomeCurrency3Array oNewCostCompMAUHArray oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Inventory.DetermineCompanyOwnedValueDependingOnValuationBasis

DLL:	whextinaapi
	This function is available from KB3544552 .
Syntax:	<pre>long Inventory.DetermineCompanyOwnedValueDependingOnValuationBasis (domain tcitem iItem, domain tccwar iWarehouse, boolean iSpecificAttributeSet, domain tcatse iAttributeSet, domain tcqivl iCompanyOwnedInventory, domain whina.valb iValuationBasis, boolean iByAttributeSet, domain whina.prap iLevelMarketPrice, domain whina.prap iLevelPurchasePrice, domain tcprus iPriceToBeUsed, domain tcdte iValuationDate, boolean iIncludeAntedated, domain tccur iCurrency, domain tcrtyp iRateType, ref long oNumberOfCostComponents,</pre>

Public Interfaces for Warehousing

	<pre> ref domain tccpcp oCostComponents() fixed, ref domain tcamnt oCompanyOwnedInventoryValue(), ref domain tcamnt oCompanyOwnedInventoryTotalValue, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl: This Public Interface will determine the inventory value on a certain date, based on a defined valuation basis (so not based on actual Inventory Valuation Method). It is only meant for company owned inventory. Pre: N.A. Post: oCostComponents and oCompanyOwnedInventoryValue will be allocated, so a free.mem() must be used to free the memory which is used by these arrays. Input: iItem The item for which inventory value needs to be determined. This is mandatory to fill. iWarehouse The warehouse for which inventory value needs to be determined. This is mandatory to fill. iSpecificAttributeSet Indicator (True/False) if inventory value needs to be determined for a specific attribute set. This indicator can only be set when iItem is dimension controlled. This is mandatory to fill. iAttributeSet The attribute set for which inventory value needs to be determined. This is mandatory to fill if iSpecificAttributeSet is set to True. iCompanyOwnedInventory The company owned inventory (quantity) on specific date (iValuationDate). This is the quantity as returned by public interface Inventory.DetermineQuantityAndValue (any version) and/or Inventory.DetermineQuantity. iValuationBasis The Valuation Basis against which the inventory value should be reported. Possible options: * MAUC by Warehouse (whina.valb.mauc) * Standard Cost (whina.valb.ftp) * Market Value Including Surcharges (whina.valb.market.incl.sur) * Market Value Excluding Surcharges (whina.valb.market.excl.sur) * Average Purchase Price Including Surcharges (whina.valb.aver.incl.sur) * Average Purchase Price Excluding Surcharges (whina.valb.aver.excl.sur) This is mandatory to fill iByAttributeSet Indicator (True/False) if valuation basis should be based on Attribute Set Level if applicable, else Item Level is used. This indicator can only be set to True when: - iItem is dimension controlled. - iSpecificAttributeSet is True This is mandatory to fill iLevelMarketPrice The Level Market Price should be used to indicate at </pre>

	which level the market price is searched. Possible options: * Item (whina.prpa.item) Price search is done within iItem. * Item/Warehouse (whina.prpa.itwh) Price search is done within iItem and iWarehouse. * Item/Warehouse Valuation Group (whina.prpa.itwv) Price search is done within iItem and the valuation group of iWarehouse. * Item/Site (whina.prpa.itsi) Price search is done within iItem and the site of iWarehouse. This is mandatory to fill if iValuationBasis is Market Value Including Surcharges or Market Value Excluding Surcharges iLevelPurchasePrice The Level Purchase Price should be used to indicate at which level the purchase price is searched. Possible options: * Item (whina.prpa.item) Price search is done within iItem. * Item/Site (whina.prpa.itsi) Price search is done within iItem and the valuation group of iWarehouse. This is mandatory to fill if iValuationBasis is Average Purchase Price Including Surcharges or Average Purchase Price Excluding Surcharges iPriceToBeUsed Price Used if purchase receipts exist after the last Market Value. Possible options: * Last Available Market Price (tcprus.market) The price is taken from the market values. * Last Available Order Price (tcprus.order) The price is taken from the receipt transactions. This is mandatory to fill if iValuationBasis is Market Value Including Surcharges or Market Value Excluding Surcharges iValuationDate The date for which inventory value needs to be determined. This is optional to fill. When leaving this empty the current date will be used. iIncludeAntedated Indicator (True/False) if antedated transactions should be included or not. Antedated transactions are transactions having transaction date before the valuation date and actual log date after the valuation date. This is mandatory to fill. iCurrency The currency in which the inventory value needs to be expressed. This is mandatory to fill. iRateType The rate type for which inventory value needs to be determined. This is optional to fill. If empty the internal rate type is used. Output: oNumberOfCostComponents The number of cost components.

	oCostComponents Array with the found cost components. oCompanyOwnedInventoryValue, Array with the found inventory values by cost component. oCompanyOwnedInventoryTotalValue, The total inventory value. oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - The inventory and inventory value have been found successfully. <> 0 - Error. The inventory and inventory value could not be determined.

Inventory.DetermineQuantity

DLL:	whextinaapi
Syntax:	<pre> long Inventory.DetermineQuantity(domain tcitem iItem, domain tccwar iWarehouse, domain tcowns iOwnership, boolean iSpecificOwnerBuyFromBusinessPartner, domain tccom.bpid iOwner, domain tccom.bpid iBuyFromBusinessPartner, boolean iSpecificProject, domain tccprj iProject, boolean iSpecificLot, domain tcclot iLot, boolean iSpecificSerial, domain tcibd.sern iSerial, boolean iQuarantineInventory, domain tcdate iValuationDate, boolean iIncludeAntedated, ref domain tcqivl oInventory, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface will determine the inventory quantity at a certain date. Pre: N.A. Post: N.A. Input: iItem The item for which inventory quantity needs to be determined. This is mandatory to fill. iWarehouse The warehouse for which inventory quantity needs to be determined. This is mandatory to fill. iOwnership The ownership for which inventory quantity needs to be determined. This is mandatory to fill. Possible options: * Company Owned (tcowns.comp.owned)

	* Consigned (tcowns.consigned) * Customer Owned (tcowns.cust.owned) iSpecificOwnerBuyFromBusinessPartner Indicator (True/False) if inventory quantity needs to be determined for a specific Owner and Buy from Business Partner. This indicator can only be set to True when iOwnership is Consigned or Customer Owned. This is mandatory to fill. iOwner The owner for which inventory quantity needs to be determined. This is mandatory to fill when iSpecificOwnerBuyFromBusinessPartner is set to True iBuyFromBusinessPartner The buy-from business partner for which inventory quantity needs to be determined. This is mandatory to fill when iSpecificOwnerBuyFromBusinessPartner is set to True and iOwnership is Consigned. iSpecificProject Indicator (True/False) if inventory quantity needs to be determined for a specific project. This is mandatory to fill. iProject The project for which inventory quantity needs to be determined. This is not mandatory to fill. When iSpecificProject is set to False then all inventory will be taken into account regardless of project. When iSpecificProject is set to True and iProject is filled then the inventory of the specified project will be taken into account. When iSpecificProject is set to True and iProject is not filled then all not project related inventory will be taken into account. iSpecificLot Indicator (True/False) if inventory quantity needs to be determined for a specific lot. This indicator can only be set when: - iOwnership is Company Owned and - item is defined as lot controlled in inventory and - iSpecificProject = False This is mandatory to fill. iLot The lot for which inventory quantity needs to be determined. This is mandatory to fill if iSpecificLot is set to True. iSpecificSerial Indicator (True/False) if inventory quantity needs to be determined for a specific serial. This indicator can only be set when: - iOwnership is Company Owned and - item is defined as serialized in inventory and - iSpecificProject = False This is mandatory to fill. iSerial The serial for which inventory quantity needs to be determined. This is mandatory to fill if iSpecificSerial is set to True. iQuarantineInventory Indicator if the inventory quantity needs to

	be determined for quarantine inventory. This is mandatory to fill. iValuationDate The date for which inventory quantity needs to be determined. This is not mandatory to fill. When leaving empty the current date will be used. iIncludeAntedated Indicator (True/False) if antedated transactions should be included or not. Antedated transactions are transactions having transaction date before the valuation date and actual log date after the valuation date. This is mandatory to fill. Output: oInventory The determined inventory quantity oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - The inventory and inventory value have been found successfully. <> 0 - Error. The inventory and inventory value could not be determined.

Inventory.DetermineQuantityAndValue

DLL:	whextinaapi
Syntax:	<pre> long Inventory.DetermineQuantityAndValue(domain tcitem iItem, domain tccwar iWarehouse, domain tcowns iOwnership, domain tccom.bpid iOwner, domain tccom.bpid iBuyFromBusinessPartner, domain tcyesno iSpecificProject, domain tccprj iProject, domain tcyesno iQuarantineInventory, domain tcdate iValuationDate, domain tccur iCurrency, domain tcrtyp iRateType, ref domain tcqivl oInventory, ref long oNumberOfCostComponents, ref domain tccpcp oCostComponents() fixed, ref domain tcphrs oInventoryHours(), ref domain tcamnt oInventoryValue(), ref domain tcphrs oInventoryTotalHours, ref domain tcamnt oInventoryTotalValue, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface will determine the inventory and the inventory value based on the valuation method on a certain date. Note: This Public Interface does not include antedated transactions and is replaced by a new version. Use

	Inventory.DetermineQuantityAndValueV2 which has: 1) the option to define whether antedated transactions should be included yes or no. 2) the option to determine inventory quantity and value by specific lot (lot pricing) or serial (serial pricing). Pre: N.a. Post: oCostComponents, oInventoryHours and oInventoryValue will be allocated, so a free.mem() must be used to free the memory which is used by these arrays. Input: iItem The item for which inventory and inventory value needs to be determined. This is mandatory to fill. iWarehouse The warehouse for which inventory and inventory value needs to be determined. This is mandatory to fill. iOwnership The ownership for which inventory and inventory value needs to be determined. This is mandatory to fill. Possible options: * Company Owned (tcowns.comp.owned) * Consigned (tcowns.consigned) * Customer Owned (tcowns.cust.owned) iOwner The owner for which inventory and inventory value needs to be determined. This is mandatory to fill when iOwnership is Consigned or Customer Owned iBuyFromBusinessPartner The buy-from business partner for which inventory and inventory value needs to be determined. This is mandatory to fill when iOwnership is Consigned. iSpecificProject Indicator if inventory and inventory value needs to be determined for a specific project. This is mandatory to fill. iProject The project for which inventory and inventory value needs to be determined. This is not mandatory to fill. When iSpecificProject is set to No then all inventory will be taken into account. When iSpecificProject is set to yes and iProject is filled then the inventory of the specified project will be taken into account. When iSpecificProject is set to yes and iProject is not filled then all not project related inventory will be taken into account. iQuarantineInventory Indicator if the inventory and inventory value needs to be determined for quarantine inventory. This is mandatory to fill. iValuationDate The date for which inventory and inventory value needs to be determined. This is not mandatory to fill. When leaving it empty the current date will be used. iCurrency The currency in which the inventory value needs to be expressed. This is mandatory to fill. iRateType The rate type for which inventory and inventory value needs to be determined. This is not mandatory to fill. If empty the internal rate type is used. Output: oInventory The determined inventory.

	oNumberOfCostComponents The number of cost components. oCostComponents Array with the found cost components. oInventoryHours Array with the found inventory hours. oInventoryValue, Array with the found inventory values. oInventoryTotalHours The total inventory hours. oInventoryTotalValue, The total inventory value. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - The inventory and inventory value have been found successfully. <> 0 - Error. The inventory and inventory value could not be determined.

Inventory.DetermineQuantityAndValueV2

DLL:	whextinaapi
Syntax:	<pre> long Inventory.DetermineQuantityAndValueV2(domain tcitem iItem, domain tccwar iWarehouse, domain tcowns iOwnership, domain tccom.bpid iOwner, domain tccom.bpid iBuyFromBusinessPartner, boolean iSpecificProject, domain tccprj iProject, boolean iSpecificLot, domain tcclot iLot, boolean iSpecificSerial, domain tcibd.sern iSerial, boolean iQuarantineInventory, domain tcdate iValuationDate, boolean iIncludeAntedated, domain tcccur iCurrency, domain tcrtyp iRateType, ref domain tcqivl oInventory, ref long oNumberOfCostComponents, ref domain tccpcp oCostComponents() fixed, ref domain tcphrs oInventoryHours(), ref domain tcamnt oInventoryValue(), ref domain tcphrs oInventoryTotalHours, ref domain tcamnt oInventoryTotalValue, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface will determine the inventory quantity and value based on the inventory valuation method on a

	certain date. Pre: N.A. Post: oCostComponents, oInventoryHours and oInventoryValue will be allocated, so a free.mem() must be used to free the memory which is used by these arrays. Input: iItem The item for which inventory quantity and value needs to be determined. This is mandatory to fill. iWarehouse The warehouse for which inventory quantity and value needs to be determined. This is mandatory to fill. iOwnership The ownership for which inventory quantity and value needs to be determined. Possible options: * Company Owned (tcowns.comp.owned) * Consigned (tcowns.consigned) * Customer Owned (tcowns.cust.owned) This is mandatory to fill. iOwner The owner for which inventory quantity and value needs to be determined. This is mandatory to fill when iOwnership is Consigned or Customer Owned iBuyFromBusinessPartner The buy-from business partner for which inventory quantity and value needs to be determined. This is mandatory to fill when iOwnership is Consigned. iSpecificProject Indicator (True/False) if inventory quantity and value needs to be determined for a specific project. This is mandatory to fill. iProject The project for which inventory quantity and value needs to be determined. This is not mandatory to fill. When iSpecificProject is set to True then all inventory will be taken into account regardless of project. When iSpecificProject is set to False and iProject is filled then the inventory of the specified project will be taken into account. When iSpecificProject is set to True and iProject is not filled then all not project related inventory will be taken into account. iSpecificLot Indicator (True/False) if inventory quantity and value needs to be determined for a specific lot. This indicator can only be set when: - iOwnership is Company Owned and - actual inventory valuation method for item/warehouse combination is Lot Price and - iSpecificProject = False This is mandatory to fill. iLot The lot for which inventory quantity and value needs to be determined. This is mandatory to fill if iSpecificLot is set to True. iSpecificSerial Indicator (True/False) if inventory quantity and value

	needs to be determined for a specific serial. This indicator can only be set when: - iOwnership is Company Owned and - actual inventory valuation method for item/warehouse combination is Serial Price and - iSpecificProject = False This is mandatory to fill. iSerial The serial for which inventory quantity and value needs to be determined. This is mandatory to fill if iSpecificSerial is set to True. iQuarantineInventory Indicator (True/False) if the inventory quantity and value needs to be determined for quarantine inventory. This is mandatory to fill. iValuationDate The date for which inventory and inventory value needs to be determined. This is not mandatory to fill. When leaving this empty the current date will be used. iIncludeAntedated Indicator (True/False) if antedated transactions should be included or not. Antedated transactions are transactions having transaction date before the valuation date and actual log date after the valuation date. This is mandatory to fill. iCurrency The currency in which the inventory value needs to be expressed. This is mandatory to fill. iRateType The rate type for which inventory and inventory value needs to be determined. This is not mandatory to fill. If empty the internal rate type is used. Output: oInventory The determined inventory. oNumberOfCostComponents The number of cost components. oCostComponents Array with the found cost components. oInventoryHours Array with the found inventory hours. oInventoryValue, Array with the found inventory values. oInventoryTotalHours The total inventory hours. oInventoryTotalValue, The total inventory value. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - The inventory and inventory value have been found successfully. <> 0 - Error. The inventory and inventory value could not be determined.

Inventory.GetAnonymousInventoryQuantities

DLL:	whextwmdapi																												
Syntax:	<pre> long Inventory.GetAnonymousInventoryQuantities (domain tccwar iWarehouse, domain whloca iLocation, domain tcitem iItem, domain tcclot iLot, domain tcinvdate iInventoryDate, domain tccuni iStorageUnit, domain whwmd.pkdf iPackageDefinition, ref domain tcqstl oQuantityOnHand, ref domain tcqstl oQuantityAvailable, ref domain tcqstl oQuantityAllocated, ref domain tcqstl oQuantityBlocked, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>																												
Usage:	Expl: This public interface returns the anonymous stock quantities of the specified stock point divided in available, allocated and blocked stock. This function only returns the quantities for the specified inventory structure record. Pre: N.A. Post: N.A. Input: <table border="0"> <tr> <td>iWarehouse</td><td>- Warehouse (Mandatory)</td></tr> <tr> <td>iLocation</td><td>- Location (Mandatory, if warehouse and item are location controlled)</td></tr> <tr> <td>iItem</td><td>- Item (Mandatory)</td></tr> <tr> <td>iLot</td><td>- Lot</td></tr> <tr> <td>iInventoryDate</td><td>- Inventory Date</td></tr> <tr> <td>iStorageUnit</td><td>- Storage Unit (Mandatory)</td></tr> <tr> <td>iPackageDefinition</td><td>- Package definition</td></tr> </table> Only fixed package definitions are allowed. Output: <table border="0"> <tr> <td>oQuantityOnHand</td><td>- Quantity On Hand</td></tr> <tr> <td>oQuantityAvailable</td><td>- Quantity Available</td></tr> <tr> <td>oQuantityAllocated</td><td>- Quantity Allocated</td></tr> <tr> <td>oQuantityBlocked</td><td>- Quantity Blocked</td></tr> <tr> <td>oExceptionMessage</td><td>- The last message if the return value is not equal to 0.</td></tr> <tr> <td></td><td>If more than one message is given, these are present in the oExceptionID</td></tr> <tr> <td>oExceptionID</td><td>- An ID that refers to all error information. Use the functions in Exception to get all relevant information.</td></tr> </table> Return: 0 - Success <> 0 - Error	iWarehouse	- Warehouse (Mandatory)	iLocation	- Location (Mandatory, if warehouse and item are location controlled)	iItem	- Item (Mandatory)	iLot	- Lot	iInventoryDate	- Inventory Date	iStorageUnit	- Storage Unit (Mandatory)	iPackageDefinition	- Package definition	oQuantityOnHand	- Quantity On Hand	oQuantityAvailable	- Quantity Available	oQuantityAllocated	- Quantity Allocated	oQuantityBlocked	- Quantity Blocked	oExceptionMessage	- The last message if the return value is not equal to 0.		If more than one message is given, these are present in the oExceptionID	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
iWarehouse	- Warehouse (Mandatory)																												
iLocation	- Location (Mandatory, if warehouse and item are location controlled)																												
iItem	- Item (Mandatory)																												
iLot	- Lot																												
iInventoryDate	- Inventory Date																												
iStorageUnit	- Storage Unit (Mandatory)																												
iPackageDefinition	- Package definition																												
oQuantityOnHand	- Quantity On Hand																												
oQuantityAvailable	- Quantity Available																												
oQuantityAllocated	- Quantity Allocated																												
oQuantityBlocked	- Quantity Blocked																												
oExceptionMessage	- The last message if the return value is not equal to 0.																												
	If more than one message is given, these are present in the oExceptionID																												
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.																												

Inventory.GetTimePhasedAvailability

DLL:	whextinapi
Syntax:	<pre> long Inventory.GetTimePhasedAvailability(domain tcitem iItem, domain tccwar iWarehouse, domain tcddate iDate, ref domain tcqivl oAvailableQuantity, ref domain tccuni oInventoryUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function retrieves the available inventory on the given date for the given item and warehouse, based on the current inventory and the planned inventory transactions. Inventory of a financial warehouse is not taken into account and will return zero for the available quantity. Input: - iItem - Item (Mandatory) - iWarehouse - Warehouse (Mandatory) - iDate - Date (Mandatory) Output: - oAvailableQuantity - Available Quantity on the given date expressed in the inventory unit. In case the available quantity is negative, the value zero is returned. - oInventoryUnit - Inventory unit. - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Available Inventory retrieved <> 0 - Error

Inventory.GetTimePhasedAvailabilityV2

DLL:	whextinapi
Syntax:	<pre> long Inventory.GetTimePhasedAvailabilityV2(domain tcitem iItem, domain tccwar iWarehouse, domain tcddate iDate, boolean iIncludeOnOrderTransactions, boolean iIncludeIssueTransactions, boolean iAllowNegativeAvailableQuantity, ref domain tcqivl oAvailableQuantity, ref domain tccuni oInventoryUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function retrieves the available inventory on the given date for the given item and warehouse, based on the current inventory and the planned receipt/issue transactions.

	Inventory of a financial warehouse is not taken into account and will return zero for the available quantity. Reasoning Version 2: With the introduction of the input variables iIncludeOnOrderTransactions, iIncludeIssueTransactions and iAllowNegativeAvailableQuantity the user is now much more in control on how the available inventory must be calculated. Input: iItem - Item (Mandatory) iWarehouse - Warehouse (Mandatory) iDate - Date (Mandatory) iIncludeOnOrderTransactions - Include On Order Transactions (Mandatory) - True - Planned receipt transactions are taken into account. - False - Planned receipt transactions are not taken into account. iIncludeIssueTransactions - Include Issue Transactions (Mandatory) - True - Planned issue transactions are taken into account. - False - Planned issue transactions are not taken into account. iAllowNegativeAvailableQuantity - Allow Negative Available Quantity (Mandatory) - True - If available quantity is negative then this value will be returned. - False - If available quantity is negative then the value zero will be returned. Output: oAvailableQuantity - Available Quantity on the given date expressed in the inventory unit. oInventoryUnit - Inventory unit. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Available Inventory retrieved <> 0 - Error

Inventory.Start360

DLL:	whextwmdapi
	This function is available from KB2317999 .
Syntax:	long Inventory.Start360(long iStartMode,

	<pre> domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcyesno iSetFilterFields, domain tcsite iSite, domain tccwar iWarehouse, domain tccitg iItemGroup, domain tcitem iItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl: This function starts the session Inventory 360 (whwmd4300m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "": iSessionIndex Not Used. iQueryExtend A specific query to be used when zooming to this session. iSetFilterFields: Used to set a filter on the Inventory for a Warehouse or an Item Group. Possible values are: Yes: A filter on the inventory for a Warehouse or an Item Group can be applied. No: No filter for Warehouse or Item Group will be applied. Empty: If not set, the saved session defaults will be used and applied for these filter fields. iSite: The Site to be used with iSetFilterFields. iWarehouse: The Warehouse to be used with iSetFilterFields. iItem Group: The Item Group to be used with iSetFilterFields. iItem Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error </pre>

Public Interfaces for SpecificationInventory

The following functions are available:

[SpecificationInventory.GetUnallocatedAvailableQuantity](#)

SpecificationInventory.GetUnallocatedAvailableQuantity

DLL:	whextwmdapi
	This function is available from KB2272169 .
Syntax:	<pre> long SpecificationInventory.GetUnallocatedAvailableQuantity(domain tccwar iWarehouse, domain tcitem iItem, domain tcuef.effn iEffectivityUnit, ref domain tcqivl oUnallocatedAvailableQuantity, ref domain tccuni oInventoryUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function determines the unallocated available quantity for an item in a warehouse. When inventory is allocated to a demand peg, it is not available as unallocated inventory. When effectivity units are in use, this public interface will determine if the iEffectivityUnit is interchangeable with the effectivity unit which is found for the iItem and iWarehouse and will include the available quantity in the oAvailableUnallocatedQuantity. When the iEffectivityUnit is passed as empty (zero) the same rule applies. Pre: N.A. Post: N.A. Input: iWarehouse - Warehouse (Mandatory) iItem - Item (Mandatory) iEffectivityUnit - Effectivity Unit Output: oUnallocatedAvailableQuantity - Unallocated available quantity in Inventory Unit oInventoryUnit - Inventory Unit Return: 0 - Success <> 0 - Error

Public Interfaces for InventoryCommitment

The following functions are available:

[InventoryCommitment.AllowedForOrderLine](#)[InventoryCommitment.CalculateCommitmentDate](#)[InventoryCommitment.CommitInventoryToOrderLine](#)[InventoryCommitment.ConsumeFromBuffer](#)[InventoryCommitment.CreateAllocationBuffer](#)[InventoryCommitment.Generate](#)[InventoryCommitment.GetAvailableQuantityToCommit](#)

InventoryCommitment.AllowedForOrderLine

DLL:	whextinpapi		
	This function is available from KB2308183 .		
Syntax:	<pre> long InventoryCommitment.AllowedForOrderLine (domain whinp.corg iOrderOrigin, domain tcorno iOrderNumber, domain tcpono iOrderLine, domain tcpono iOrderSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>		
Usage:	Expl: This function checks if inventory commitment for the given order line is allowed. Commitment can only take place in case no quantity has been adviced yet. (Status of warehouse order line is 'open') Input: - iOrderOrigin - Order Origin (Mandatory) - iOrderNumber - Order Number (Mandatory) - iOrderLine - Order Line - iOrderSequence - Order Sequence Output: - oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - No error has been detected. Inventory Commitment is allowed. <> 0 - An Error is detected.		

InventoryCommitment.CalculateCommitmentDate

DLL:	whextinpapi
	This function is available from KB2310180 .
Syntax:	<pre> long InventoryCommitment.CalculateCommitmentDate(domain whinp.corg iOrderOrigin, domain tccwar iWarehouse, domain tcitem iItem, domain tctrns.date iPlannedDeliveryDate, ref domain tctrns.date oCommitmentDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function calculates the commitment date. If generation of inventory commitment is based on planned delivery date (whinp000.icbo), then this function will return zero as Commitment Date. Input: iOrderOrigin - Order Origin (Mandatory) iWarehouse - Warehouse (Mandatory) iItem - Item (Mandatory) iPlannedDeliveryDate - PlannedDeliveryDate (Mandatory) Output: oCommitmentDate - Commitment Date oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - No error has been detected. <> 0 - An Error is detected.

InventoryCommitment.CommitInventoryToOrderLine

DLL:	whextinpapi
	This function is available from KB2310181 .
Syntax:	<pre> long InventoryCommitment.CommitInventoryToOrderLine(domain whinp.corg iOrderOrigin, domain tcorno iOrderNumber, domain tcpono iOrderLine, domain tcpono iOrderSequence, domain tcpono iBillofMaterialLine, domain tcitem iItem, domain tcuef.effn iOldEffectivityUnit, domain tcuef.effn iEffectivityUnit, domain tcguid iSpecification, domain tccwar iWarehouse, domain tcqivl iCommittedInOrderUnit, domain tccuni iOrderUnit, domain tcqivl iOrderQuantity, domain tctrns.date iPlannedDeliveryDate, </pre>

		boolean	iInventoryBuffer,
		boolean	iUpdateRelatedOrderLine,
ref	domain	tcqivl	oCommittedInInventoryUnit,
ref	domain	tcqivl	oCommittedInOrderUnit,
ref	domain	tcmcs.s999m	oExceptionMessage mb,
ref	long		oExceptionID)
Usage:	Expl: This function allows to update the inventory committed quantity to an "issue" order like: Sales Orders, Production Orders, Service Orders, PRP Orders, Sales Schedules, Transfer Orders. The function checks the available to commit inventory, and the commitment date. The committed quantity cannot exceed the available to commit quantity, nor the order quantity. A check is executed on the minimum commitment quantity/rate. After commitment, the issue orders (sales orders) are updated accordingly. The new committed quantity becomes iCommittedInOrderUnit (if possible) When iCommittedInOrderUnit = 0.0, the function perform a de-commitment (remove the committed quantity). It is expected that the issue order has been read and checked whether the commitment is allowed from the order standpoint. Input: - iOrderOrigin - Order Origin (Mandatory) - iOrderNumber - Order Number (Mandatory) - iOrderLine - Order Line - iOrderSequence - Order Sequence - iBillOfMaterialLine - Bill of Material Line - iItem - Item (Mandatory) - iOldEffectivityUnit - Old Effectivity Unit - iEffectivityUnit - Effectivity Unit - iSpecification - Specification - iWarehouse - Warehouse (Mandatory) - iCommittedInOrderUnit - New committed quantity (to update the old one) expressed in order units. - iOrderUnit - Order Unit - iOrderQuantity - Order Quantity - iPlannedDeliveryDate - Planned Delivery Date (Mandatory) - iInventoryBuffer - To avoid the check on the available to commit quantity. This is used for "Consumption of Inventory Buffers" ONLY ! - true: the DLL is used for commitment within the consumption of inventory buffers - false: otherwise (common value) iUpdateRelatedOrderLine - Update Related Order Line - true: committed quantity is also updated towards the related order line (e.g. Sales line) - false: otherwise Output: - o.committed.in.inventory.unit - Committed quantity in inventory unit - o.committed.in.order.unit - Committed quantity in order unit - oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information.		

	Return: 0 <> 0	- No error has been detected. - An Error is detected.

InventoryCommitment.ConsumeFromBuffer

DLL:	whextinpapi	
	This function is available from KB3568815 .	
Syntax:	<pre>long InventoryCommitment.ConsumeFromBuffer(domain tcorno iOrderNumberFrom, domain tcpono iOrderLineFrom, domain whinp.corg iOrderOriginTo, domain tcorno iOrderNumberTo, domain tcpono iOrderLineTo, domain tcpono iOrderSequenceTo, domain tcpono iBillOfMaterialLine, domain tcqivl iTransferQuantity, domain tccuni iTransferUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: This function commits inventory that was previously reserved in inventory buffers, to specific orders. Pre: db.retry.point must be set Post: Commit the transaction in case of success Abort the transaction in case of failure Input: iOrderNumberFrom - Mandatory iOrderLineFrom - Optional iOrderOriginTo - Mandatory The following values are allowed: whinp.corg.sales whinp.corg.production whinp.corg.transfer whinp.corg.transfer.man whinp.corg.enterprise.plan whinp.corg.prp whinp.corg.schedule whinp.corg.service whinp.corg.maint.sales whinp.corg.maint.work whinp.corg.customer.claim whinp.corg.supplier.claim iOrderNumberTo - Mandatory iOrderLineTo - Mandatory iOrderSequenceTo - Optional iBillOfMaterialLine - Optional iTransferQuantity - Mandatory iTransferUnit - Mandatory Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information.	

	Return: 0	- No error has been detected.
	<> 0	- An Error is detected.

InventoryCommitment.CreateAllocationBuffer

DLL:	whextinpapi
	This function is available from KB2272147 .
Syntax:	<pre> long InventoryCommitment.CreateAllocationBuffer(domain tccwar iWarehouse, domain tcitem iItem, domain tcuef.effn iEffectivityUnit, domain tccom.bpid iSoldToBusinessPartner, domain tccom.bpid iShipToBusinessPartner, domain tcalbt iAllocationBusinessObjectType, domain tcboid iAllocationBusinessObject, domain tcborf iAllocationBusinessObjectReference, domain tcrefa iAllocationReference mb, domain tccpva iProductVariant, domain tcolid iOptionList, domain tcqivl iQuantityToAllocate, ref domain tcqivl oQuantityAllocated, ref domain tccuni oInventoryUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function generates an allocation buffer. If an allocation buffer already exists for the specification then this buffer will be updated. Input: - iWarehouse - Warehouse (Mandatory) - iItem - Item (Mandatory) - iEffectivityUnit - Effectivity Unit The following input fields are referred to the specification (tcibd420) - iSoldToBusinessPartner - Sold-To Business Partner (tcibd420.albp) - iShipToBusinessPartner - Ship-To Business Partner (tcibd420.alst) - iAllocationBusinessObjectType - Allocation Business Object Type (tcibd420.albt) - iAllocationBusinessObject - Allocation Business Object (tcibd420.albo) - iAllocationBusinessObjectReference - Allocation Business Object Reference (tcibd420.abor) - iAllocationReference - Allocation Reference (tcibd420.alrf) - iProductVariant - Product Variant (tcibd420.cpva) - iOptionList - Option List (tcibd420.olid) - iQuantityToAllocate - Quantity to Allocate expressed in inventory unit (Mandatory) Output: - oQuantityAllocated - Allocated Quantity expressed in inventory unit - oInventoryUnit - Inventory Unit - oExceptionMessage - The last message if the return value is not equal to 0.

	If more than one message is given, these are present in the oExceptionID. - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - No error has been detected. Allocation Buffer has been created, a part has been created of nothing has been created. - An Error is detected.
oExceptionID	
Return: 0	
<> 0	

InventoryCommitment.Generate

DLL:	whextinpapi										
	This function is available from KB3560508 .										
Syntax:	<pre>long InventoryCommitment.Generate(domain whinp.corg iOrderOrigin, domain tcorno iOrderNumber, domain tcpono iOrderLine, domain tcmcs.str15 iDevice, long iProcessingOptionSet, ref boolean oDataProcessed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>										
Usage:	Expl: This function generates inventory commitments for the iOrderOrigin, iOrderNumber and iOrderLine using the defaults or options as provided in the iProcessingOptionSet. Be aware that transaction management is handled within this function. Opening/Closing of the report is handled by this function. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: <table border="0"> <tr> <td>iOrderOrigin</td><td>Mandatory</td></tr> <tr> <td>iOrderNumber</td><td>Optional</td></tr> <tr> <td>iOrderLine</td><td>Optional</td></tr> <tr> <td>iDevice</td><td>Mandatory</td></tr> <tr> <td>iProcessingOptionSet</td><td>Optional, if 0, the default options are applied.</td></tr> </table> A Processing Option Set can be created via a call to function ProcessingOptionSet.Create(). In case iOrderNumber is filled then the following options of the iProcessingOptionSet will be ignored: - selection range fields (From/To) - OrderLineArray The inventory commitments will then be generated for the given iOrderOrigin, iOrderNumber (and specific iOrderLine, if filled). In case option OrderLineArray is set then the selection	iOrderOrigin	Mandatory	iOrderNumber	Optional	iOrderLine	Optional	iDevice	Mandatory	iProcessingOptionSet	Optional, if 0, the default options are applied.
iOrderOrigin	Mandatory										
iOrderNumber	Optional										
iOrderLine	Optional										
iDevice	Mandatory										
iProcessingOptionSet	Optional, if 0, the default options are applied.										

range fields (From/To) and OrderOriginPriorityArray of the iProcessingOptionSet will be ignored.
The inventory commitments will then be generated for the order lines in the array.

Processing Options have a direct relationship with the form fields on session Generate Inventory Commitments (whinp2200m000) and are not explained in further detail here.
Please refer to the session help for additional information.
Processing Options that are set while a required Implemented Software Component is not available are ignored.

Explanation about setting of defaults:

Minimum Value: Minimum value of domain is taken as default value.

Maximum Value: Maximum value of domain is taken as default value.

NAME	TYPE	DEFAULT
GenerateBasedOn	domain whinp.icbo	whinp000.icbo
OrderOriginFrom	domain whinp.corg	Minimum Value
OrderOriginTo	domain whinp.corg	Maximum Value
SoldToBusinessPartnerFrom	domain tccom.bpid	Minimum Value
SoldToBusinessPartnerTo	domain tccom.bpid	Maximum Value
ShipToBusinessPartnerFrom	domain tccom.bpid	Minimum Value
ShipToBusinessPartnerTo	domain tccom.bpid	Maximum Value
OrderNumberFrom	domain tcorno	Minimum Value
OrderNumberTo	domain tcorno	Maximum Value
OrderLineFrom	domain tcpono	Minimum Value
OrderLineTo	domain tcpono	Maximum Value
OperationFrom	domain tcopno	Minimum Value
OperationTo	domain tcopno	Maximum Value
ProductionDateTo	domain tcutcm	Current Date/Time
PlannedDeliveryDateFrom	domain tcdate	Minimum Value
PlannedDeliveryDateTo	domain tcdate	Current Date/Time
RouteFrom	domain tccrte	Minimum Value
RouteTo	domain tccrte	Maximum Value
ProjectFrom	domain tccprj	Minimum Value
ProjectTo	domain tccprj	Maximum Value
WarehouseFrom	domain tccwar	Minimum Value
WarehouseTo	domain tccwar	Maximum Value
EffectivityUnitFrom	domain tcuef.effn	Minimum Value
EffectivityUnitTo	domain tcuef.effn	Maximum Value
ItemFrom	domain tcitem	Minimum Value
ItemTo	domain tcitem	Maximum Value
OrderLineArray	domain ttjson	0
AccordingShippingConstraints	domain tcyesno	tcyesno.yes
CancelForIncompleteOrders	domain tcyesno	tcyesno.yes
OnlyWithProcedureStep	domain tcyesno	tcyesno.yes
OrderOriginPriorityArray	domain ttjson	0
OrderByPlannedDeliveryDate	domain tcyesno	tcyesno.no
ReportName	domain tcmcs.str16	Empty String

JSON Object OrderLineArray has the following structure:

```
"OrderLineArray": [
  {
    "OrderOrigin": 20,
    "OrderNumber": "JSC000030",
    "OrderLine": 10
  },
  {
    "OrderOrigin": 10,
    "OrderNumber": "SLS000012",
    "OrderLine": 20
  }
]
```


	<pre> }] This structure can be created with the following code: OrderLineArray = Json.newArray() OrderLine = Json.newObject() Json.setNumber(OrderLine, "OrderOrigin", 20) Json.setString(OrderLine, "OrderNumber", "JSC000030") Json.setNumber(OrderLine, "OrderLine", 10) Json.setNumber(OrderLine, "OrderSequence", 1) Json.add(OrderLineArray, OrderLine) OrderLine = Json.newObject() Json.setNumber(OrderLine, "OrderOrigin", 10) Json.setString(OrderLine, "OrderNumber", "SLS000012") Json.setNumber(OrderLine, "OrderLine", 10) Json.setNumber(OrderLine, "OrderSequence", 1) Json.add(OrderLineArray, OrderLine) JSON Object OrderOriginPriorityArray has the following structure: "OrderOriginPriorityArray": [{ "OrderOrigin": 10 }, { "OrderOrigin": 20 }] This structure can be created with the following code: OrderOriginPriorityArray = Json.newArray() OrderOrigin = Json.newObject() Json.setNumber(OrderOrigin, "OrderOrigin", 10) Json.add(OrderOriginPriorityArray, OrderOrigin) OrderOrigin = Json.newObject() Json.setNumber(OrderOrigin, "OrderOrigin", 20) Json.add(OrderOriginPriorityArray, OrderOrigin) ReportName only needs to be filled for customized reports, otherwise the standard report is automatically used. ReportName must start with an "r", e.g. "rwhinp220001001" Output: oDataProcessed - true: Inventory commitments generated false: Nothing generated. Return: 0 - No Error <> 0 - Error </pre>

InventoryCommitment.GetAvailableQuantityToCommit

DLL:	whextinpapi
	This function is available from KB2304092 .
Syntax:	<pre> long InventoryCommitment.GetAvailableQuantityToCommit(domain tccwar iWarehouse, domain tcitem iItem, domain tcuef.effn iEffectivityUnit, </pre>

	domain tcguid iSpecification, boolean iForAllocationBuffer, domain tccuni iUnit, ref domain tcqivl oAvailableQuantityToCommit, ref domain tccuni oUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function calculates the available to commit quantity To Commit Quantity = Quantity on Hand - Quantity on Hold - Committed Quantity - Location Allocated Quantity + Committed Quantity in Process Input: iWarehouse - Warehouse (Mandatory) iItem - Item (Mandatory) iEffectivityUnit - Effectivity Unit (Optional) iSpecification - Specification (Optional) iForAllocationBuffer - Indicator (true/false) if calculation must be done for allocation buffer with only allocations (true). Otherwise specify false. (Mandatory) iUnit - Unit in which the Quantity to Commit has to be expressed. When left empty then the inventory unit will be taken. (Optional) Output: oAvailableQuantityToCommit - Available quantity to commit oUnit - The specified unit (i.unit) or when left empty it will be the inventory unit of the item oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Quantity to Commit is calculated <> 0 - Error

Public Interfaces for SiteItemInventory

The following functions are available:

[SiteItemInventory.StartDetail](#)

[SiteItemInventory.StartOverview](#)

SiteItemInventory.StartDetail

DLL:	whextwmdapi
Syntax:	<pre> long SiteItemInventory.StartDetail(long iStartMode, domain tcsite iSite, domain tcitem iItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the detail session Site - Item Inventory (whwmd4504m100). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iSite iItem Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

SiteItemInventory.StartOverview

DLL:	whextwmdapi
Syntax:	<pre> long SiteItemInventory.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcsite iSite, domain tcitem iItem, ref domain tcsite oSite, ref domain tcitem oItem, ref domain tcmcs.s999m oExceptionMessage mb, </pre>

	ref	long	oExceptionID)
Usage:	Expl:	This function starts the overview session Site - Item Inventory (whwmd4504m100).	
	Input:	iStartMode	Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously.
		iStartFilter	Specifies the start filter that is to be applied to the started session. Possible values are: "bySite": data is displayed by Site session will be started on index 1 view field: Site "byItem": data is displayed by Item session will be started on index 2 view fields: Item
		iSessionIndex	Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a iStartFilter the session index will match the value of this variable.
		iQueryExtend	A specific query to be used when zooming to this session.
		iSite	Mandatory when iStartFilter "bySite" is used or iSessionIndex = 1 and iStartMode is MODELESS
		iItem	Mandatory when iStartFilter "byItem" is used or iSessionIndex = 2 and iStartMode is MODELESS
	Output:	for iStartMode MODAL:	
		oSite	- Site of selected inventory.
		oItem	- Item of selected inventory.
		oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
		oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
	Return:	0	- Session started
		<> 0	- Error

Public Interfaces for WarehouseItemInventory

The following functions are available:

[WarehouseItemInventory.StartDetail](#)

[WarehouseItemInventory.StartMultiMain](#)

[WarehouseItemInventory.StartOverview](#)

WarehouseItemInventory.StartDetail

DLL:	whextwmdapi
Syntax:	<pre> long WarehouseItemInventory.StartDetail(long iStartMode, domain tccwar iWarehouse, domain tcitem iItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the detail session Warehouse - Item Inventory (whwmd2515s000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iWarehouse iItem Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

WarehouseItemInventory.StartMultiMain

DLL:	whextwmdapi
Syntax:	<pre> long WarehouseItemInventory.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tccwar iWarehouse, domain tcitem iItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the detail session Warehouse - Item Inventory (whwmd2515s000). Input: <i>iStartMode</i> Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. <i>iStartFilter</i> Specifies the start filter that is to be applied to the started session. Possible values are: ""; <i>iQueryExtend</i> <i>iWarehouse</i> <i>iItem</i> Output: <i>oExceptionMessage</i> - The last message if the return value is not equal to 0. If more than one message is given, these are present in the <i>oExceptionID</i> <i>oExceptionID</i> - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

WarehouseItemInventory.StartOverview

DLL:	whextwmdapi
Syntax:	<pre> long WarehouseItemInventory.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccwar iWarehouse, domain tcitem iItem, </pre>

	<pre> ref domain tccwar oWarehouse, ref domain tcitem oItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the overview session Warehouse - Item Inventory (whwmd2515m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byWarehouse": data is displayed by Warehouse session will be started on index 1 view field: Warehouse "byItem": data is displayed by Item session will be started on index 2 view fields: Item iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a iStartFilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iWarehouse Mandatory when iStartFilter "byWarehouse" is used or iSessionIndex = 1 and iStartMode is MODELESS. iItem Mandatory when iStartFilter "byItem" is used or iSessionIndex = 2 and iStartMode is MODELESS. Output: for iStartMode MODAL: oWarehouse - Warehouse of selected inventory. oItem - Item of selected inventory. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

Public Interfaces for WarehouseReceipt

The following functions are available:

[WarehouseReceipt.PrintReceipts](#)

[WarehouseReceipt.StartDetail](#)

[WarehouseReceipt.StartMultiMain](#)

[WarehouseReceipt.StartOverview](#)

[WarehouseReceipts.StartWorkbench](#)

WarehouseReceipt.PrintReceipts

DLL:	whextinhapi This function is available from KB2327745 .						
Syntax:	<pre> long WarehouseReceipt.PrintReceipts(domain whinh.shpm iReceipt, domain tcmcs.str15 iDevice, ref long iProcessingOptionSet, ref boolean oDataProcessed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>						
Usage:	Expl: This public interface will print the receipt(s) for the iReceipt using the defaults or options as provided in the iProcessingOptionSet. Opening/Closing of the report is handled by this function. Pre: N.a. Post: N.a. Input: <table border="0"> <tr> <td>iReceipt</td><td>Optional</td></tr> <tr> <td>iDevice</td><td>Mandatory</td></tr> <tr> <td>iProcessingOptionSet</td><td>Optional, if 0, the default options are applied.</td></tr> </table> A Processing Option Set can be created via a call to function <code>ProcessingOptionSet.Create()</code>. In case iReceipt is filled then the following options of the iProcessingOptionSet will be ignored: - selection range fields (From/To) - ReceiptArray The Receipt will then be printed for the given iReceipt. In case option ReceiptArray is set then the selection range fields (From/To) of the iProcessingOptionSet will be ignored. The Receipt will then be printed for the receipts in the array. Processing Options have a direct relationship with the form fields on session Print Receipts (whinh3412m000) and are not explained in	iReceipt	Optional	iDevice	Mandatory	iProcessingOptionSet	Optional, if 0, the default options are applied.
iReceipt	Optional						
iDevice	Mandatory						
iProcessingOptionSet	Optional, if 0, the default options are applied.						

further detail here.

Please refer to the session help for additional information.

Processing Options that are set while a required Implemented Software Component is not available are ignored.

Explanation about setting of defaults:

Minimum Value: Minimum value of domain is taken as default value.

Maximum Value: Maximum value of domain is taken as default value.

NAME	TYPE	DEFAULT
ReceiptFrom	domain whinh.shpm	Minimum Value
ReceiptTo	domain whinh.shpm	Maximum Value
ReceiptLineFrom	domain tcpono	Minimum Value
ReceiptLineTo	domain tcpono	Maximum Value
ReceiptDateFrom	domain tcdate	Minimum Value
ReceiptDateTo	domain tcdate	Current Date/Time
ShipFromTypeFrom	domain tctyps	Minimum Value
ShipFromTypeTo	domain tctyps	Maximum Value
ShipFromCodeFrom	domain tccshp	Minimum Value
ShipFromCodeTo	domain tccshp	Maximum Value
ShipToTypeFrom	domain tctyps	Minimum Value
ShipToTypeTo	domain tctyps	Maximum Value
ShipToCodeFrom	domain tccshp	Minimum Value
ShipToCodeTo	domain tccshp	Maximum Value
EffectivityUnitFrom	domain tcuef.effn	Minimum Value
EffectivityUnitTo	domain tcuef.effn	Maximum Value
OwnerFrom	domain tccom.bpid	Minimum Value
OwnerTo	domain tccom.bpid	Maximum Value
ItemFrom	domain tcitem	Minimum Value
ItemTo	domain tcitem	Maximum Value
OrderOriginFrom	domain whinh.oorg	Minimum Value
OrderOriginTo	domain whinh.oorg	Maximum Value
OrderNumberFrom	domain tcorno	Minimum Value
OrderNumberTo	domain tcorno	Maximum Value
OrderSetFrom	domain tcwset	Minimum Value
OrderSetTo	domain tcwset	Maximum Value
OrderLineFrom	domain tcpono	Minimum Value
OrderLineTo	domain tcpono	Maximum Value
PrintReceiptLines	domain whinh.prec	whinh.prec.all
PrintComponents	domain tcyesno	tcyesno.yes
PrintStockPointDetails	domain tcyesno	tcyesno.yes
PrintOwnership	domain tcyesno	tcyesno.no
PrintPegDistribution	domain tcyesno	tcyesno.no
PrintLandedCosts	domain tcyesno	tcyesno.no
PrintReceiptLineText	domain tcyesno	tcyesno.yes
PrintInspectionText	domain tcyesno	tcyesno.no
ReportNumber	domain tcmcs.long	1
ReceiptArray	domain ttjson	0
ReportName	domain tcmcs.str16	Empty String

Possible values of ReportNumber are:

- 1 - Receipts
- 2 - Receipt Lines by Order
- 3 - Receipt Lines by Item
- 4 - Receipt Lines by Receipt Number
- 5 - Receipt Lines by Business Partner

JSON Object ReceiptArray has the following structure:

```
"ReceiptArray": [
  {
    "Receipt": "REC000252"
  },
  ...
]
```


	Primary Key Field: iReceipt Output: oExceptionMessage oExceptionID Return: 0 <> 0	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. - Session started - Error

WarehouseReceipt.StartMultiMain

DLL:	whextinhapi This function is available from KB2301526 .	
Syntax:	<pre>long WarehouseReceipt.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain whinh.shpm iReceipt, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: This function starts the multi main table session Warehouse Receipts (whinh3512m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend A specific query to be used when zooming to this session. Primary Key Fields: iReceipt Output: oExceptionMessage oExceptionID Return: 0 - iReceipt must refer to an existing warehouse receipt. - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. - Session started	

	<> 0	- Error

WarehouseReceipt.StartOverview

DLL:	whextinhapi
	This function is available from KB2301526 .
Syntax:	<pre> long WarehouseReceipt.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain whinh.shpm iReceipt, ref domain whinh.shpm oReceipt, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the overview session Warehouse Receipts Overview (whinh3110m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend A specific query to be used when zooming to this session. iReceipt Optional Output: for iStartMode MODAL: oReceipt oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

WarehouseReceipts.StartWorkbench

DLL:	whextinhapi
Syntax:	<pre> long WarehouseReceipts.StartWorkbench(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), const string iInboundLineFilter(), domain tccwar iWarehouse, domain tccom.bpid iBusinessPartner, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the workbench session Warehouse Receipts (whinh2610m100). Input: <code>iStartMode</code> Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. <code>iStartFilter</code> Not Used. <code>iSessionIndex</code> Not Used. <code>iQueryExtend</code> Not Used. <code>iInboundLineFilter</code> A specific query to be used when zooming to the Inbound Lines session. Following input variables form the filtering fields, these fields are not mandatory. <code>iWarehouse</code> <code>iBusinessPartner</code> Output: <code>oExceptionMessage</code> - The last message if the return value is not equal to 0. If more than one message is given, these are present in the <code>oExceptionID</code> <code>oExceptionID</code> - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

Public Interfaces for Warehousing

The following functions are available:

[Warehousing.ConvertKindOfOrderToOrderOrigin](#)

[Warehousing.ConvertOrderOriginToKindOfOrder](#)

[Warehousing.GetSettings](#)

[Warehousing.StartAutomaticInboundProcessing](#)

[Warehousing.StartAutomaticOutboundProcessing](#)

Warehousing.ConvertKindOfOrderToOrderOrigin

DLL:	whextinhapi This function is available from KB2308457 .
Syntax:	<pre> long Warehousing.ConvertKindOfOrderToOrderOrigin(domain tckoor iKindOfOrder, ref domain whinh.oorg oOrderOrigin, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function converts the domain whinh.oorg to tckoor. Pre: N.a. Post: N.a. Input: oKindOfOrder - Kind of Order (Mandatory) Output: iOrderOrigin - Order Origin oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - No Error <> 0 - Error

Warehousing.ConvertOrderOriginToKindOfOrder

DLL:	whextinhapi This function is available from KB2308457 .
Syntax:	<pre> long Warehousing.ConvertOrderOriginToKindOfOrder(domain whinh.oorg iOrderOrigin, ref domain tckoor oKindOfOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function converts the domain whinh.oorg to tckoor. Pre: N.a.

	Post: N.a. Input: iOrderOrigin - Order Origin (Mandatory) Output: oKindOfOrder - Kind of Order oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - No Error <> 0 - Error

Warehousing.GetSettings

DLL:	whextwmdapi001
Syntax:	<pre> long Warehousing.GetSettings(domain tcncmp iLogisticCompany, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves warehousing settings. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter tables whina000, whinh000, whinp000, whinr000, whltc000 or whwmd000), or site level (settings table whwmd201) depending on the specified input field(s) (mnemonic), input argument Site and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per site) are also retrieved. 2. Fields that are available at table whwmd201 and at the warehouse table may not be read using this Public Interface, but should be read from the warehouse table. The warehouse table contains the most detailed value. 3. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when warehouse order series, step size for warehouse order lines and the first element of slow-moving percentage are needed, the function must be called as follows: <pre> if Warehousing.GetSettings(* Fixed arguments: company, --> input site, --> input force.read, --> input context.is.master.data, --> input exception.message, --> output </pre>

	<pre> exception.id, --> output /* Variable arguments: "wosr", --> input series.for.warehousing.orders, --> output "woss", --> input step.size.for.order.lines, --> output "slmp(1)", --> input slow.moving.percentage) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif 4. For field "psrm" (Packing Slip Mandatory on Receipts for), an additional argument is needed: the order origin. The output has domain tcyesno. The function call will look like this: if Warehousing.GetSettings(/* Fixed arguments: company, --> input site, --> input force.read, --> input context.is.master.data, --> input exception.message, --> output exception.id, --> output /* Variable arguments: "psrm" --> input order.origin, --> input output) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif 5. The fields "qiap" (Quarantine Inventory Available for Planning) and "qipo" (Quarantine Inventory Available for Planning by Origin) are related. Both fields will return the same value. If qiap is 'No', 'Yes' or 'By Due Date', then this value will be returned. if qiap is 'By Order Origin', the value of "qipo" for the specified order origin is returned. ! So for these fields, an additional argument 'order origin' is needed. This argument is described in note 4. The domain of the return value is whinh.qipo. Pre: None Post: None Input: iLogisticCompany - Logistic Company: Mandatory iSite - Site: Not Mandatory iForceRead - For future use iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. If the field mnemonic is 'psrm', the order origin must be passed as well. </pre>

	Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - Error.

Warehousing.StartAutomaticInboundProcessing

DLL:	whextinhapi
	This function is available from KB3541337 .
Syntax:	<pre>long Warehousing.StartAutomaticInboundProcessing(ref boolean oNotificationsFound, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface will start the processing of the next automatic inbound activity for the notified inbound object. This public interface will be able to automatically perform the next automatic steps. This public interface will stop the automatic process when the next activity is not set to automatic. This function must not be called within a logical transaction as this function will have it's own transaction handling. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: N.a. Output: oNotificationsFound - Inbound Notifications to be processed are found. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Warehousing.StartAutomaticOutboundProcessing

DLL:	whextinhapi						
	This function is available from KB3541337 .						
Syntax:	<pre>long Warehousing.StartAutomaticOutboundProcessing(ref boolean oNotificationsFound, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>						
Usage:	Expl: This public interface will start the processing of the next automatic outbound activity for the notified outbound object. This public interface will be able to automatically perform the next automatic steps. This public interface will stop the automatic process when the next activity is not set to automatic. This function must not be called within a logical transaction as this function will have it's own transaction handling. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: N.a. Output: <table border="0"> <tr> <td>oNotificationsFound</td><td>- Outbound Notifications to be processed are found.</td></tr> <tr> <td>oExceptionMessage</td><td>- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.</td></tr> <tr> <td>oExceptionID</td><td>- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.</td></tr> </table> Return: 0: OK, <> 0: Error	oNotificationsFound	- Outbound Notifications to be processed are found.	oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.	oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.
oNotificationsFound	- Outbound Notifications to be processed are found.						
oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.						
oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.						

Public Interfaces for StockPoint

The following functions are available:

[StockPoint.CheckBlocking](#)

[StockPoint.Move](#)

[StockPoint.PackHandlingUnits](#)

StockPoint.CheckBlocking

DLL:	whextinrapi		
Syntax:	<pre>long StockPoint.CheckBlocking(domain tccwar iWarehouse, domain whloca iLocation, domain tcitem iItem, domain whhuid iHandlingUnit, domain tcclot iLot, domain tcibd.sern iSerial, domain tctrns.date iInventoryDate, domain whwmd.pkdf iPackageDefinition, domain tccuni iStorageUnit, domain tcqstl iQuantityInStorageUnit, domain tcqivl iQuantityInInventoryUnit, boolean iInterWarehouseTransfer, boolean iIntraWarehouseTransfer, boolean iIssueFromQuarantine, boolean iProcessingPickOutboundAdvice, boolean iProcessingShipment, boolean iCheckManualBlock, ref boolean oStockPointBlocked, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>		
Usage:	Expl: This function detects whether inventory is blocked which prevents the inventory from being used. Manual Blockings are never checked for serials and only evaluated in case one of the following variables is true: - iProcessingPickOutboundAdvice - iProcessingShipment - iCheckManualBlock Input: iWarehouse - Warehouse - Mandatory iLocation - Location - Mandatory when warehouse and item are location controlled. iItem - Item - Mandatory when iHandlingUnit is empty. iHandlingUnit - Handling Unit - Mandatory when iItem is empty. iLot - Lot (in inventory) - Mandatory when for the item the lots are registered in inventory and iHandlingUnit is empty. iSerial - Serial (in inventory) - Mandatory when for the item, the serials are registered in inventory and iHandlingUnit is empty. iInventoryDate - Inventory Date - Mandatory when the item has outbound method FIFO or LIFO and iHandlingUnit is empty. iPackageDefinition - Package Definition - Not Mandatory iStorageUnit - Storage Unit - Mandatory when iSerial is empty. iQuantityInStorageUnit - Quantity in Storage Unit - Mandatory when iSerial is empty. iQuantityInInventoryUnit - Quantity in Inventory Unit - Mandatory when Handling Unit is empty. iInterWarehouseTransfer - Intra Warehouse Transfer - Mandatory Movement between warehouses.		

	Value should be true or false.
iIntraWarehouseTransfer	- Inter Warehouse Transfer - Mandatory Movement within a warehouse between locations.
	Value should be true or false.
iIssueFromQuarantine	- Issue from Quarantine Inventory
	- Mandatory
	Value should be true or false.
iProcessingPickOutboundAdvice	- Processing step is Pick Outbound Advice - Mandatory
	Value should be true or false.
iProcessingShipment	- Processing step is Shipping
	- Mandatory
	Overrules processing step Pick Outbound Advice when both are true.
	Value should be true or false.
iCheckManualBlock	- Check on Manual Block is performed.
	- Mandatory
	Value should be true or false.
Output: oStockPointBlocked	- Stock Point Blocked, true/false.
oExceptionMessage	- The last message if the return value is not equal to 0.
	If more than one message is given, these are present in the oExceptionID
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
Return: 0: OK, <> 0: Error	

StockPoint.Move

DLL:	whextinrapi
Syntax:	<pre> long StockPoint.Move(domain tccwar iWarehouse, domain tcitem iItem, domain whloca iOriginatingLocation, domain whhuid iHandlingUnit, domain tcclot iLot, domain tcinvdate iInventoryDate, domain whwmd.pkdf iPackageDefinition, domain tcyesno iFullStockPoint, domain tcqstl iQuantityToMove, domain tccuni iStorageUnit, ref domain tcibd.sern iSerialArray() fixed, boolean iMoveBlockedStockPoint, domain whloca iDestinationLocation, ref boolean oStockPointMoved, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function will move inventory from the originating stock point to the destination stock point.

Pre:	db.retry.point() must be set
Post:	abort/commit transaction
Input:	iWarehouse - Warehouse - Mandatory iItem - Item - Mandatory iOriginatingLocation - Originating Location - Mandatory iHandlingUnit - Handling Unit - Not Mandatory When the stock point is packed inside a handling unit, the handling unit should be passed. iLot - Lot (in inventory) Mandatory when iItem is lot controlled in inventory, and the iHandlingUnit is empty. If iHandlingUnit is filled the handling unit will determine the lot itself, as the lot is stored in the handling unit stock point details. iInventoryDate - Inventory Date Mandatory when iItem has outbound method FIFO or LIFO and the iHandlingUnit is empty. If iHandlingUnit is filled the handling unit will determine the inventory date itself, as the inventory date is stored in the handling unit stock point details. iPackageDefinition - Fixed Package Definition Mandatory when the stock point has inventory structure which uses a fixed package definition. Otherwise the inventory structure cannot be updated properly. Input will not be validated, if no inventory can be found the stock point cannot be moved. iFullStockPoint - Full Stock Point - Not Mandatory When set to tyesno.yes, the iQuantityToMove is not mandatory anymore, because the complete stock point must be moved to the iDestinationLocation. iQuantityToMove - Quantity to Move Mandatory when iHandlingUnit is empty and iFullStockPoint is set to tyesno.no. Should be greater than zero when iHandlingUnit is empty and iFullStockPoint is set to tyesno.no. iStorageUnit - Storage Unit Mandatory when iHandlingUnit is empty and iFullStockPoint is set to tyesno.no. iSerialArray - Serial Array Can be passed to indicate which serials must be moved, if empty the logic will pick serials from the stock point which is to be moved. If iHandlingUnit is passed as empty the memory should be allocated for the array, if no inventory allocation is done, an API error will be given. iMoveBlockedStockPoint - Move Blocked Inventory - Mandatory Value should be true or false.

	Output: iDestinationLocation - Destination Location - Mandatory oStockPointMoved - Stock Point Moved, true/false. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

StockPoint.PackHandlingUnits

DLL:	whextinrapi
Syntax:	<pre>long StockPoint.PackHandlingUnits(domain tccwar iWarehouse, domain whloca iLocation, domain tcitem iItem, domain tcclot iLot, domain tcibd.sern iSerial, domain tcinv.date iInventoryDate, domain tccuni iStorageUnit, domain tcqstl iQuantityInStorageUnit, domain whwmd.pkdf iPackageDefinition, domain whhuid iHandlingUnit, domain tcuef.mask iMask, domain tcyesno iSSCC, domain whinh.oorg iOrderOrigin, domain tcorno iOrder, domain tcwset iOrderSet, ref domain tcmcs.long oNumberOfHandlingUnits, ref domain whhuid oHandlingUnitArray() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface will pack the anonymous stock in one or more handling units. This public interface can be used only when for the iItem and iWarehouse combination handling units are used in stock. Pre: db.retry.point() Post: commit/abort transaction Input: iWarehouse - Warehouse - Mandatory iLocation - Location - Mandatory when iWarehouse and iItem are location controlled. iItem - Item - Mandatory. iLot - Lot (in inventory) - Mandatory when for the iItem the lots are registered in inventory. iSerial - Serial (in inventory) - Mandatory when the iItem is serialized in inventory iInventoryDate - Inventory Date - Mandatory when the iItem has outbound method FIFO or LIFO. iStorageUnit - Storage Unit - Mandatory when iSerial

	is empty. iQuantityInStorageUnit - Quantity in Storage Unit - Mandatory when iSerial is empty. iPackageDefinition - Package Definition - Not Mandatory Package definition must be of type Variable. iHandlingUnit - Handling Unit with this Id will be generated - Not Mandatory iMask - Handling Unit Mask - Not Mandatory if empty default mask will be used. iSSCC - iMask is with SSCC standard. iOrderOrigin - Order Origin - can be used when iMask contains a segment with this field - Not Mandatory. iOrder - Order Number - can be used when iMask contains a segment with this field - Not Mandatory. iOrderSet - Order Set - can be used when iMask contains a segment with this field - Not Mandatory. Output: oNumberHandlingUnits - Number of handling units in array. oHandlingUnitArray - Array with generated handling units. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Public Interfaces for StockPointInventory

The following functions are available:

[StockPointInventory.StartDetail](#)

[StockPointInventory.StartOverview](#)

StockPointInventory.StartDetail

DLL:	whextinrapi
	This function is available from KB2297960 .

Syntax:	<pre> long StockPointInventory.StartDetail(long iStartMode, domain tccwar iWarehouse, domain whloca iLocation, domain tcitem iItem, domain tcclot iLot, domain tctrns.date iInventoryDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the detail session Stock Point Inventory (whinr1540m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variable form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iWarehouse iLocation iItem iLot iInventoryDate Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

StockPointInventory.StartOverview

DLL:	whextinrapi This function is available from KB2297960 .
Syntax:	<pre> long StockPointInventory.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccwar iWarehouse, domain whloca iLocation, domain tcitem iItem, </pre>

	<pre> domain tcclot iLot, domain tctrns.date iInventoryDate, ref domain tccwar oWarehouse, ref domain whloca oLocation, ref domain tcitem oItem, ref domain tcclot oLot, ref domain tctrns.date oInventoryDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl This function starts the overview session Stock Point Inventory (whinr1540m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byItemWarehouse": data is displayed by Item and Warehouse session will be started on index 1 view fields: Item, Warehouse "byWarehouse": data is displayed by Warehouse session will be started on index 2 view field: Warehouse "byWarehouseItem": data is displayed by Warehouse and Item session will be started on index 3 view fields: Warehouse, Item "byWarehouseLocation": data is displayed by Warehouse, Location and Item session will be started on index 4 view fields: Warehouse, Location, Item "byItemLot": data is displayed by Item and Lot session will be started on index 5 view fields: Item, Lot iSessionIndex Specifies the table index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iWarehouse Mandatory if iStartMode = MODELESS and i.session.index = 1,2,3 or 4 (or i.start.filter = "byItemWarehouse" or i.start.filter = "byWarehouse" or i.start.filter = "byWarehouseItem" or i.start.filter = "byWarehouseLocation") iLocation Optional iItem </pre>

	Mandatory if iStartMode = MODELESS and i.session.index = 1,3,4 or 5 (or i.start.filter = "byItemWarehouse" or i.start.filter = "byWarehouseItem" or i.start.filter = "byWarehouseLocation" or i.start.filter = "byItemLot") iLot Optional iInventoryDate Optional Output: for iStartMode MODAL: oWarehouse - Warehouse of selected record oLocation - Location of selected record oItem - Item of selected record oLot - Lot of selected record oInventoryDate - Inventory Date of selected record oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

Public Interfaces for Warehouse

The following functions are available:

[Warehouse.StartDetail](#)

[Warehouse.StartOverview](#)

Warehouse.StartDetail

DLL:	whextwmdapi
Syntax:	<pre>long Warehouse.StartDetail(long iStartMode, domain tccwar iWarehouse, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the detail session Warehouses (whwmd2500m000). Input: iStartMode

	Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Field: iWarehouse Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

Warehouse.StartOverview

DLL:	whextwmdapi
Syntax:	<pre>long Warehouse.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccwar iWarehouse, ref domain tccwar oWarehouse, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the overview session Warehouses (whwmd2500m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byWarehouse":

	data is displayed by Warehouse session will be started on index 1 view field: Warehouse iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a iStartFilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iWarehouse Mandatory when iStartFilter "byWarehouse" is used or iSessionIndex = 1 and iStartMode is MODELESS. Output: for iStartMode MODAL: oWarehouse - Warehouse of selected inventory. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

Public Interfaces for WarehouseLocation

The following functions are available:

[WarehouseLocation.StartDetail](#)

[WarehouseLocation.StartOverview](#)

[WarehouseLocation.StartRecalculateOccupation](#)

WarehouseLocation.StartDetail

DLL:	whextwmdapi
	This function is available from KB3502697 .
Syntax:	<pre>long WarehouseLocation.StartDetail(long iStartMode, domain tccwar iWarehouse, domain whloca iLocation,</pre>

	ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function starts the detail session Warehouse Location (whwmd3100s000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Field: iWarehouse iLocation Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

WarehouseLocation.StartOverview

DLL:	whextwmdapi
	This function is available from KB3502697 .
Syntax:	<pre>long WarehouseLocation.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccwar iWarehouse, domain whloca iLocation, ref domain tccwar oWarehouse, ref domain whloca oLocation, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the overview session Warehouse Locations (whwmd3500m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be

	started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byWarehouseLocation": data is displayed by Warehouse and Location session will be started on index 1 view field: Warehouse "byWarehouseFixedLocation": data is displayed by Warehouse and Fixed Location session will be started on index 2 view fields: Warehouse, Fixed Location "byWarehouseLocationOccupied": data is displayed by Warehouse and Location Occupied session will be started on index 3 view fields: Warehouse, Location Occupied "byWarehouseZone": data is displayed by Warehouse and Zone session will be started on index 4 view fields: Warehouse, Zone "byWarehouseSearchArgument": data is displayed by Warehouse and Search Argument session will be started on index 5 view field: Warehouse "byWarehouseOwnerOfInventory": data is displayed by Warehouse and Owner session will be started on index 6 view fields: Warehouse, Owner "byWarehouseLocationType": data is displayed by Warehouse and Location Type session will be started on index 7 view field: Warehouse iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a iStartFilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iWarehouse Mandatory when iStartFilter is used or iStartMode = MODELESS and started in MULTI_OCC mode. iLocation Optional Output: for iStartMode MODAL: oWarehouse - Selected Warehouse. oLocation - Selected Location. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

WarehouseLocation.StartRecalculateOccupation

DLL:	whextwmdapi
	This function is available from KB3502697 .
Syntax:	<pre> long WarehouseLocation.StartRecalculateOccupation(long iStartMode, boolean iIgnoreSelectionFields, domain tccwar iFromWarehouse, domain whloca iFromLocation, domain tccwar iToWarehouse, domain whloca iToLocation, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl This function starts the session Recalculate Occupation of Location (whwmd3201m000). Depending on the main table of the calling session, Non-Consecutive Record Selection (NCRS) is used. When the main table is: Locations (whwmd300) or Location Capacity (whwmd301) then the range selections will be imported based on the selected records in the parent session and the selection related fields will be ignored. Input: iStartMode Not Used. iIgnoreSelectionFields If true, From/To selection fields will not be filled in the session. iFromWarehouse From Warehouse selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iFromLocation From Location selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToWarehouse To Warehouse selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToLocation To Location selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

Public Interfaces for CompanyItemInventory

The following functions are available:

[CompanyItemInventory.StartDetail](#)

[CompanyItemInventory.StartOverview](#)

CompanyItemInventory.StartDetail

DLL:	whextwmdapi	
Syntax:	<pre> long CompanyItemInventory.StartDetail(long iStartMode, domain tcitem iItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>	
Usage:	Expl: This function starts the detail session Company - Item Inventory (whwmd4100m100). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iItem Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error	

CompanyItemInventory.StartOverview

DLL:	whextwmdapi
Syntax:	<pre> long CompanyItemInventory.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, ref domain tcitem oItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the overview session Company - Item Inventory (whwmd4100m100). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byItem": data is displayed by Item session will be started on index 1 view fields: None iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a iStartFilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iItem Mandatory when iStartFilter "byItem" is used or iSessionIndex = 1 and iStartMode is MODELESS. Output: for iStartMode MODAL: oItem - Item of selected inventory. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

Public Interfaces for ShipmentNotice

The following functions are available:

[ShipmentNotice.ReceiveDirectDelivery](#)

[ShipmentNotice.StartDetail](#)

[ShipmentNotice.StartMultiMain](#)

[ShipmentNotice.StartOverview](#)

ShipmentNotice.ReceiveDirectDelivery

DLL:	whextinhapi
Syntax:	<pre> long ShipmentNotice.ReceiveDirectDelivery(domain tccom.bpid iShipFromBusinessPartner, domain whinh.shpm iShipment, ref long oNumberOfLinesReceived, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function executes option Receive Direct Delivery for the given Shipment Notice (ASN). This is allowed if: - The ASN header exists with status Scheduled or Scheduled (Manually), and - If Header status is Scheduled, at least one ASN Line(s) exist with approval status unequal to Approved - The security settings by Warehouse allows usage, and - ASN Line(s) having Direct Delivery Line(s) exist Be aware that transaction management is handled within this function. This function informs Order Management Purchase of any receipts that are made in Warehousing. As soon as this notification is given, Purchase can continue with any automatic processing of the order/schedule line that is required. Pre: n.a. Post: n.a. Input: iShipFromBusinessPartner- Ship-from Business Partner (mandatory) iShipment - Shipment (mandatory) Output: oNumberOfLinesReceived - Number of received Shipment Notice Lines, having a direct delivery line. oExceptionMessage - The last message if the return value is not equal to 0.

	If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

ShipmentNotice.StartDetail

DLL:	whextinhapi
Syntax:	<pre> long ShipmentNotice.StartDetail(long iStartMode, domain tccom.bpid iShipFromBusinessPartner, domain whinh.shpm iShipment, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the detail session Shipment Notices (whinh3100m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iShipFromBusinessPartner iShipment Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

ShipmentNotice.StartMultiMain

DLL:	whextinhapi
Syntax:	<pre> long ShipmentNotice.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccom.bpid iShipFromBusinessPartner, domain whinh.shpm iShipment, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the detail session Shipment Notices (whinh3600m000). Input: <code>iStartMode</code> Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. <code>iStartFilter</code> Not Used. <code>iSessionIndex</code> Specifies the session index that is to be used. <code>iQueryExtend</code> <code>iShipFromBusinessPartner</code> <code>iShipment</code> Output: <code>oExceptionMessage</code> - The last message if the return value is not equal to 0. If more than one message is given, these are present in the <code>oExceptionID</code> <code>oExceptionID</code> - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

ShipmentNotice.StartOverview

DLL:	whextinhapi
Syntax:	<pre> long ShipmentNotice.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccom.bpid iShipFromBusinessPartner, domain whinh.shpm iShipment, </pre>

	<pre> domain whinh.stat iStatus, domain whinh.load iFreightLoad, domain tcaref iSuppliersASN mb, ref domain tccom.bpid oShipFromBusinessPartner, ref domain whinh.shpm oShipment, ref domain tcmcs.s999m oExceptionMessage mb, ref domain long oExceptionID) </pre>
Usage:	<pre> Expl: This function starts the overview session Shipment Notices (whinh3100m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byShipFromBusinessPartnerShipment": session will be started on index 1 view field: None "byStatus": session will be started on index 2 view field: Status "byFreightLoad": session will be started on index 3 view field: Freight Load "byShipFromBusinessPartnerSuppliersASN": session will be started on index 4 view fields: ShipFromBusinessPartner SuppliersASN iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iShipFromBusinessPartner Mandatory when iStartFilter "byShipFromBusinessPartnerShipment" or "byShipFromBusinessPartnerSuppliersASN" is used and the session is started in overview mode with start mode MODELESS iShipment Mandatory when iStartFilter "byShipFromBusinessPartnerShipment" is used and the session is started in overview mode with start mode MODELESS iStatus Mandatory when iStartFilter "byStatus" is used and the session is started in overview mode with start mode MODELESS iFreightLoad iSuppliersASN Output: for iStartMode MODAL: oShipFromBusinessPartner - Ship-from Business Partner of </pre>

	oShipment oExceptionMessage oExceptionID Return: 0 <> 0	selected ASN. - Shipment of selected ASN. - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Session started - Error

Public Interfaces for ShipmentNoticeLine

The following functions are available:

[ShipmentNoticeLine.Receive](#)

ShipmentNoticeLine.Receive

DLL:	whextinhapi	
Syntax:	<pre>long ShipmentNoticeLine.Receive(domain tccom.bpid iShipFromBusinessPartner, domain whinh.shpm iShipmentNotice, domain tcpono iShipmentNoticeLine, domain whinh.shpm iReceipt, domain tcqstl iReceivedQuantity, domain tccuni iReceivedUnit, domain tcpksp iPackingSlip mb, domain whinh.load iLoad, domain whinh.shpm iShipment, ref domain whinh.shpm oReceipt, ref long oNumberOfReceiptLines, ref domain tcpono oReceiptLineArray(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: This function executes option Receive for the given Shipment Notice Line. Pre: db.retry.point must be set oReceiptLineArray must be declared as a based variable, this function will allocate the memory. Post: Commit the transaction in case of success Abort the transaction in case of failure After the oReceiptLineArray is used, free the memory.	

	Input: iShipFromBusinessPartner- iShipmentNotice iShipmentNoticeLine iReceipt - Ship-from Business Partner - Mandatory - Shipment Notice - Mandatory - Shipment Notice Line - Mandatory - Receipt header on which lines must be added. Optional, if empty a new receipt header will be created. iReceivedQuantity iReceivedUnit - Received Quantity - Optional Value should not be less than zero. - Received Unit - Mandatory if received quantity is greater than zero). iPackingSlip iLoad iShipment - Packing Slip - Optional - Load - Optional - Shipment - Optional Output: oReceipt - Receipt that is created (or populated with the iReceipt when iReceipt is filled) oNumberOfReceiptLines - Number of receipt lines that are created. oReceiptLineArray oExceptionMessage - Receipt lines that are created. - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Public Interfaces for WarehouseOrder

The following functions are available:

[WarehouseOrder.CalculateDeliveryDate](#)

[WarehouseOrder.CreateReturn](#)

[WarehouseOrder.PrintLabelForActivity](#)

[WarehouseOrder.StartDetail](#)

[WarehouseOrder.StartMultiMain](#)

[WarehouseOrder.StartOverview](#)

WarehouseOrder.CalculateDeliveryDate

DLL:	whextinhapi This function is available from KB2292294 .
Syntax:	<pre> long WarehouseOrder.CalculateDeliveryDate(domain whinh.oorg iOrderOrigin, domain whinh.ittp iTransactionType, domain tctyps iShipFromType, domain tccshp iShipFromCode, domain tcccom.cadr iShipFromAddress, domain tcncmp iShipFromCompany, domain tcccom.cadr iShipToAddress, domain tcitem iItem, domain tccfrw iCarrier, domain tccrte iRoute, domain tcmcs.serv iServiceLevel, domain tcccom.trmd iTransportCategory, domain whinh.cmtg iTransportMeansGroup, domain tctmcb iTransportMeansCombination, domain tcdsca iMeansOfTransport mb, domain tccdec iDeliveryTerms, domain tcptpa iPointOfTitlePassage, domain tcdate iPlannedReceiptDate, ref domain tcdate oPlannedReceiptDate, ref domain tcdate oPlannedDeliveryDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function calculates the planned delivery date based on the planned receipt date. 1. If the planned receipt date is not set (unknown) it will be the current date. 2. The time needed for transportation is calculated backwards from the planned receipt date. Except for the origins production and production manual. Per definition the time between planned delivery and planned receipt for production is the routing time and not the transportation time. This function does not do a routing calculation. 3. A calendar correction is done on the ship from site. Pre: N.a. Post: N.a. Input: - iOrderOrigin - Order Origin; Mandatory - iTransactionType - Transaction Type; Mandatory - iShipFromType - Ship From Type; Mandatory - iShipFromCode - Ship From Code; Mandatory - iShipFromAddress - Ship From Address; Mandatory - iShipFromCompany - Ship From Company; Mandatory - iShipToAddress - Ship To Address; Mandatory - iItem - Item; Not Mandatory - iCarrier - Carrier; Not Mandatory - iRoute - Route; Not Mandatory - iServiceLevel - Service Level; Not Mandatory - iTransportCategory - Transport Category; Not Mandatory - iTransportMeansGroup - Transport Means Group; Not Mandatory - iTransportMeansCombination - Transport Mean Combination; Not Mandatory - iMeansOfTransport - Means of Transport; Not Mandatory - iDeliveryTerms - Terms of Delivery; Not Mandatory - iPointOfTitlePassage - Point of title Passage; Not Mandatory - iPlannedReceiptDate - Planned Receipt Date; Not Mandatory

	Output: oPlannedReceiptDate - Planned Receipt Date oPlannedDeliveryDate - Planned Delivery Date oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Planned Dates calculated successfully DALHOOKERROR - An error occurred when calculating Planned Dates

WarehouseOrder.CreateReturn

DLL:	whextinhapi
Syntax:	<pre> long WarehouseOrder.CreateReturn(domain whinh.oorg iSourceOrderOrigin, domain tcorno iSourceOrder, domain tcwset iSourceOrderSet, domain tcorno iTargetOrder, domain tcseri iTargetSeries, domain tccotp iOrderType, domain tcdate iDeliveryDate, domain tccwoc iOffice, domain tcyesno iCopyText, domain whqty.based.on iBasedOnQuantity, domain tcyesno iAllOrderLines, long iNumberOfReceiptLines, ref domain whinh.shpm iReceiptArray() fixed, ref domain tcpono iReceiptLineArray(), long iNumberOfShipmentLines, ref domain whinh.shpm iShipmentArray() fixed, ref domain tcpono iShipmentLineArray(), long iNumberOfOrderLines, ref domain tcpono iOrderLineArray(), ref domain tcpono iOrderSequenceArray(), ref domain whinh.oorg oTargetOrderOrigin, ref domain tcorno oTargetOrder, ref domain tcwset oTargetOrderSet, ref domain tcpono oCreatedNumberOfOrderLines, ref domain tcmcs.s999m oExceptionMessage mb, ref domain long oExceptionID) </pre>
Usage:	Expl: This function creates a return order for a specified warehouse order header. Be aware that transaction management is handled within this function. Pre: N.a. Post: N.a. Input: iSourceOrderOrigin The Order origin to be copied from. This is mandatory to fill. iSourceOrder Order number to be copied from. This is mandatory to fill. iSourceOrderSet Order set to be copied from. This is mandatory to fill.

	iTargetOrder iTargetSeries There are two options. Specify the order number (iTargetOrder) to copy to or the series (iTargetSeries) to be used to create an order number. iOrderType The order type for the created order. Not mandatory to fill. iDeliveryDate The delivery date for the created order. Mandatory to fill. iOffice The office for the created order. Not mandatory to fill. iCopyText Copy text indicator (Yes/No). Mandatory to fill. iBasedOnQuantity Specify which quantity should be used to create the return order. - whqty.based.on.received: Based on received quantity - whqty.based.on.shipped: Based on shipped quantity - whqty.based.on.ordered: Based on ordered quantity iAllOrderLines Return all lines indicator (Yes/No). Mandatory to fill. When not returning all lines then you have to specify the (receipt, shipment or order) lines to be copied. This needs to be done according your iBasedOnQuantity value. iNumberOfReceiptLines Specify the number of receipt lines. This is mandatory when iAllOrderLines is No and iBasedOnQuantity is received quantity. iReceiptArray Specify the receipts which need to be returned. This is mandatory when iAllOrderLines is No and iBasedOnQuantity is received quantity. iReceiptLineArray Specify the receipt lines which need to be returned. This is mandatory when iAllOrderLines is No and iBasedOnQuantity is received quantity. The receipt of receipt line should match with the receipt on the same position of the iReceiptArray. iNumberOfShipmentLines Specify the number of shipment lines. This is mandatory when iAllOrderLines is No and iBasedOnQuantity is shipped quantity. iShipmentArray Specify the shipments which need to be returned. This is mandatory when iAllOrderLines is No and iBasedOnQuantity is shipped quantity. iShipmentLineArray Specify the shipment lines which need to be returned. This is mandatory when iAllOrderLines is No and iBasedOnQuantity is shipped quantity. The shipment of shipment line should match with the shipment on the same position of the iShipmentArray. iNumberOfOrderLines Specify the number of order lines. This is mandatory when iAllOrderLines is No and iBasedOnQuantity is ordered quantity. iOrderLineArray Specify the order lines which need to be returned. This is mandatory when iAllOrderLines is No and

	iBasedOnQuantity is ordered quantity. iOrderSequenceArray Specify the order line sequences which need to be returned. This is mandatory when iAllOrderLines is No and iBasedOnQuantity is ordered quantity. The line number of the sequence should match with the line on the same position of the iOrderLineArray. Output: oTargetOrderOrigin Origin of the created order. oTargetOrder Order number of the created order. oTargetOrderSet Order set of the created order. oCreatedNumberOfOrderLines Number of order lines which are created. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Success - Return order has been created successfully. <> 0 - Error - Return order could not be created. r

WarehouseOrder.PrintLabelForActivity

DLL:	whextinhapi
Syntax:	<pre>long WarehouseOrder.PrintLabelForActivity(domain whinh.prtty iProcedureType, domain tcmcs.st20m iActivity mb, domain tccom.bpid iShipFromBusinessPartner, domain whinh.shpm iShipmentNotice, domain tcpono iShipmentNoticeLine, domain whinh.shpm iReceipt, domain tcpono iReceiptLine, domain tcpono iBomLine, domain tcorno iInboundAdvice, domain tcpono iAdviceLine, domain tcpono iBomLineSequence, domain tcorno iInspection, domain tcpono iInspectionSequence, domain whinh.oorg iOrderOrigin, domain tcorno iOrder, domain tcwset iOrderSet, domain tcpono iOrderLine, domain tcpono iOrderSequence, domain tcpono iOutboundAdvice, domain whinh.shpm iShipment, domain tcpono iShipmentLine, domain whinh.plbs iLabelPrinting, domain tclabl iLabelLayout, domain whwmd.lbpb iLabelPrintedBy, domain tcmcs.long iNumberOfCopies, domain tcyesno iHandlingUnitOnly,</pre>

	domain whinh.prmt iPrintingMethod, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This public interface supports printing of Labels for warehouse order activities. Pre: db.retry.point(). Post: abort/commit transaction. Input: iProcedureType - Procedure Type (Mandatory) Allowed Values: Inbound Procedure (whinh.prtty.receipt.line) Inspection Procedure (whinh.prtty.receipt.isnpect) Outbound Procedure (whinh.prtty.issue.advice) Shipment Procedure (whinh.prtty.issue.shipment) iActivity - Warehousing Activity (Mandatory) Allowed Values: The values are depending on the procedure type. Inbound Procedure "whinh3101m000" - Shipment Notice - Lines "whinh3512m000" - Receipts "whinh3201m000" - Generate Inbound Advice "whinh3203m000" - Put Away Inbound Advice "whinh3415m000" - Print Storage List "whinh3525m100" - Confirm Storage List Inspection Procedure "whinh3122m000" - Inspection "whinh3201m000" - Generate Inbound Advice "whinh3203m000" - Put Away Inbound Advice "whinh3415m000" - Print Storage List "whinh3525m100" - Confirm Storage List Outbound Procedure: "whinh4200m000" - Process Outbound Advice "whinh4201m000" - Generate Outbound Advice "whinh4202m000" - Release Outbound Advice "whinh4415m000" - Print Picking List "whinh4525m100" - Confirm Picking List "whinh3122m000" - Inspection Shipment Procedure: "whinh4275m001" - Freeze Shipment "whinh4275m000" - Confirm Shipment Fields below are applicable when Activity = "whinh3101m000"; otherwise these fields will be ignored. iShipFromBusinessPartner- Ship from BP (Mandatory) iShipmentNotice - Shipment Notice (Mandatory) iShipmentNoticeLine - Shipment Notice Line (Mandatory) Fields below are applicable when Activity = "whinh3512m000"; otherwise these fields will be ignored. iReceipt - Receipt (Mandatory) iReceiptLine - Receipt Line (Mandatory) iBomLine - BOM Line (Optional) Fields below are applicable when Activity = "whinh3201m000" or "whinh3203m000" or "whinh3415m000" or "whinh3525m100"; otherwise these fields will be ignored. iInboundAdvice - Inbound Advice (Mandatory) iAdviceLine - Inbound Advice Line (Mandatory) iBomLineSequence- BOM Line Sequence (Optional) Fields below are applicable when Activity = "whinh3122m000"; otherwise these fields will be ignored. iInspection - Inspection (Mandatory)

	<pre> iInspectionSequence - Inspection Sequence (Mandatory) iBomLineSequence- BOM Line Sequence (Optional) Fields below are applicable when Activity = "whinh4200m000" or "whinh4201m000" or "whinh4202m000" or "whinh4415m000" or "whinh4525m100"; otherwise these fields will be ignored. iOrderOrigin - Order Origin (Mandatory) iOrder - Order (Mandatory) iOrderSet - Order Set (Optional) iOrderLine - Order Line (Optional) iOrderSequence - Order Sequence (Optional) iOutboundAdvice - Outbound Advice Number (Mandatory) Fields below are applicable when Activity = "whinh4275m001" or "whinh4275m000"; otherwise these fields will be ignored. iShipment - Shipment (Mandatory) iShipmentLine - Shipment Line (Mandatory) iLabelPrinting - Print Labels setup (Mandatory) Allowed Values: whinh.plbs.override Override Order Settings whinh.plbs.order.settingsAccording Order Settings The fields below are applicable when Print Labels Setup is on Override Order Settings iLabelLayout - Label Layout (Mandatory if applicable) iLabelPrintedBy - Label print by Infor LN or external application (Mandatory) iNumberOfCopies - Number of Copies (Mandatory if applicable) iHandlingUnitOnly - Print labels only if handling units are present for the activity. (Mandatory) iPrintingMethod - Printing Method Allowed Values: whinh.prtm.by.line - one label per line whinh.prtm.by.unit - one label per unit (Mandatory if applicable) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error </pre>

WarehouseOrder.StartDetail

DLL:	whextinhapi
Syntax:	<pre> long WarehouseOrder.StartDetail(long iStartMode, domain whinh.oorg iOrderOrigin, domain tcorno iOrder, </pre>

	<pre> domain tcmcs.s999m iOrderSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the details session Warehousing Orders (whinh2100m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Primary Key Fields: iOrderOrigin iOrder iOrderSet - iOrderOrigin, iOrder and iOrderSet must refer to an existing warehousing order. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - Session started - Error

WarehouseOrder.StartMultiMain

DLL:	whextinhapi
Syntax:	<pre> long WarehouseOrder.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain whinh.oorg iOrderOrigin, domain tcorno iOrder, domain tcwset iOrderSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the multi main table session Warehousing Orders (whinh2100m100). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated

	simultaneously. iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend A specific query to be used when zooming to this session. Primary Key Fields: iOrderOrigin iOrder iOrderSet Output: oExceptionMessage oExceptionID Return: 0 <> 0	- iOrderOrigin, iOrder and iOrderSet must refer to an existing warehousing order. - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Session started - Error

WarehouseOrder.StartOverview

DLL:	whextinhapi
Syntax:	<pre>long WarehouseOrder.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain whinh.oorg iOrderOrigin, domain tcorno iOrder, domain tcwset iOrderSet, domain tccfrw iCarrier, domain tcncmp iShipFromCompany, domain tcbsite iShipFromSite, domain tctyps iShipFromType, domain tccshp iShipFromCode, domain tcncmp iShipToCompany, domain tcbsite iShipToSite, domain tctyps iShipToType, domain tccshp iShipToCode, domain tcitem iItem, ref domain whinh.oorg oOrderOrigin, ref domain tcorno oOrder, ref domain tcwset oOrderSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the overview session Warehousing Orders (whinh2100m000). Input: iStartMode Specifies the start mode for the session.

	Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byOrderOrigin": data is filtered for order origin order by can be set with iSessionIndex "byOrder": data is filtered for order order by can be set with iSessionIndex "byCarrier": data is displayed by Carrier session will be started on index 3 view field: Carrier "byShipFrom": data is displayed by: iShipFromSite iShipFromCompany iShipFromType iShipFromCode session will be started on index 4 view fields: ShipFromSite (if sites applicable) ShipFromType "byShipTo": data is displayed by: iShipToSite iShipToCompany iShipToType iShipToCode session will be started on index 5 view fields: ShipToSite (if sites applicable) ShipToType "byItem": data is filtered for: iItem (mandatory) iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iOrderOrigin Mandatory when iStartFilter "byOrderOrigin" or "byOrder" is used and the session is started with start mode MODELESS iOrder Mandatory when iStartFilter "byOrder" is used and the session is started with start mode MODELESS iOrderSet Not mandatory. iCarrier Not mandatory.

	<pre> iShipFromCompany Not mandatory. iShipFromType Not mandatory. iShipFromCode Not mandatory. iShipFromSite Not mandatory. iShipToCompany Not mandatory. iShipToType Not mandatory. iShipToCode Not mandatory. iShipToSite Not mandatory. iItem Mandatory when iStartFilter "byItem" is used and the session is started in with start mode MODELESS Output: for iStartMode MODAL: oOrderOrigin oOrder oOrderSet oExceptionMessage oExceptionID Return: 0 <> 0 </pre>
	<pre> - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Session started - Error </pre>

Public Interfaces for PurchaseOrderAdvice

The following functions are available:

[PurchaseOrderAdvice.Transfer](#)

PurchaseOrderAdvice.Transfer

DLL:	whextinaapi
	This function is available from KB2271572 .

Syntax:	<pre> long PurchaseOrderAdvice.Transfer(domain tcorno iPurchaseOrderAdvice, domain tcseri iOrderSeries, domain tcseri iDirectDeliverySeries, domain tccotp iOrderType, domain tccotp iDirectDeliveryOrderType, domain tccotp iDirectDeliveryReturnOrderType, ref domain tcorno oPurchaseOrderOrSchedule, ref domain tcpur.otyp oOrderType, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface will transfer a purchase advice to a purchase order or purchase schedule. This depends on the set-up in the purchase module. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iPurchaseOrderAdvice The purchase order advice which needs to be transferred. This is mandatory to fill. iOrderSeries The order series which has to be used when the advice is transferred to a purchase order. If empty the default order series will be used. iDirectDeliverySeries The direct delivery order series which has to be used when the advice is transferred to a direct delivery purchase order. If empty the default direct delivery order series will be used. iOrderType The order type which has to be used when the advice is transferred to a purchase order. If empty the default order type will be used. iDirectDeliveryOrderType The direct delivery order type which has to be used when the advice is transferred to a direct delivery purchase order. If empty the default direct delivery order type will be used. iDirectDeliveryReturnOrderType The direct delivery return order type which has to be used when the advice is transferred to a direct delivery return purchase order. If empty the default direct delivery return order type will be used. Output: oPurchaseOrderOrSchedule The purchase order or purchase schedule to which the purchase advice is transferred. oOrderType The purchase order or purchase schedule indicator oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - The purchase order advice has been transferred successfully. <> 0 - Error. The purchase order advice could not be transferred.

Public Interfaces for ProductionOrderAdvice

The following functions are available:

[ProductionOrderAdvice.StartDetail](#)

[ProductionOrderAdvice.StartOverview](#)

ProductionOrderAdvice.StartDetail

DLL:	whextinaapi										
	This function is available from KB2297822 .										
Syntax:	<pre> long ProductionOrderAdvice.StartDetail(long iStartMode, domain tcpdno iOrderNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>										
Usage:	Expl: This function starts the detail session Production Order Advice (whina3100m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variable form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: <table border="0"> <tr> <td>iOrderNumber</td><td>- The primary key fields must refer to an existing Production Order Advice</td></tr> <tr> <td>Output: oExceptionMessage</td><td>- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.</td></tr> <tr> <td>oExceptionID</td><td>- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.</td></tr> </table> Return: 0 <> 0 <table border="0"> <tr> <td></td><td>- Session started</td></tr> <tr> <td></td><td>- Error.</td></tr> </table>	iOrderNumber	- The primary key fields must refer to an existing Production Order Advice	Output: oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.	oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.		- Session started		- Error.
iOrderNumber	- The primary key fields must refer to an existing Production Order Advice										
Output: oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.										
oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.										
	- Session started										
	- Error.										

ProductionOrderAdvice.StartOverview

DLL:	whextinaapi This function is available from KB2297822 .
Syntax:	<pre> long ProductionOrderAdvice.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcpdno iOrderNumber, domain tcitem iItem, domain tccwar iWarehouse, ref domain tcpdno oOrderNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl This function starts the overview session Production Order Advice (whina3l00m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byOrder": data is displayed by Order session will be started on index 1 view field: None "byItem": data is displayed by Item session will be started on index 2 view field: Item "byWarehouse": data is displayed by Warehouse session will be started on index 3 view field: Warehouse iSessionIndex Specifies the table index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iOrderNumber The Production Order Advice to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 1 (or iStartFilter = "byOrder") iItem The Item to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 2 (or iStartFilter = "byItem") iWarehouse The Warehouse to be started

	Mandatory if iStartMode = MODELESS and iSessionIndex = 3 (or iStartFilter = "byWarehouse") Output: for iStartMode MODAL: oOrderNumber - Production Order Advice of selected record oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

Public Interfaces for PackageDefinition

The following functions are available:

[PackageDefinition.DetermineMultipleOfQuantity](#)

[PackageDefinition.Validate](#)

PackageDefinition.DetermineMultipleOfQuantity

DLL:	whextwmdapi
Syntax:	<pre>long PackageDefinition.DetermineMultipleOfQuantity(domain whwmd.pkdf iPackageDefinition, domain tcitem iItem, ref boolean oMultipleOfQuantityFound, ref domain tcqivl oMultipleOfQuantity, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface determines the multiple of quantity which is required for shipping the item. Within the handling unit template the user can define flags called "Full Packages Only" and "Full Layers Only". When one of these flag is set, then the package definition requires full packages/layers only, which impacts the order quantity. The following examples will reflect what is determined in this function. EXAMPLE 1: <pre>----- Node Pack Qty Qty Stack Full Full Fill Item Pack St Height Pack Layers Up</pre>

		Item	Unit		Only	Only	With
1	Pallet	1	100 pcs	0	Yes	No	N.A.
2	Box	10	100 pcs	5	Yes	No	N.A.
oMultipleOfQuantity = 100.0 (Pallet must be full)							

EXAMPLE 2:							

Node	Pack	Qty	Qty	Stack	Full	Full	Fill
	Item	Pack	St	Height	Pack	Layers	Up
		Item	Unit		Only	Only	With
1	Pallet	1	100 pcs		No	No	N.A.
2	Box	10	100 pcs		Yes	No	N.A.
oMultipleOfQuantity = 10.0 (Box must be full)							

EXAMPLE 3:							

Node	Pack	Qty	Qty	Stack	Full	Full	Fill
	Item	Pack	St	Height	Pack	Layers	Up
		Item	Unit		Only	Only	With
1	Pallet	1	100 pcs	0	No	No	N.A.
2	Box	10	100 pcs	5	No	Yes	FullHUs
oMultipleOfQuantity = 20.0 (Box Layer contains 2 Boxes of 10 pcs)							

EXAMPLE 4:							

Node	Pack	Qty	Qty	Stack	Full	Full	Fill
	Item	Pack	St	Height	Pack	Layers	Up
		Item	Unit		Only	Only	With
1	Pallet	1	100 pcs	0	No	No	N.A.
2	Box	10	100 pcs	5	Yes	Yes	FullHUs
oMultipleOfQuantity = 20.0 (Box Layer contains 2 Boxes of 10 pcs)							

EXAMPLE 5:							

Node	Pack	Qty	Qty	Stack	Full	Full	Fill
	Item	Pack	St	Height	Pack	Layers	Up
		Item	Unit		Only	Only	With
1	Pallet	1	100 pcs	0	No	No	N.A.
2	Box	10	100 pcs	5	No	Yes	EmptyHU.
oMultipleOfQuantity is 0.0, since Empty Handling Units are used and zero quantities are allowed.							

EXAMPLE 6:							

Node	Pack	Qty	Qty	Stack	Full	Full	Fill
	Item	Pack	St	Height	Pack	Layers	Up
		Item	Unit		Only	Only	With
1	Pallet	1	100 pcs	0	No	No	N.A.
2	Box	10	100 pcs	5	Yes	Yes	EmptyHU.

	<pre> oMultipleOfQuantity = 10.0 (Box must be full) The oMultipleOfQuantity is always in inventory unit! The iPackageDefinition and iItem combination must exist in the entity Package Definitions by Item (whwmd4130m000) and the Package Definition must be Validated. When no multiple of quantity is to be used, as there is no full packages only or full layers only setup for the package definition, the oMultipleOfQuantity remains zero. Pre: N.a. Post: N.a. Input: iPackageDefinition - Package Definition (Mandatory) Package Definition Type should be Variable, otherwise an error is given iItem - Item (Mandatory) Output: oMultipleOfQuantityFound - Multiple of quantity is found, for the given iPackageDefinition and iItem oMultipleOfQuantity - Multiple of quantity in inventory unit. Will only be filled when a multiple of quantity can be determined, otherwise zero. Return: 0: OK, <> 0: Error </pre>

PackageDefinition.Validate

DLL:	whextwmdapi This function is available from KB3562830 .
Syntax:	<pre> long PackageDefinition.Validate(domain tcitem iItem, domain whwmd.pkdf iPackageDefinition, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl: This Public Interface validates package definition. Pre: db.retry.point() is set. Post: commit.transaction() / abort.transaction() Input: iItem - Item (Optional). If empty not item specific package definition will be validated. iPackageDefinition - Package Definition (Mandatory) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Package Defintion validated succesfully DALHOOKERROR - Package Definition not validated; error message set. </pre>

Public Interfaces for HandlingUnit

The following functions are available:

[HandlingUnit.ApproveRemaining](#)

[HandlingUnit.CalculateWeightAndDimensions](#)

[HandlingUnit.Close](#)

[HandlingUnit.ConfirmReceipt](#)

[HandlingUnit.Create](#)

[HandlingUnit.GenerateEmptyHandlingUnits](#)

[HandlingUnit.GenerateLots](#)

[HandlingUnit.GenerateSerials](#)

[HandlingUnit.Link](#)

[HandlingUnit.MoveToLoad](#)

[HandlingUnit.MoveToShipment](#)

[HandlingUnit.PrintLabelByPackageLabelBOD](#)

[HandlingUnit.PrintLabels](#)

[HandlingUnit.Receive](#)

[HandlingUnit.RejectRemaining](#)

[HandlingUnit.RepackInInventory](#)

[HandlingUnit.SetToNotShipped](#)

[HandlingUnit.SetToShipped](#)

[HandlingUnit.StartAutomaticInboundProcessing](#)

[HandlingUnit.StartDetail](#)

[HandlingUnit.StartOverview](#)

[HandlingUnit.StartPrintLabels](#)

[HandlingUnit.Transfer](#)

[HandlingUnit.Unlink](#)

[HandlingUnit.Unpack](#)

HandlingUnit.ApproveRemaining

DLL:	whextwmdapi
Syntax:	<pre> long HandlingUnit.ApproveRemaining(domain whhuid iHandlingUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function Approves the (remaining) quantity of the given handling unit that is not yet approved, rejected, or (inbound only) destroyed. If no inspection results have been specified for the handling unit yet, all items are approved. The Warehouse Inspection should have status Open or In Process and not reside in a WMS-controlled warehouse. This function does not process the inspection results. For processing, use WarehouseInspection.Process Pre: db.retry.point() is set. Post: commit.transaction() / abort.transaction() Input: iHandlingUnit - Handling Unit (mandatory) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

HandlingUnit.CalculateWeightAndDimensions

DLL:	whextwmdapi
	This function is available from KB3529999 .
Syntax:	<pre> long HandlingUnit.CalculateWeightAndDimensions(domain whhuid iHandlingUnit, boolean iOVERRIDEFixedWeightAndDimensions, ref domain tccuni oWeightUnit, ref domain tcwght oGrossWeight, ref domain tcwght oNetWeight, ref domain tccuni oDimensionUnit, ref domain tcleng oDepth, ref domain tcleng oWidth, ref domain tcleng oHeight, ref domain tcflsp oFloorSpace, ref domain tcvolm oVolume, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface will calculate and optionally update weight and dimensions of iHandlingUnit. Pre: db.retry.point() is set. Post: commit/abort.transaction. Input: iHandlingUnit - Handling Unit (Mandatory).

	<code>iOverrideFixedWeightAndDimensions</code> - if <code>iHandlingUnit</code> is marked with a flag <code>Fixed Dimensions</code> this flag will be reset and Weight and Dimensions updated (Mandatory) Output: <code>oWeightUnit</code> - Weight Unit <code>oGrossWeight</code> - Calculated Gross Weight <code>oNetWeight</code> - Calculated Net Weight <code>oDimensionUnit</code> - Dimension Unit <code>oDepth</code> - Calculated Depth <code>oWidth</code> - Calculated Width <code>oHeight</code> - Calculated Height <code>oFloorSpace</code> - Calculated Floor Space <code>oVolume</code> - Calculated Volume <code>oExceptionMessage</code> - The last message if any message is found. If more than one message is given, these are present in the <code>oExceptionID</code>. <code>oExceptionID</code> - An ID that refers to all error information. Use the functions in <code>Exception</code> to get all relevant information. Return: 0 - Handling Unit structure is validated successfully <> 0 - Error

HandlingUnit.Close

DLL:	whextwmdapi
Syntax:	<pre>long HandlingUnit.Close(domain whhuid iHandlingUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface will set the status of a handling unit and all handling unit childs if any to Closed. <code>iHandlingUnit</code> can be closed if: <code>iHandlingUnit</code> is not blocked; <code>iHandlingUnit</code> has status <code>Inactive</code> or <code>iHandlingUnit</code> has status <code>Shipped</code> and related shipment line is already confirmed or <code>iHandlingUnit</code> has status <code>InStock</code> and - Ownership registration or allocation registration in the Item Data by Warehouse (see session <code>whwmd2110s000</code>) are not defined at a Physical Item Level. - <code>iHandlingUnit</code> is not under location change (session <code>whwmd3105m000</code>); - <code>iHandlingUnit</code> is not present with an active or proposed stock in the Prepacking Advice (sessions <code>whwmd5640m000</code>). Pre: <code>db.retry.point()</code> Post: <code>abort.transaction()</code> or <code>commit.transaction()</code> Input: <code>iHandlingUnit</code> - Handling Unit which must be closed (Mandatory) Output: <code>oExceptionMessage</code> - The last message if the return value is not equal to 0. If more than one message is given, these are present in the <code>oExceptionID</code> <code>oExceptionID</code> - An ID that refers to all error information. Use the functions in <code>Exception</code> to get all relevant information. Return: 0 - <code>iHandlingUnit</code> is closed. <> 0 - Error

HandlingUnit.ConfirmReceipt

DLL:	whextinhapi
Syntax:	<pre> long HandlingUnit.ConfirmReceipt(domain whhuid iHandlingUnit, ref boolean oHandlingUnitConfirmed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function will confirm the given handling unit. When successful, also the automatic inbound process is started for this receipt line. Be aware that transaction management is handled within this function. Pre: N.a. Post: N.a. Input: iHandlingUnit - Mandatory Output: oHandlingUnitConfirmed - Handling Unit is confirmed oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

HandlingUnit.Create

DLL:	whextwmdapi
Syntax:	<pre> long HandlingUnit.Create(domain whhuid iHandlingUnit, domain tcuef.mask iHandlingUnitMask, domain tcyesno iSSCCMask, domain tcitem iPackagingItem, domain tcqivl iPackagingItemQuantity, domain tccwar iWarehouse, domain tccom.bpid iBusinessPartner, domain whinh.oorg iOrderOrigin, domain tcorno iOrderNumber, domain tcpono iOrderLine, domain tcpono iOrderLineSequence, ref domain whhuid oHandlingUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>

Usage:	Expl: This Public Interface will create an inactive handling unit. Handling unit can contain some extra information based on input data: Packaging item and packaging item quantity (when not empty); Order line related data (when known): - Warehouse; - Business Partner; Handling Unit mask will be defaulted following below described priority: - iHandlingUnitMask (if not empty) - Business Partner (when iBusinessPartner is filled or when the Business Partner can be defaulted based on the passed order line). - Business Partner in Ship To role mask; - Business Partner in Sold To role mask; - Warehouse Internal mask (if iWarehouse is filled and internal mask is defined); Note: if iWarehouse and/or iBusinessPartner are empty they can be read from the passed order line if any; - Inventory Parameters mask Input data will be verified as follows: - If iHandlingUnit is not empty, handling unit with the same Id cannot exist in the system (table whwmd530); - If iPackagingItem is not empty, iPackagingItemQuantity must have a positive value. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iHandlingUnit - The Handling Unit ID (if empty a handling unit Id will be generated automatically based on handling unit mask). iHandlingUnitMask (optional) iSSCCMask - SSCC mask (yes/no) iPackagingItem - Packaging item (optional) iPackagingItemQuantity - Packaging Item Quantity iWarehouse - Warehouse (optional) iBusinessPartner - Business Partner (optional) iOrderOrigin - Order Origin (optional) iOrderNumber - Order Number (optional) iOrderLine = Order Line (optional) iOrderLineSequence - Order Line Sequence (optional) Output: oHandlingUnit - Generated handling unit oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Handling Unit has been generated successfully <> 0 - Error

HandlingUnit.GenerateEmptyHandlingUnits

DLL:	whextwmdapi
Syntax:	<pre>long HandlingUnit.GenerateEmptyHandlingUnits(domain whhuid iHandlingUnit, ref domain tcmcs.s999m oExceptionMessage mb,</pre>

	ref	long	oExceptionID)
Usage:	Expl:	This public interface will generate empty handling units for the given iHandlingUnit. This public interface will only generate new empty handling units when the following conditions are met: - Package Definition must be filled on the iHandlingUnit - Handling Unit Template used for the Package Definition should have nodes which use the fill up with empty packages enabled. - iHandlingUnit should not be a multi item handling unit - iHandlingUnit should have status Projected or Staged - iHandlingUnit should have child Handling Units - iHandlingUnit should not be empty itself The functionality will make sure that based on the handling unit template, the packaging structure is stabilized. For example, a pallet can contain 4 crates, when only 3 crates are put on a pallet, it can be required to add an empty crate (no items are inside) to form a stable packaging structure. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iHandlingUnit - Handling Unit for which empty handling units must be created (Mandatory) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Handling Unit is stabilized <> 0 - Error	

HandlingUnit.GenerateLots

DLL:	whextwmdapi
Syntax:	<pre>long HandlingUnit.GenerateLots(domain whhuid iHandlingUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface will generate lot(s) for the handling unit. Generating lot code for the handling unit is allowed when: - iHandlingUnit has status Inactive, Open or Receipt Open. - iHandlingUnit contains lot-controlled item(s). - iHandlingUnit does not contain lots. - iHandlingUnit with a status Receipt Open is not in transit. When handling unit has status Receipt Open, generated lots will be also stored in the table Receipt Line Stock Point Details (whinh320). Example: Handling Unit HU1 contains 3 pieces of item. Item is lot-controlled and serialized. Serials SER1, SER2 and SER3 are already stored in handling unit. Lot LOT1 is generated. -----

	<pre> HU stock point details Receipt Line (whwmd536) Stock Point Details (whinh320) ----- HU1 LOT1 SER1 1 pcs LOT1 SER1 1 pcs HU1 LOT1 SER2 1 pcs LOT1 SER2 1 pcs HU1 LOT1 SER3 1 pcs LOT1 SER3 1 pcs </pre> Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iHandlingUnit - for this handling unit the lot code will be generated (Mandatory) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Lot is generated and linked to iHandlingUnit. <> 0 - Error

HandlingUnit.GenerateSerials

DLL:	whextwmdapi
Syntax:	<pre> long HandlingUnit.GenerateSerials(domain whhuid iHandlingUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will generate serials for the handling unit. Generating serials for the handling unit is allowed when: - iHandlingUnit has status Inactive, Open or Receipt Open. - iHandlingUnit contains serialized item(s). - iHandlingUnit does not contain serials. - iHandlingUnit with a status Receipt Open is not in transit. If item is serialized NOT IN inventory, generated serials always will be stored in the entity Handling Unit Stock Point Details (table whwmd536) of the bottom level Handling Units. Example 1: Handling Unit HU1 contains 3 pieces of item. Item is serialized not in inventory. Serials SER1, SER2 and SER3 are generated. <pre> ----- HU stock point details Receipt Line (whwmd536) Stock Point Details (whinh320) ----- HU1 SER1 1 pcs SER1 1 pcs HU1 SER2 1 pcs SER2 1 pcs HU1 SER3 1 pcs SER3 1 pcs </pre> If item is serialized IN inventory and handling unit quantity is greater than 1.0 and no package definition is defined or handling unit template does not allow multiple stock points (field Allow Multi Stock Points in the table whwmd460

is on NO), extra handling units can be generated to store serial numbers.

Example 2:

Handling Unit HU1 contains 3 pieces of item, no package definition.

Item is serialized in inventory.

Serials SER1, SER2 and SER3 are generated.

Child handling units HU11, HU12 and HU13 are generated and linked to HU1

```
-----
| HU stock point details | Receipt Line
| (whwmd536)            | Stock Point Details (whinh320)
-----
| HU11 SER1  1 pcs      | SER1  1 pcs
| HU12 SER2  1 pcs      | SER2  1 pcs
| HU13 SER3  1 pcs      | SER3  1 pcs
-----
```

If item is serialized IN inventory and handling unit quantity is greater than 1.0, package definition is defined and handling unit template allows multiple stock points (field Allow Multi Stock Points in the table whwmd460 is on Yes), generated serials will be stored in the entity Handling Unit Stock Point Details (table whwmd536) of the bottom level Handling Units.

Example 3:

Handling Unit HU1 contains 3 pieces of item.

Package definition is known and handling unit template allows multiple stock points,

Item is serialized in inventory.

Serials SER1, SER2 and SER3 are generated.

```
-----
| HU stock point details | Receipt Line
| (whwmd536)            | Stock Point Details (whinh320)
-----
| HU1  SER1  1 pcs      | SER1  1 pcs
| HU1  SER2  1 pcs      | SER2  1 pcs
| HU1  SER3  1 pcs      | SER3  1 pcs
-----
```

When handling unit has status Receipt Open, all generated serials will be also stored in the table Receipt Line Stock Point Details (whinh320).

Pre: db.retry.point()

Post: abort.transaction() or commit.transaction()

Input: iHandlingUnit - for this handling unit the serial numbers will be generated (Mandatory)

Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID

oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information.

Return: 0 - Serials are generated and linked to iHandlingUnit.

<> 0 - Error

HandlingUnit.Link

DLL:	whextwmdapi
Syntax:	<pre> long HandlingUnit.Link(domain whhuid iChildHandlingUnit, domain whhuid iParentHandlingUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface will compose handling unit structure by means of linking iChildHandlingUnit to iParentHandlingUnit. Input data will be verified as follows: - Both iChildHandlingUnit and iParentHandlingUnit must be filled; - Both iChildHandlingUnit and iParentHandlingUnit must be present in the table Handling Units (whwmd530); - iChildHandlingUnit and iParentHandlingUnit cannot have the same value; - Status of iChildHandlingUnit cannot be equal to: - Closed - Frozen - Partially Frozen - Confirming - Shipped - InTransit - Status of iParentHandlingUnit must be Inactive or equal to the status of iChildHandlingUnit. Combination of handling units with status Projected and Staged is also allowed. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iChildHandlingUnit - Child Handling Unit (Mandatory) iParentHandlingUnit - Parent Handling Unit (Mandatory) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Handling Unit structure is composed successfully <> 0 - Error

HandlingUnit.MoveToLoad

DLL:	whextinhapi
Syntax:	<pre> long HandlingUnit.MoveToLoad(domain whinh.shpm iShipment, domain tcpono iShipmentLine, domain whhuid iHandlingUnit, domain whinh.load iDestinationLoad, domain tccdis iReason, ref domain whinh.shpm oDestinationShipment, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface supports moving a staged or projected handling unit to the destination load.

	If handling unit quantity is less than shipment line quantity, shipment line will be split; otherwise shipment line will be moved to other load and shipment. Pre: db.retry.point(). Post: abort/commit transaction. Input: iShipment - Shipment; Mandatory iShipmentLine - Shipment Line; Mandatory iHandlingUnit - Handling Unit; Mandatory Only handling units with status Staged and Projected are allowed. iDestinationLoad - Destination Load; Mandatory iReason - Compose Reason; Mandatory if destination load is planned by FM. Output: oDestinationShipment - Destination Shipment oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

HandlingUnit.MoveToShipment

DLL:	whextinhapi
Syntax:	<pre>long HandlingUnit.MoveToShipment(domain whinh.shpm iShipment, domain tcpono iShipmentLine, domain whhuid iHandlingUnit, ref domain whinh.shpm ioDestinationShipment, domain tccdis iReason, boolean iAcceptWarnings, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface supports moving a staged or projected handling unit to the destination shipment. If handling unit quantity is less than shipment line quantity, shipment line will be split; otherwise complete shipment line will be moved to the destination shipment. Pre: db.retry.point(). Post: abort/commit transaction. Input: iShipment - Shipment; Mandatory iShipmentLine - Shipment Line; Mandatory iHandlingUnit - Handling Unit; Mandatory Only handling units with status Staged or Projected are allowed. ioDestinationShipment - Destination Shipment; Optional If empty new shipment will be created. iReason - Compose Reason; Mandatory if destination shipment belongs to the load which is planned by FM. iAcceptWarnings - Accept Warnings; Mandatory

	Output: ioDestinationShipment - Destination Shipment oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

HandlingUnit.PrintLabelByPackageLabelBOD

DLL:	whextwmdapi
Syntax:	<pre>long HandlingUnit.PrintLabelByPackageLabelBOD(domain tclabl iLabelLayout, domain tcmcs.long iNumberOfCopies, ref domain whhuid iHandlingUnitArray() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface will publish the PackageLabelBOD for the handling units that are passed in the iHandlingUnitArray(). For each handling unit the handling unit structure is searched for labeled handling units. For each handling unit passed, the PackageLabelBOD will be published. For example: Handling unit structure contains a pallet, with 2 crates and 4 boxes per crate. The pallets and crates are labeled. If the following handling unit structure is present HU0001 - Pallet - HU0002 - Crate - HU0003 - BOX - HU0004 - BOX - HU0005 - BOX - HU0006 - BOX - HU0007 - Crate - HU0008 - BOX - HU0009 - BOX - HU0010 - BOX - HU0011 - BOX When this function is called with handling unit HU0001, the following labels will be printed via the PackageLabelBOD: HU0001, HU0002, HU0007; When this function is called with handling unit HU0002, the following labels will be printed via the PackageLabelBOD: HU0002; When this function is called with handling unit HU0003, there will be no PackageLabelBOD, because handling unit HU0003 is not labeled.

	Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: <table> <tr> <td>iLabelLayout</td><td>- Label Layout</td></tr> <tr> <td>iNumberOfCopies</td><td>- Number of copies; By default 1 label is printed per labeled handling unit, so passing this variable as 1 will mean 2 labels are printed, the original one and a copy. Maximum value is 99.</td></tr> <tr> <td>iHandlingUnitArray</td><td>- Array of handling units for which labels must be printed; Mandatory</td></tr> </table> Output: N.a. Return: 0 - Success <> 0 - Error	iLabelLayout	- Label Layout	iNumberOfCopies	- Number of copies; By default 1 label is printed per labeled handling unit, so passing this variable as 1 will mean 2 labels are printed, the original one and a copy. Maximum value is 99.	iHandlingUnitArray	- Array of handling units for which labels must be printed; Mandatory
iLabelLayout	- Label Layout						
iNumberOfCopies	- Number of copies; By default 1 label is printed per labeled handling unit, so passing this variable as 1 will mean 2 labels are printed, the original one and a copy. Maximum value is 99.						
iHandlingUnitArray	- Array of handling units for which labels must be printed; Mandatory						

HandlingUnit.PrintLabels

DLL:	whextwmdapi
	This function is available from KB3560682 .
Syntax:	<pre>long HandlingUnit.PrintLabels(domain whhuid iHandlingUnit, domain whwmd.lbpb iLabelPrintedBy, domain tclabl iLabelLayout, domain tcmcs.str15 iLabelDevice, domain tcmcs.long iNumberOfCopies, domain whwmd.lbpr iPrintOption, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This Public Interface prints handling unit labels. The ranges can be based upon order activities or can be related to handling units. The ranges can be defined in the iProcessingOptionSet. The layouts of these labels are defined in session "Label Layouts" (whwmd5520m000). Within the label layout the user can define several label fields which are process dependent, a certain label field can contain a different value within multiple processes. The table fields in the layout are defined with the aid of variables preceded by hashes (#). The table fields/variables are replaced by the value of the table fields. See LN Documentation, topic Label Fields and Expressions for the possible fields. Opening/Closing of the report is handled by this function. Pre: There should be no pending logical transaction before calling

this function.

Post: No need to commit or abort the process, that is handled within the function.

Input: iHandlingUnit - Handling Unit Id (Optional).
 If iHandlingUnit is filled then iProcessingOptionSet will be ignored.
 iLabelPrintedBy - Label printed By Infor LN or External application. (Mandatory)
 iLabelLayout - Label Layout (Mandatory if iLabelPrintedBy = Infor LN)
 iLabelDevice - Label Device (Mandatory)
 iNumberOfCopies - Number of Copies (Mandatory)
 iPrintOption - Print option indicator for Handling Unit Labels only. Range or Structure. (Mandatory)
 iProcessingOptionSet Optional, if 0, the default options are applied.

Processing Options have a direct relationship with the form fields on session Print Labels (whwmd5430m100) and are not explained in further detail here.

Please refer to the session help for additional information.

Processing Options that are set while a required Implemented Software Component is not available are ignored.

Explanation about setting of defaults:

Minimum Value: Minimum value of domain is taken as default value.

Maximum Value: Maximum value of domain is taken as default value.

NAME	TYPE	DEFAULT
HandlingUnitFrom	domain whhuid	Minimum Value
HandlingUnitTo	domain whhuid	Maximum Value
ShipFromTypeFrom	domain tctyps	Minimum Value
ShipFromCodeFrom	domain tccshp	Minimum Value
ShipFromTypeTo	domain tctyps	Maximum Value
ShipFromCodeTo	domain tccshp	Maximum Value
ShipToTypeFrom	domain tctyps	Minimum Value
ShipToCodeFrom	domain tccshp	Minimum Value
ShipToTypeTo	domain tctyps	Maximum Value
ShipToCodeTo	domain tccshp	Maximum Value
OwnerFrom	domain tccom.bpid	Minimum Value
OwnerTo	domain tccom.bpid	Maximum Value
InboundOrder	domain tcyesno	tcyesno.no
InboundOrderOriginFrom	domain whinh.oorg	Minimum Value
InboundOrderOriginTo	domain whinh.oorg	Maximum Value
InboundOrderOriginFrom	domain whinh.oorg	Minimum Value
InboundOrderOriginTo	domain whinh.oorg	Maximum Value
InboundOrderLineFrom	domain tcpono	Minimum Value
InboundOrderLineTo	domain tcpono	Maximum Value
OutboundOrder	domain tcyesno	tcyesno.no
OutboundOrderOriginFrom	domain whinh.oorg	Minimum Value
OutboundOrderOriginTo	domain whinh.oorg	Maximum Value
OutboundOrderOriginFrom	domain whinh.oorg	Minimum Value
OutboundOrderOriginTo	domain whinh.oorg	Maximum Value
OutboundOrderLineFrom	domain tcpono	Minimum Value
OutboundOrderLineTo	domain tcpono	Maximum Value
AdjustmentOrder	domain tcyesno	tcyesno.no
AdjustmentOrderFrom	domain tcorno	Minimum Value
AdjustmentOrderTo	domain tcorno	Maximum Value
CycleCountOrder	domain tcyesno	tcyesno.no
CycleCountOrderFrom	domain tcorno	Minimum Value
CycleCountOrderTo	domain tcorno	Maximum Value
Inspection	domain tcyesno	tcyesno.no
InspectionFrom	domain tcorno	Minimum Value

InspectionTo	domain tcorno	Maximum Value
InspectionSequenceFrom	domain tcpono	Minimum Value
InspectionSequenceTo	domain tcpono	Maximum Value
ShipmentNotice	domain tcyesno	tcyesno.no
ShipFromBPFFrom	domain tccom.bpid	Minimum Value
ShipFromBPTo	domain tccom.bpid	Maximum Value
ShipmentNoticeFrom	domain whinh.shpm	Minimum Value
ShipmentNoticeTo	domain whinh.shpm	Maximum Value
ShipmentNoticeLineFrom	domain tcpono	Minimum Value
ShipmentNoticeLineTo	domain tcpono	Maximum Value
Receipt	domain tcyesno	tcyesno.no
ReceiptFrom	domain whinh.shpm	Minimum Value
ReceiptTo	domain whinh.shpm	Maximum Value
ReceiptLineFrom	domain tcpono	Minimum Value
ReceiptLineTo	domain tcpono	Maximum Value
InboundAdvice	domain tcyesno	tcyesno.no
InboundAdviceFrom	domain tcorno	Minimum Value
InboundAdviceTo	domain tcorno	Maximum Value
InboundAdviceLineFrom	domain tcpono	Minimum Value
InboundAdviceLineTo	domain tcpono	Maximum Value
InboundInspection	domain tcyesno	tcyesno.no
InboundInspectionFrom	domain tcorno	Minimum Value
InboundInspectionTo	domain tcorno	Maximum Value
OutboundAdvice	domain tcyesno	tcyesno.no
OutboundAdviceOrderOriginFrom	domain whinh.oorg	Minimum Value
OutboundAdviceOrderOriginTo	domain whinh.oorg	Maximum Value
OutboundAdviceOrderFrom	domain tcorno	Minimum Value
OutboundAdviceOrderTo	domain tcorno	Maximum Value
OutboundAdviceOrderSetFrom	domain tcsern	Minimum Value
OutboundAdviceOrderSetTo	domain tcsern	Maximum Value
OutboundAdviceOrderLineFrom	domain tcpono	Minimum Value
OutboundAdviceOrderLineTo	domain tcpono	Maximum Value
OutboundAdviceOrderSequenceFrom	domain tcpono	Minimum Value
OutboundAdviceOrderSequenceTo	domain tcpono	Maximum Value
OutboundAdviceFrom	domain tcpono	Minimum Value
OutboundAdviceTo	domain tcpono	Maximum Value
OutboundInspection	domain tcyesno	tcyesno.no
OutboundInspectionFrom	domain tcorno	Minimum Value
OutboundInspectionTo	domain tcorno	Maximum Value
Shipment	domain tcyesno	tcyesno.no
ShipmentFrom	domain whinh.shpm	Minimum Value
ShipmentTo	domain whinh.shpm	Maximum Value
ShipmentLineFrom	domain tcpono	Minimum Value
ShipmentLineTo	domain tcpono	Maximum Value
LoadFrom	domain whinh.load	Minimum Value
LoadTo	domain whinh.load	Maximum Value
ContainerFrom	domain whinh.cntr	Minimum Value
ContainerTo	domain whinh.cntr	Maximum Value
HandlingUnitArray	domain ttjson	0
Possible values of ReportNumber is 1.		
JSON Object HandlingUnitArray has the following structure:		
<pre> "HandlingUnitArray": [{ "HandlingUnit": "HU100" }, { "HandlingUnit": "HU200" }] </pre>		

	This structure can be created with the following code: <pre>HandlingUnitArray = Json.newArray() HandlingUnit = Json.newObject() Json.setString(HandlingUnit, "HandlingUnit", "HU100") Json.add(HandlingUnitArray, HandlingUnit) HandlingUnit = Json.newObject() Json.setString(HandlingUnit, "HandlingUnit", "HU200") Json.add(HandlingUnitArray, HandlingUnit)</pre> Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

HandlingUnit.Receive

DLL:	whextinhapi
Syntax:	<pre>long HandlingUnit.Receive(domain whhuid iHandlingUnit, domain whinh.shpm iReceipt, domain tcpksp iPackingSlip mb, domain whinh.load iLoad, domain whinh.shpm iShipment, ref domain whinh.shpm oReceipt, ref long oNumberOfReceiptLines, ref domain tcpono oReceiptLineArray(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function will receive the given handling unit. Pre: db.retry.point must be set oReceiptLineArray must be declared as a based variable, this function will allocate the memory. Post: Commit the transaction in case of success Abort the transaction in case of failure After the oReceiptLineArray is used, free the memory. Input: iHandlingUnit - Mandatory iReceipt - Receipt header on which lines must be added. Optional, if empty a new receipt header will be created. iPackingSlip - Packing Slip - Optional iLoad - Load - Optional iShipment - Shipment - Optional Output: oReceipt - Receipt that is created (or populated with the iReceipt when iReceipt is filled) oNumberOfReceiptLines - Number of receipt lines that are created. oReceiptLineArray - Receipt lines that are created. oExceptionMessage - The last message if the return value is not equal to 0.

	If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

HandlingUnit.RejectRemaining

DLL:	whextwmdapi
Syntax:	<pre>long HandlingUnit.RejectRemaining(domain whhuid iHandlingUnit, domain tccdis iRejectReason, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function Rejects the (remaining) quantity of the given handling unit that is not yet approved, rejected, or (inbound only) destroyed. If no inspection results have been specified for the handling unit yet, all items are rejected. The Warehouse Inspection should have status Open or In Process and not reside in a WMS-controlled warehouse. This function does not process the inspection results. For processing, use WarehouseInspection.Process Pre: db.retry.point() is set. Post: commit.transaction() / abort.transaction() Input: iHandlingUnit - Handling Unit (mandatory) iRejectReason - Reject Reason (mandatory) Reason must be of type 'Rejection of Goods'. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

HandlingUnit.RepackInInventory

DLL:	whextwmdapi
	This function is available from KB3528024 .
Syntax:	<pre>long HandlingUnit.RepackInInventory(domain whhuid iHandlingUnit,</pre>

	domain tcqstl iQuantityToRepack, domain tccuni iStorageUnit, boolean iRepackToTargetHandlingUnit, domain whhuid iTargetHandlingUnit, domain whwmd.pkdf iPackageDefinition, ref domain tcmcs.long oNumberHandlingUnits, ref domain whhuid oCreatedHandlingUnitsArray() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This Public Interface supports partial or complete unpack of the handling unit in inventory and moving of the unpacked parts to the existing or newly created handling units. Pre: db.retry.point() oCreatedHandlingUnitsArray must be declared as based variable. This function will allocate the memory. Post: abort.transaction() or commit.transaction() Input: iHandlingUnit - Handling Unit to be repacked (Mandatory) iHandlingUnit is valid if: - iHandlingUnit has status In Stock; - iHandlingUnit is not blocked. iQuantityToRepack - Quantity to be repacked from the handling unit (Mandatory) iStorageUnit - Storage unit (Mandatory) iRepackToTargetHandlingUnit - Repack to existing Target Handling Unit (Mandatory) iTargetHandlingUnit - defines code of existing target handling Unit (Mandatory if iRepackToTargetHandlingUnit is True, otherwise must be empty) iTargetHandlingUnit is valid if: iTargetHandlingUnit has status In Stock; iTargetHandlingUnit is not blocked; iTargetHandlingUnit is not fully packed (only for handling units with a package definition); iPackageDefinition - package definition (Must be empty when iTargetHandlingUnit is filled) Only package definition of type Variable is allowed. Output: oNumberHandlingUnits - Number of newly created top level handling units. oCreatedHandlingUnitsArray - Array of created handling units. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Handling Unit structure is unlinked successfully <> 0 - Error

HandlingUnit.SetToNotShipped

DLL:	whextwmdapi
	This function is available from KB2277253 .
Syntax:	long HandlingUnit.SetToNotShipped(

	domain whhuid iHandlingUnit, ref tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This public interface will set expected not shipped quantity in the iHandlingUnit outbound process and related shipment line. This handling unit will remain in stock when related shipment line is confirmed. iHandling.Unit can be set to not shipped if: 1. iHandlingUnit has status Staged; 2. Expected not shipped quantity in the iHandlingUnit outbound process is zero. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iHandlingUnit - Handling Unit which must be set not shipped (Mandatory) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - iHandlingUnit is set to not shipped. <> 0 - Error

HandlingUnit.SetToShipped

DLL:	whextwmdapi This function is available from KB2277253 .
Syntax:	long HandlingUnit.SetToShipped(domain whhuid iHandlingUnit, ref tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This public interface will reset expected not shipped quantity in the iHandlingUnit outbound process and related shipment line. This handling unit will be shipped when related shipment line is confirmed. iHandling.Unit can be set to shipped if: 1. iHandlingUnit has status Staged; 2. Expected not shipped quantity in the iHandlingUnit outbound process is greater than zero. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iHandlingUnit - Handling Unit which must be set shipped (Mandatory) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - iHandlingUnit is set to not shipped. <> 0 - Error

HandlingUnit.StartAutomaticInboundProcessing

DLL:	whextinhapi
Syntax:	<pre>long HandlingUnit.StartAutomaticInboundProcessing(domain whhuid iHandlingUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface will start the processing of the next automatic activity after the receipt line has been confirmed for a handling unit. This public interface will generate inbound advices when this is the next automatic activity of the related inbound order line and given handling unit. When there is already an advice present for the Handling Unit which must be putaway, please use one of the following public interfaces: InboundAdvice.StartAutomaticInboundProcessing InboundRun.StartAutomaticInboundProcessing This public interface will execute the following steps: 1) Clear any prior notifications done for the automatic inbound process, within the current process. 2) Notify the handling unit for automatic processing 3) Start the automatic inbound process for the notified Handling Unit The way to implement this public interface is to make sure that the next activity is set to automatic. This can be already set properly based on the order type setup, or can be achieved by using public interface InboundOrderLineActivity.ModifyAutomaticProcessing() for the activity that should be automatically processed. See below code example: <pre>--- db.retry.point() /* Set the activity to automatic ret = InboundOrderLineActivity.ModifyAutomaticProcessing(.. if ret <> 0 then abort.transaction() return(ret) endif commit.transaction() /* Start the automatic order process ret.val = HandlingUnit.StartAutomaticInboundProcessing(.. db.retry.point() /* Set the activity to be not automatic ret = InboundOrderLineActivity.ModifyAutomaticProcessing(..</pre>

	<pre> if ret <> 0 then abort.transaction() return(ret) endif commit.transaction() return(ret.val) ---</pre> This public interface will be able to automatically generate an inbound advice for the given handling unit. This public interface will stop the automatic process when the next activity is not set to automatic. This function must not be called within a logical transaction as this function will have it's own transaction handling. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iHandlingUnit - Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

HandlingUnit.StartDetail

DLL:	whextwmdapi
	This function is available from KB3572724 .
Syntax:	<pre> long HandlingUnit.StartDetail(long iStartMode, domain whhuid iHandlingUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session Handling Units in Detail mode (whwmd5130m000). Pre: oHandlingUnitArray must be declared as based array. Post: NA Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session.

	MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iHandlingUnit - Mandatory Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0: Error

HandlingUnit.StartOverview

DLL:	whextwmdapi This function is available from KB3572724.
Syntax:	<pre> long HandlingUnit.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, domain whhuid iHandlingUnit, domain tccwar iWarehouse, domain whloca iLocation, domain whinh.shpm iReceipt, domain tcpono iReceiptLine, domain tcorno iInspection, domain tcpono iInspectionSequence, domain whinh.shpm iShipment, domain tcpono iShipmentLine, ref long oNumberHandlingUnits, ref domain whhuid oHandlingUnitsArray() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session Handling Units (whwmd5130m000). Handling units can be selected in this session and returned to the calling function. Maximum length of output array is 30. Pre: oHandlingUnitArray must be declared as based array. Post: NA Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously.

	iStartFilter Not used iSessionIndex Specifies the session index that is to be used. 0 - No index specified 1 - By Handling Unit 3 - By Item 4 - By Parent Handling Unit 5 - By Warehouse 6 - By Receipt 7 - By Inspection 8 - By Shipment iQueryExtend A specific query to be used when zooming to this session. iItem - Mandatory if iSessionIndex = 3 iHandlingUnit - Mandatory if iSessionIndex = 1 or 4 iWarehouse - Mandatory if iSessionIndex = 5 iLocation - Optional iReceipt - Mandatory if iSessionIndex = 6 iReceipt.line - Optional iInspection - Mandatory if iSessionIndex = 7 iInspectionSequence - Optional iShipment - Mandatory if iSessionIndex = 8 iShipment.line - Optional Output: oNumberHandling.units - Number of handling units in array. (maximum value is 30) oHandlingUnitsArray - Array with handling units. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0: Error

HandlingUnit.StartPrintLabels

DLL:	whextwmdapi
	This function is available from KB3571649 .
Syntax:	<pre> long HandlingUnit.StartPrintLabels(long iStartMode, boolean iIgnoreSelectionFields, domain whhuid iFromHandlingUnit, domain whhuid iToHandlingUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl This function starts the session Print Labels (whwmd5430m100). Depending on the main table of the calling session, Non-Consecutive Record Selection (NCRS) is used. When the main table is:

	Handling Units (whwmd530) or Prepacking Advice (whwmd540) or Inventory Ownership Change Orders (whinh100) or Inventory Ownership Change Order Lines (whinh110) or Inventory Ownership Change Order - Inventory Movement (whinh115) or Allocation Change Orders (whinh120) or Allocation Change Order Lines (whinh130) or Warehousing Orders (whinh200) or Inbound Order Lines (whinh210) or Warehouse Inspections (whinh211) or Inbound Advice (whinh215) or Outbound Order Lines (whinh220) or Outbound Advice (whinh225) or Warehouse Inspection Handling Units (whinh234) or ASN Headers (whinh300) or ASN Lines (whinh301) or Receipt Headers (whinh310) or Receipt Lines (whinh312) or Shipping Containers (whinh425) or Shipments (whinh430) or Shipment Lines (whinh431) or Cycle Counting Orders (whinh500) or Cycle Counting Order Lines (whinh501) or Adjustment Orders (whinh520) or Adjustment Order Lines (whinh521) then the range selections will be imported based on the selected records in the parent session and the selection related fields will be ignored. Input: iStartMode Not Used. iIgnoreSelectionFields If true, From/To selection fields will not be filled in the session. iFromHandlingUnit From HandlingUnit selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToHandlingUnit To HandlingUnit selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

HandlingUnit.Transfer

DLL:	whextwmdapi

	This function is available from KB3536908 .
Syntax:	<pre> long HandlingUnit.Transfer(domain tcseri iSeries, domain tccopt iOrderType, domain whhuid iHandlingUnit, domain tcncmp iShipToCompany, domain tctyps iShipToType, domain tccshp iShipToCode, domain whloca iShipToLocation, domain tccprj iProject, domain tcpdm.cspa iElement, domain tcpdm.cact iActivity, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface will support direct transfer of handling unit from one warehouse/location to another. Both intra- and inter warehouse transfers are supported. Handling unit must have status In Stock. Pre: NA Post: No need to commit or abort the process, that is handled within the function. Input: iSeries - Series for transfer order to be created (Optional) iOrderType - Order type for transfer order to be created (Optional) iHandlingUnit - Handling Unit (Mandatory) iShipToCompany - Transfer To Company (Optional, current company will used if 0) iShipToType - Transfer To Type (Mandatory, only value tctyps.warehouse is allowed) iShipToCode - Transfer To Code (Mandatory) iShipToLocation - Transfer To Location (Mandatory if iShipToCode is warehouse with locations and iHandlingUnit contains location-controlled item) iProject - Project (Must be empty when project pegging is not implemented) iElement - Project Element (Must be empty when project pegging is not implemented) iActivity - Project Activity (Must be empty when project pegging is not implemented) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Handling Unit has been transferred successfully DALHOOKERROR - Otherwise.

HandlingUnit.Unlink

DLL:	whextwmdapi
Syntax:	<pre> long HandlingUnit.Unlink(domain whhuid iHandlingUnit, ref domain tcmcs.s999m oExceptionMessage mb, </pre>

	ref long oExceptionID)
Usage:	Expl: This Public Interface will unlink the handling unit from its parent handling unit. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iHandlingUnit - Handling Unit (Mandatory) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Handling Unit structure is unlinked successfully <> 0 - Error

HandlingUnit.Unpack

DLL:	whextwmdapi
Syntax:	<pre> long HandlingUnit.Unpack(domain whhuid iHandlingUnit, boolean iAllowHandlingUnitWithChilds, domain tcqstl iQuantityToSplit, domain tccuni iStorageUnit, domain whinh.oorg iOrderOrigin, domain tcorno iOrder, domain tcwset iOrderSet, domain tcpono iOrderLine, domain tcpono iOrderSequence, boolean iSetTargetHandlingUnitId, domain whhuid iTargetHandlingUnitId, boolean iToPackageDefinition, domain whwmd.pkdf iPackageDefinition, domain tcitem iPackagingItem, domain tcqivl iQuantityOfPackagingItem, boolean iCheckSplittable, domain tcmcs.long iNumberOfStockPointDetails, ref domain tcclot iStockPointDetailLotArray() fixed, ref domain tcibd.sern iStockPointDetailSerialArray() fixed, ref domain tcqstl iStockPointDetailQuantityArray(), ref domain tccuni iStockPointDetailStorageUnitArray() fixed, ref domain tcmcs.long oNumberHandlingUnits, ref domain whhuid oCreatedHandlingUnitsArray() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface supports unpack of the handling unit and moving of the unpacked parts to the newly created handling units. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iHandlingUnit - Handling Unit to be unpacked (Mandatory) iHandlingUnit is valid if: - iHandlingUnit has status Receipt Open, Received, Approved, In Stock, Partially Allocated, Partially Released or Staged; - iHandlingUnit has empty package definition or package definition of type Variable. iAllowHandlingUnitWithChilds - Allow unpack of the handling unit

	with children. If this parameter is on True iHandlingUnit can have bottom level handling units. First bottom level handling unit with quantity which is greater than iQuantityToSplit will be unpacked. If no such handling unit is found an error message will be returned. iQuantityToSplit - Quantity to be split from the handling unit (Mandatory) iStorageUnit - Storage unit (Mandatory) iOrderOrigin - Order origin (Optional) iOrder - Order number (Optional) iOrderSet - Order set (Optional) iOrderLine - Order Line (Optional) iOrderSequence - Order Line Sequence (Optional) Order data is needed to generate handling unit code for the new handling unit if handling unit mask contains order related fields. iSetTargetHandlingUnitId - Target Handling Unit code is defined (Mandatory) iTargetHandlingUnitId - defines code of the new handling unit. Note: it is not allowed to unpack handling unit to the existing target handling unit in stock. This parameter only defines the code of the newly created handling unit. If handling unit with this code already exists it must have status Inactive. If iTargetHandlingUnitId is empty a new handling unit will be generated. (Mandatory if iSetTargetHandlingUnitId is True) iToPackageDefinition - Create new handling unit with a package definition (Mandatory) iPackageDefinition - package definition (Mandatory if iToPackageDefinition is True) Only package definition of type Variable is allowed. iPackageDefinition is only allowed when iHandlingUnit has status In Stock, Partially Allocated or Partially Released. iPackageDefinition is not allowed if iTargetHandlingUnitId is filled. Reason: multiple handling units can be generated when repacking to the new iPackageDefinition. iPackagingItem - packaging item (Optional) iPackagingItem is not allowed in combination with iPackageDefinition. iQuantityOfPackagingItem - Quantity of packaging item (Mandatory if iPackagingItem is filled) iCheckSplittable - Check whether handling unit is splittable. (Mandatory) If iCheckSplittable is False a checkbox Splittable in iHandlingUnit will be ignored. iNumberOfStockPointDetails - Number of stock point details (Mandatory) iStockPointDetailLotArray - Array of Lots iStockPointDetailSerialArray - Array of serials iStockPointDetailQuantityArray - Array with quantities iStockPointDetailStorageUnitArray - Array with storage units Output: oNumberHandlingUnits - Number of newly created top level handling units. oCreatedHandlingUnitsArray - Array of created handling units. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use

	the functions in Exception to get all relevant information. Return: 0 - Handling Unit structure is unlinked successfully <> 0 - Error

Public Interfaces for HandlingUnitStructure

The following functions are available:

[HandlingUnitStructure.GetBottomLevel](#)

[HandlingUnitStructure.GetComponentLevel](#)

[HandlingUnitStructure.GetInventoryLevel](#)

[HandlingUnitStructure.GetOrderLineLevel](#)

[HandlingUnitStructure.GetWholeTree](#)

HandlingUnitStructure.GetBottomLevel

DLL:	whextwmdapi
Syntax:	<pre>long HandlingUnitStructure.GetBottomLevel(domain whhuid iHandlingUnit, ref domain tcmcs.long oNumberOfHandlingUnits, ref domain whhuid oHandlingUnitArray() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface will search in the handling unit structure, starting at the iHandlingUnit for bottom level handling units. To explain this is more detail some examples will be given: Handling Unit Structure has three levels: Level 1 - container; Level 2 - pallets; Level 3 - boxes. This function will return array with all boxes. Pre: NA Post: NA Input: iHandlingUnit - Handling Unit (mandatory) Output: oNumberHandlingUnits - Number of handling units in array. oHandlingUnitArray - Array with handling units. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID

	<code>oExceptionID</code> - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

HandlingUnitStructure.GetComponentLevel

DLL:	whextwmdapi
Syntax:	<pre>long HandlingUnitStructure.GetComponentLevel(domain whhuid iHandlingUnit, ref domain tcmcs.long oNumberOfHandlingUnits, ref domain whhuid oHandlingUnitArray() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface will search in the handling unit structure, starting at the iHandlingUnit for handling units which contain components of the main item. To explain this in more detail an example will be given: Handling Unit Structure is created for List/BOM item and contains both main item MAIN_ITEM and components COMP1 and COMP2. Handling Unit Structure has three levels: Level 1 - HU1 container with MAIN_ITEM Level 2 - HU2.1 pallet with component COMP1 Level 3 - HU 3.1 10 boxes with component COMP1 Level 2 - HU2.2 pallet with component COMP2 Level 3 - HU3.2 10 boxes with component COMP2 This public interface will return array with 2 handling units: - HU2.1 containing component COMP1 - HU2.2 containing component COMP2 Pre: NA Post: NA Input: iHandlingUnit - Handling Unit (mandatory) Output: oNumberHandlingUnits - Number of handling units in array. oHandlingUnitArray - Array with handling units. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

HandlingUnitStructure.GetInventoryLevel

DLL:	whextwmdapi
Syntax:	<pre> long HandlingUnitStructure.GetInventoryLevel(domain whhuid iHandlingUnit, ref domain tcmcs.long oNumberOfHandlingUnits, ref domain whhuid oHandlingUnitArray() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will search in the handling unit structure, starting at the iHandlingUnit for handling units which contain the inventory level. To explain this is more detail some examples will be given: Example 1: Item A_LOT is lot-controlled item. Top handling unit iHandlingUnit is multi-lot HU. First child HU_CHILD1 contains LOT1. Second child HU_CHILD2 contains LOT2. Function will return array with 2 handling units: - HU_CHILD1 - HU_CHILD2 Example 2: Item A_LOT is lot-controlled item. Top handling unit iHandlingUnit and both child handling units HU_CHILD1 and HU_CHILD2 contain the same lot LOT0. Function will return array with 1 handling unit HU_TOP. Example 3: Item A_LOT is lot-controlled and serialized item. Top handling unit iHandlingUnit and both child handling units HU_CHILD1 and HU_CHILD2 contain the same lot LOT0. Handling Unit HU_CHILD1 contains serial number SERIAL1. Handling Unit HU_CHILD2 contains serial number SERIAL2. Function will return array with 2 handling units: - HU_CHILD1 - HU_CHILD2 Pre: NA Post: NA Input: iHandlingUnit - Handling Unit (mandatory) Output: oNumberOfHandlingUnits - Number of handling units in array. oHandlingUnitArray - Array with handling units. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

HandlingUnitStructure.GetOrderLineLevel

DLL:	whextwmdapi
Syntax:	<pre> long HandlingUnitStructure.GetOrderLineLevel(domain whhuid iHandlingUnit, ref domain tcmcs.long oNumberOfHandlingUnits, ref domain whhuid oHandlingUnitArray() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will search in the handling unit structure, starting at the iHandlingUnit for handling units which contain the order line level. To explain this is more detail some examples will be given: Example 1: Top handling unit HU_TOP does not linked to any specific order line (e.g. receipt header handling unit) First child HU_CHILD1 is linked to the order line ORDER1/10. Second child HU_CHILD2 is linked to the order line ORDER1/20. Function will return array with 2 handling units: - HU_CHILD1 - HU_CHILD2 Example 2: Top handling unit HU_TOP and both child handling units HU_CHILD1 HU_CHILD2 are linked to the same order line ORDER2/10. Function will return array with 1 handling unit HU_TOP. Pre: NA Post: NA Input: iHandlingUnit - Handling Unit (mandatory) Output: oNumberHandlingUnits - Number of handling units in array. oHandlingUnitArray - Array with handling units. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

HandlingUnitStructure.GetWholeTree

DLL:	whextwmdapi
Syntax:	<pre> long HandlingUnitStructure.GetWholeTree(domain whhuid iHandlingUnit, ref domain tcmcs.long oNumberOfHandlingUnits, ref domain whhuid oHandlingUnitArray() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will search in the handling unit structure, starting at the iHandlingUnit for all handling units. To explain this is more detail an example will be given:

```

Handling Unit Structure has three levels:
Handling Unit Structure has three levels:
Level 1 - HU1                                1 Container
      Level 2 - HU2.1                        1 Pallet
            Level 3 - HU3.1                  2 Boxes
            Level 3 = HU3.2                  2 Boxes
      Level 2 - HU2.2                        1 Pallet
            Level 3 - HU3.3                  2 Boxes
            Level 3 - HU3.4                  2 Boxes

This public interface will return array with all handling units
in this order:
Position in array
Handling Unit
1      2      3      4      5      6      7
HU1, HU2.1, HU3.1, HU3.2, HU2.2, HU3.3, HU3.4

Pre:      NA
Post:     NA
Input:    iHandlingUnit      - Handling Unit (mandatory)
Output:   oNumberHandlingUnits - Number of handling units in array.
          oHandlingUnitArray  - Array with handling units.
          oExceptionMessage    - The last message if the return
                                value is not equal to 0.
                                If more than one message is
                                given, these are present in the
                                oExceptionID
          oExceptionID        - An ID that refers to all error
                                information. Use the functions in
                                Exception to get all relevant
                                information.

Return:   0: OK, <> 0: Error

```

Public Interfaces for InboundOrderLineActivity

The following functions are available:

InboundOrderLineActivity.ModifyAutomaticProcessing

InboundOrderLineActivity.ModifyAutomaticProcessing

DLL:	whextinhapi
Syntax:	<pre> long InboundOrderLineActivity.ModifyAutomaticProcessing(domain whinh.oorg iOrderOrigin, domain tcorno iOrderNumber, domain tcpono iOrderLine, domain tcpono iOrderSequence, domain whinh.prtv iProcedureType, ... </pre>

	domain tcmcs.st20m iActivity mb, domain tcyesno iAutomatic, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function will update the automatic attribute of the inbound order line activity that is provided. The automatic field will only be updated if the activity is applicable and the value of the automatic attribute is different than the value passed in variable iAutomatic. Depending on the order type, warehouse and inspection setup some activities are not applicable. For example the handling unit inbound advices are not applicable if the receiving warehouse is not location controlled. Pre: db.retry.point should be set Post: commit or abort transaction depending on the return value. Input: - iOrderOrigin - Mandatory - iOrderNumber - Mandatory - iOrderLine - Not Mandatory (inbound line can be 0) - iOrderSequence - Not Mandatory (inbound sequence can be 0) - iProcedureType - Mandatory - Allowed Values: - Receipt Procedure (whinh.prt.y.receipt.line) - Inspection Procedure (whinh.prt.y.receipt.inspect) - iActivity - Mandatory - Allowed Values: - The values are depending on the procedure type. Receipt Procedure: "whinh3412m100" - Print Goods Received Note "whinh3512m000" - Warehouse Receipts "whinh3201m000" - Generate Inbound Advice "whinh3415m000" - Generate Storage List "whinh3525m100" - Confirm Storage List "whinh3203m000" - Putaway Inbound Advice Inspection Procedure: "whinh3201m000" - Generate Inbound Advice "whinh3415m000" - Generate Storage List "whinh3525m100" - Confirm Storage List "whinh3203m000" - Putaway Inbound Advice Output: - iAutomatic - Mandatory - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant

	information.
	Return: 0: OK, <> 0: Error

Public Interfaces for InboundAdvice

The following functions are available:

[InboundAdvice.GetAvailableLocation](#)

[InboundAdvice.StartAutomaticInboundProcessing](#)

InboundAdvice.GetAvailableLocation

DLL:	whextinhapi
Syntax:	<pre> long InboundAdvice.GetAvailableLocation(domain tccwar iWarehouse, domain tcitem iItem, domain tcqstl iRequiredQuantity, domain tccuni iRequiredUnit, ref domain tcitem iMultiItemArray() fixed, ref domain tcqstl iMultiItemRequiredQuantityArray(), ref domain tccuni iMultiItemRequiredUnitArray() fixed, domain tcmcs.long iMultiItemNumberOfUnits, domain tcclot iLot, domain tccom.bpid iOwner, domain tcwght iRequiredWeight, domain tcleng iRequiredVolume, domain tcleng iRequiredFloor, domain tcleng iDepth, domain tcleng iWidth, domain tcleng iHeight, domain whwmd.pkdf iPackageDefinition, domain whinh.ittp iTransactionType, domain whwmd.loct iLocationType, boolean iIgnoreExpectedOccupationDecrease, boolean iAssignLocationToBusinessPartner, boolean iAllOnOneLocation, boolean iMustBeMultiItem, boolean iMustBeMultiLot, boolean iIncludeFixedLocations, boolean iMinimalPutAway, ref boolean oLocationsAvailable, ref long oNumberOfLocations, ref domain whloca oLocationsArray() fixed, ref domain tcqstl oQuantityToAdviseArray(), ref domain tcmcs.s999m oExceptionMessage mb, </pre>

	ref	long	oExceptionID)
Usage:	Expl:	This public interface will search for a location that is available. A location is available if it meets the requirements as specified by the input of this function, and if it passes the following possible constraints: - Location occupation - Multi/single item location - Multi/single lot location - Transaction (blocking) - Location ownership - Storage conditions - Location capacity When the goods to be stored do not fit into the one location, it will continue to search for other locations until all goods are stored. For this reason multiple locations can be returned by this function.	
	Pre:	oLocationsArray and oQuantityToAdviseArray must be declared as based variables, this function will allocate the memory.	
	Post:	After the oLocationsArray and oQuantityToAdviseArray are used, free the memory.	
	Input:	iWarehouse	- Warehouse - Mandatory
		iItem	- Item - Mandatory when iMultiItemNumberOfUnits is zero.
		iRequiredQuantity	- Quantity that must be advised. - Mandatory when iMultiItemNumberOfUnits is zero.
		iRequiredUnit	- Unit of iRequiredQuantity. - Mandatory when iMultiItemNumberOfUnits is zero.
		iMultiItemArray	- Array of Items. - Mandatory when iItem is not filled.
		iMultiItemRequiredQuantityArray	- Array of quantities that must be advised. - Mandatory when iItem is not filled.
		iMultiItemRequiredUnitArray	- Array of Unit of iRequiredQuantity. - Mandatory when iItem is not filled.
		iMultiItemNumberOfUnits	- The number of Items in the Multi Item array. - Mandatory when iItem is not filled.
		iLot	- Lot for which the location is searched for. - Not mandatory.
		iOwner	- Owner for which the location is searched for. - Not mandatory.
		iRequiredWeight	- Total weight of multi item array. - Not mandatory. - Value cannot be negative.
		iRequiredVolume	- Total volume of multi item array. - Not mandatory. - Value cannot be negative.
		iRequiredFloor	- Total floor space of multi item array. - Not mandatory. - Value cannot be negative.
		iDepth	- Required depth. - Not mandatory. - Value cannot be negative.

iWidth	- Required width. - Not mandatory.
iHeight	- Value cannot be negative. - Required height. - Not mandatory. - Value cannot be negative.
iPackageDefinition	- Package definition of iItem. - Not mandatory.
iTransactionType	- Transaction type. - Mandatory
iLocationType	- Location type to search for. - Mandatory
iIgnoreExpectedOccupationDecrease	- True: Expected occupation decrease is ignored in the location capacity calculation. - False: Expected occupation decrease is taken into account in the location capacity calculation. - Mandatory
iAssignLocationToBusinessPartner	- True: Only unoccupied locations are included in the search. When for the warehouse and item the ownership registration is done by location and iOwner is filled then only locations of that owner or without an owner are taken into account. - False: No checks on unoccupied locations or ownership. - Mandatory
iAllOnOneLocation	- True: The required quantity must fit on one location. - False: The required quantity does not need to fit on one location. completely - Mandatory
iMustBeMultiItem	- True: The location must be a multi item location. - False: The location does not need to be a multi item location. - Mandatory
iMustBeMultiLot	- True: The location must be a multi lot location. - False: The location does not need to be a multi lot location. - Mandatory
iIncludeFixedLocations	- True: Fixed locations are included in the search. - False: Fixed location are excluded from the search. - Mandatory
iMinimalPutAway	- True: Locations are sorted on available capacity to reduce the number locations as much as possible. - False: Locations are not sorted available capacity. - Mandatory
Output: oLocationsAvailable	- true: One or more Locations are found. - false: No locations are found.

	oNumberOfLocations - Number of available locations. oLocationsArray - Array of available locations. oQuantityToAdviseArray - Array of quantities that can be advised to this location in iRequiredUnit. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

InboundAdvice.StartAutomaticInboundProcessing

DLL:	whextinhapi
Syntax:	<pre>long InboundAdvice.StartAutomaticInboundProcessing(domain tcorno iInboundAdvice, domain tcpono iInboundAdviceLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface will start the processing of the next automatic activity after the inbound advice has been created. This public interface will be able to Putaway the Inbound Advice, Generate a Storage List or Putaway a Storage List when this is the next automatic activity of the related inbound order line. When the inbound advice is already putaway or the storage list is already putaway, this public interface will not perform any automatic step, as Inbound Inspections cannot be automated or the order line is already finished. This public interface will execute the following steps: 1) Clear any prior notifications done for the automatic inbound process, within the current process. 2) Notify the inbound advice for automatic processing 3) Start the automatic inbound process for the notified inbound advice The way to implement this public interface is to make sure that the next activity is set to automatic. This can be already set properly based on the order type setup, or can be achieved by using public interface InboundOrderLineActivity.ModifyAutomaticProcessing() for the activity that should be automatically processed. See below code example: <pre>--- db.retry.point() /* Set the activity to automatic ret = InboundOrderLineActivity.ModifyAutomaticProcessing(..)</pre>

```

        if ret <> 0 then
            abort.transaction()
            return(ret)
        endif

        commit.transaction()

        /* Start the automatic order process
        ret.val = InboundAdvice.StartAutomaticInboundProcessing(..)

        db.retry.point()

        /* Set the activity to be not automatic
        ret = InboundOrderLineActivity.ModifyAutomaticProcessing(..)

        if ret <> 0 then
            abort.transaction()
            return(ret)
        endif

        commit.transaction()

        return(ret.val)
    ---

```

This public interface will be able to putaway the inbound advice, generate a storage list or putaway the storage list. This public interface will stop the automatic process when the next activity is not set to automatic.

This function must not be called within a logical transaction as this function will have it's own transaction handling.

Pre: There should be no pending logical transaction before calling this function.

Post: No need to commit or abort the process, that is handled within the function.

Input: iInboundAdvice - Mandatory
iInboundAdviceLine - Mandatory

Output: oExceptionMessage - The last message if the return value is not equal to 0.
If more than one message is given, these are present in the oExceptionID

oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information.

Return: 0: OK, <> 0: Error

Public Interfaces for InboundInspection

The following functions are available:

[InboundInspection.StartAutomaticInboundProcessing](#)

InboundInspection.StartAutomaticInboundProcessing

DLL:	whextinhapi
Syntax:	<pre> long InboundInspection.StartAutomaticInboundProcessing(domain tcorno iInspection, domain tcpono iInspectionSequence, ref tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will start the processing of the next automatic activity after the warehouse inspection has been processed. This public interface will generate inbound advices when this is the next automatic activity of the related inbound order line. When there is already an advice present which must be putaway, please use one of the following public interfaces: InboundAdvice.StartAutomaticInboundProcessing StorageList.StartAutomaticInboundProcessing InboundRun.StartAutomaticInboundProcessing This public interface will execute the following steps: 1) Clear any prior notifications done for the automatic inbound process, within the current process. 2) Notify the Inbound Inspection for automatic processing 3) Start the automatic inbound process for the notified Inbound Inspection The way to implement this public interface is to make sure that the next activity is set to automatic. This can be already set properly based on the order type setup, or can be achieved by using public interface InboundOrderLineActivity.ModifyAutomaticProcessing() for the activity that should be automatically processed. See below code example: <pre> --- db.retry.point() /* Set the activity to automatic ret = InboundOrderLineActivity.ModifyAutomaticProcessing(..) if ret <> 0 then abort.transaction() return(ret) endif commit.transaction() /* Start the automatic order process ret.val = InboundInspection.StartAutomaticInboundProcessing(..) db.retry.point() </pre>

	<pre> * Set the activity to be not automatic ret = InboundOrderLineActivity.ModifyAutomaticProcessing(..) if ret <> 0 then abort.transaction() return(ret) endif commit.transaction() return(ret.val) ---</pre> This public interface will be able to automatically generate an inbound advice for the given warehouse inspection. This public interface will stop the automatic process when the next activity is not set to automatic. This function must not be called within a logical transaction as this function will have it's own transaction handling. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iInspection - Mandatory iInspectionSequence - Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Public Interfaces for InboundOrderLine

The following functions are available:

[InboundOrderLine.PrintGoodsReceivedNote](#)

[InboundOrderLine.Receive](#)

[InboundOrderLine.StartAutomaticInboundProcessing](#)

[InboundOrderLine.StartPrintGoodsReceivedNote](#)

InboundOrderLine.PrintGoodsReceivedNote

DLL:	whextinhapi															
	This function is available from KB2322785 .															
Syntax:	<pre>long InboundOrderLine.PrintGoodsReceivedNote (domain whinh.oorg iOrderOrigin, domain tcorno iOrderNumber, domain tcpono iOrderLine, domain tcpono iOrderSequence, domain tcmcs.str15 iDevice, long iProcessingOptionSet, ref boolean oDataProcessed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>															
Usage:	Expl	This public interface will print the goods received note using the defaults or options as provided in the iProcessingOptionSet. Selection range fields (From/To) provided in the iProcessingOptionSet are ignored when iOrderNumber is filled. In that case the goods received note is printed for the given iOrderOrigin, iOrderNumber, iOrderLine and iOrderSequence.														
	Pre:	By default the transaction handling is done within this function, so there should be no pending logical transaction before calling this function. This is not the case when IncludeTransactionHandling is set to False. The report must be opened for the Idevice.														
	Post:	By default the transaction handling is done within this function, so there is No need to commit or abort the process. that is handled within the function. When IncludeTransactionHandling is set to False then the calling process has to handle abort/commit of the process. The report must be closed for the Idevice.														
	Input:	<table><tr><td>iOrderOrigin</td><td>Mandatory</td></tr><tr><td>iOrderNumber</td><td>Optional</td></tr><tr><td>iOrderLine</td><td>Optional</td></tr><tr><td>iOrderSequence</td><td>Optional</td></tr><tr><td>iDevice</td><td>Mandatory</td></tr><tr><td>iProcessingOptionSet</td><td>Optional, if 0, the default options are applied.</td></tr></table>	iOrderOrigin	Mandatory	iOrderNumber	Optional	iOrderLine	Optional	iOrderSequence	Optional	iDevice	Mandatory	iProcessingOptionSet	Optional, if 0, the default options are applied.	A Processing Option Set can be created via a call to function ProcessingOptionSet.Create().	
iOrderOrigin	Mandatory															
iOrderNumber	Optional															
iOrderLine	Optional															
iOrderSequence	Optional															
iDevice	Mandatory															
iProcessingOptionSet	Optional, if 0, the default options are applied.															
		Processing Options have a direct relationship with the form fields on session Print Goods Received Note (whinh3412m100) and are not explained in further detail here. Please refer to the session help for additional information. Processing Options that are set while a required Implemented Software Component is not available are ignored.														
		Explanation about setting of defaults: Minimum Value: Minimum value of domain is taken as default value. Maximum Value: Maximum value of domain is taken as default value.														
		<table><tr><th>NAME</th><th>TYPE</th><th>DEFAULT</th></tr><tr><td>ReceiptDateFrom</td><td>domain tcdater</td><td>Minimum Value</td></tr><tr><td>ReceiptDateTo</td><td>domain tcdater</td><td>Maximum Value</td></tr><tr><td>OrderOriginFrom</td><td>domain whinh.oorg</td><td>Minimum Value</td></tr></table>	NAME	TYPE	DEFAULT	ReceiptDateFrom	domain tcdater	Minimum Value	ReceiptDateTo	domain tcdater	Maximum Value	OrderOriginFrom	domain whinh.oorg	Minimum Value		
NAME	TYPE	DEFAULT														
ReceiptDateFrom	domain tcdater	Minimum Value														
ReceiptDateTo	domain tcdater	Maximum Value														
OrderOriginFrom	domain whinh.oorg	Minimum Value														

OrderOriginTo	domain whinh.oorg	Maximum Value
OrderNumberFrom	domain tcorno	Minimum Value
OrderNumberTo	domain tcorno	Maximum Value
OrderSetFrom	domain tcwset	Minimum Value
OrderSetTo	domain tcwset	Maximum Value
OrderLineFrom	domain tcpono	Minimum Value
OrderLineTo	domain tcpono	Maximum Value
ReceiptFrom	domain whinh.shpm	Minimum Value
ReceiptTo	domain whinh.shpm	Maximum Value
WarehouseFrom	domain tccwar	Minimum Value
WarehouseTo	domain tccwar	Maximum Value
ShipFromBusinessPartnerFrom	domain tccom.bpid	Minimum Value
ShipFromBusinessPartnerTo	domain tccom.bpid	Maximum Value
ItemFrom	domain tcitem	Minimum Value
ItemTo	domain tcitem	Maximum Value
InOwnLanguage	domain tcyesno	tcyesno.yes
AllOrders	domain tcyesno	tcyesno.no
SortByOrderandWarehouse	domain tcyesno	tcyesno.no
PrintLotsAndSerials	domain tcyesno	tcyesno.no
PrintHandlingUnits	domain tcyesno	tcyesno.no
PrintSpecification	domain tcyesno	tcyesno.no
PrintSpaceForNotes	domain tcyesno	tcyesno.no
PrintSubcontractingData	domain tcyesno	tcyesno.yes
PrintVariantOptionDescriptions	domain tcyesno	tcyesno.no
PrintStoragezone	domain tcyesno	tcyesno.no
NumberOfCopies	domain tcsrno	0
ReprintAlreadyPrintedOrders	domain tcyesno	tcyesno.no
RePrintAllOpenLines	domain tcyesno	tcyesno.no
RePrintIncludeReceivedOrderLines	domain tcyesno	tcyesno.no
IncludeTransactionHandling	boolean	True
Output: o.data.processed - true: Goods Received Note Printed. false: Nothing Printed. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.		

InboundOrderLine.Receive

DLL:	whextinhapi
Syntax:	<pre> long InboundOrderLine.Receive(domain whinh.oorg iOrderOrigin, domain tcorno iOrderNumber, domain tcpono iOrderLine, domain tcpono iOrderSequence, domain whinh.shpm iReceipt, domain tcqstl iReceivedQuantity, domain tccuni iReceivedUnit, </pre>

	domain tcpksp iPackingSlip mb, domain tcqstl iPackingSlipQuantity, domain tccuni iPackingSlipUnit, domain whinh.load iLoad, domain whinh.shpm iShipment, domain tcclot iLot, iGenerateLot, iGenerateSerial, ref domain whinh.shpm oReceipt, ref long oNumberOfReceiptLines, ref domain tcpono oReceiptLineArray(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This public interface will allow an inbound line to be received, making one or multiple new receipt lines. When the iReceipt is passed as empty, a new Receipt Header will be created. There can be multiple receipt lines generated based on the setup related to Receipt Line Consolidation in combination with the stock points that are to be received. This function will also have the option to automatically generate lot and/or serial numbers during the receipt line creation. Pre: db.retry.point must be set oReceiptLineArray must be declared as a based variable, this function will allocate the memory. Post: Commit the transaction in case of success Abort the transaction in case of failure After the oReceiptLineArray is used, free the memory. Input: Start Order Line Key Fields - iOrderOrigin - Order Origin - Mandatory - iOrderNumber - Order Number - Mandatory - iOrderLine - Order Line - Not Mandatory - iOrderSequence - Order Sequence - Not Mandatory End Order Line Key Fiels - The order line should exist and will be validated prior to receiving is done. iReceipt - Receipt header on which lines must be added. Not mandatory, if empty a new receipt header will be created. - iReceivedQuantity - Received Quantity - Mandatory - iReceivedUnit - Received Unit - Mandatory - iPackingSlip - Packing Slip - Not Mandatory - iPackingSlipQuantity - Packing Slip Quantity - Not Mandatory - iPackingSlipUnit - Packing Slip Unit - Mandatory when iPackingSlipQuantity is filled - iLoad - Load - Not Mandatory - iShipment - Shipment - Not Mandatory - iLot - Received Lot - Not Mandatory - iGenerateLot - Generate Lot when item is lot controlled - Mandatory - iGenerateSerial - Generate Serial when item is serialized - Mandatory Output: oReceipt - Receipt that is created (or populated with the iReceipt when iReceipt is filled) - oNumberOfReceiptLines - Number of receipt lines that are created. - oReceiptLineArray - Receipt lines that are created.

	<code>oExceptionMessage</code> - The last message if the return value is not equal to 0. If more than one message is given, these are present in the <code>oExceptionID</code> <code>oExceptionID</code> - An ID that refers to all error information. Use the functions in <code>Exception</code> to get all relevant information. Return: 0: OK, <> 0: Error

InboundOrderLine.StartAutomaticInboundProcessing

DLL:	whextinhapi
Syntax:	<pre>long InboundOrderLine.StartAutomaticInboundProcessing(domain whinh.oorg iOrderOrigin, domain tcorno iOrderNumber, domain tcpono iOrderLine, domain tcpono iOrderSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This Public Interface can be used to start the automatic inbound process for the given inbound order line. This public interface will allow to start the automatic process when one of following activities for the inbound order line has been set to automatic: - Goods Received Note - Warehouse Receipt This public interface will execute the following steps: 1) Clear any prior notifications done for the automatic inbound process, within the current process. 2) Notify the inbound order line for automatic processing 3) Start the automatic inbound process for the notified Inbound Order Line The way to implement this public interface is to make sure that the next activity is set to automatic. This can be already set properly based on the order type setup, or can be achieved by using public interface <code>InboundOrderLineActivity.ModifyAutomaticProcessing()</code> for the activity that should be automatically processed. See below code example: <pre>--- db.retry.point() /* Set the activity to automatic ret = InboundOrderLineActivity.ModifyAutomaticProcessing(..) if ret <> 0 then abort.transaction() return(ret) endif</pre>

	<pre> commit.transaction() * Start the automatic order process ret.val = InboundOrderLine.StartAutomaticInboundProcessing(..) db.retry.point() * Set the activity to be not automatic ret = InboundOrderLineActivity.ModifyAutomaticProcessing(..) if ret <> 0 then abort.transaction() return(ret) endif commit.transaction() return(ret.val) ---</pre> This public interface will stop the automatic process when the next activity is not set to automatic. This only applies when either the Goods Received Note or the Warehouse Receipt activity have been handled by this public interface. When the order line is already received, and for the inbound line there is no Goods Received Note or Warehouse Receipt to be done, this public interface will not execute the next automatic step. The next steps can be executed by one of the following public interfaces: <pre> ReceiptLine.StartAutomaticInboundProcess InboundInspection.StartAutomaticInboundProcess InboundAdvice.StartAutomaticInboundProcess InboundRun.StartAutomaticInboundProcess StorageList.StartAutomaticInboundProcess HandlingUnit.StartAutomaticInboundProcess QuarantineInventory.StartAutomaticInboundProcess</pre> This function must not be called within a logical transaction as this function will have it's own transaction handling. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: <table> <tr> <td>iOrderOrigin</td><td>- Mandatory</td></tr> <tr> <td>iOrderNumber</td><td>- Mandatory</td></tr> <tr> <td>iOrderLine</td><td>- Not Mandatory</td></tr> <tr> <td>iOrderSequence</td><td>- Not Mandatory</td></tr> </table> Output: <table> <tr> <td>oExceptionMessage</td><td>- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID</td></tr> <tr> <td>oExceptionID</td><td>- An ID that refers to all error information. Use the functions in Exception to get all relevant information.</td></tr> </table> Return: 0: OK, <> 0: Error	iOrderOrigin	- Mandatory	iOrderNumber	- Mandatory	iOrderLine	- Not Mandatory	iOrderSequence	- Not Mandatory	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
iOrderOrigin	- Mandatory												
iOrderNumber	- Mandatory												
iOrderLine	- Not Mandatory												
iOrderSequence	- Not Mandatory												
oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID												
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.												

InboundOrderLine.StartPrintGoodsReceivedNote

DLL:	whextinhapi This function is available from KB2314556 .
Syntax:	<pre> long InboundOrderLine.StartPrintGoodsReceivedNote(long iStartMode, boolean iIgnoreSelectionFields, domain tcdatetime iFromPlannedReceiptDate, domain whinh.oorg iFromOrderOrigin, domain tcorno iFromOrderNumber, domain tcpono iFromOrderLine, domain whinh.shpm iFromReceipt, domain tcdatetime iToPlannedReceiptDate, domain whinh.oorg iToOrderOrigin, domain tcorno iToOrderNumber, domain tcpono iToOrderLine, domain whinh.shpm iToReceipt, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl This function starts the session Print Goods Received Note (whinh3412m100). Depending on the main table of the calling session, Non-Consecutive Record Selection (NCRS) is used. When the main table is: - Warehouse Orders (whinh200) or - Inbound Order Lines (whinh210) or - Handling Unit Process Inbound (whinh213") or - Receipt Lines (whinh312) or - Handling Units (whwmd530) then the range selections will be imported based on the selected records in the parent session and the selection related fields will be ignored. Input: - iStartMode Not Used. - iIgnoreSelectionFields If true, From/To selection fields will not be filled in the session. - iFromPlannedReceiptDate From Planned Receipt Date selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) - iFromOrderOrigin From Order Origin selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) - iFromOrderNumber From Order Number selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) - iFromOrderLine From Order Line selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) - iFromReceipt From Receipt selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) - iToPlannedReceiptDate To Planned Receipt Date selection field is filled with this value. (when iIgnoreSelectionFields is false

	and NCRS is not applicable) iToOrderOrigin To Order Origin selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToOrderNumber To Order Number selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToOrderLine To Order Line selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToReceipt To Receipt selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

Public Interfaces for InboundRun

The following functions are available:

[InboundRun.GenerateStorageList](#)

[InboundRun.StartAutomaticInboundProcessing](#)

InboundRun.GenerateStorageList

DLL:	whextinhapi
	This function is available from KB2304787 .
Syntax:	<pre>long InboundRun.GenerateStorageList(domain whinh.btno iInboundRun, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface will generate a storage list for the

	<pre> inbound run. No data is printed. Be aware that transaction management is handled within this function. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iInboundRun - Inbound run (Mandatory) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error </pre>

InboundRun.StartAutomaticInboundProcessing

DLL:	whextinhapi
Syntax:	<pre> long InboundRun.StartAutomaticInboundProcessing(domain whinh.btno iInboundRun, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will start the processing of the next automatic activity after the for the notified Run. This public interface will handle the next automatic inbound activity for the given inbound run. Please be aware that multiple inbound advice lines and/or inbound order lines could be processed when starting the automatic inbound process for the run, especially when runs are reused. This public interface will execute the following steps: 1) Clear any prior notifications done for the automatic inbound process, within the current process. 2) Notify the run for automatic processing 3) Start the automatic inbound process for the notified run The way to implement this public interface is to make sure that the next activity is set to automatic. This can be already set properly based on the order type setup, or can be achieved by using public interface InboundOrderLineActivity.ModifyAutomaticProcessing() for the activity that should be automatically processed. See below code example: <pre> --- db.retry.point() /* Set the activity to automatic ret = InboundOrderLineActivity.ModifyAutomaticProcessing(..) </pre>

```

if ret <> 0 then
    abort.transaction()
    return(ret)
endif

commit.transaction()

|* Start the automatic order process
ret.val = InboundRun.StartAutomaticInboundProcessing(..)

db.retry.point()

|* Set the activity to be not automatic
ret = InboundOrderLineActivity.ModifyAutomaticProcessing(..)

if ret <> 0 then
    abort.transaction()
    return(ret)
endif

commit.transaction()

return(ret.val)
---
```

This public interface will be able to automatically perform the next automatic steps for the given run. This public interface will stop the automatic process when the next activity is not set to automatic.

This function must not be called within a logical transaction as this function will have it's own transaction handling.

Pre: There should be no pending logical transaction before calling this function.

Post: No need to commit or abort the process, that is handled within the function.

Input: iInboundRun - Mandatory

Output: oExceptionMessage - The last message if the return value is not equal to 0.
If more than one message is given, these are present in the oExceptionID

oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information.

Return: 0: OK, <> 0: Error

Public Interfaces for ProjectCostPegTransfer

The following functions are available:

[ProjectCostPegTransfer.CheckAllowedBetweenPegs](#)[ProjectCostPegTransfer.CheckInventoryAvailable](#)[ProjectCostPegTransfer.Generate](#)[ProjectCostPegTransfer.StartDetail](#)[ProjectCostPegTransfer.StartMultiMain](#)[ProjectCostPegTransfer.StartOverview](#)

ProjectCostPegTransfer.CheckAllowedBetweenPegs

DLL:	whextinhapi
	This function is available from KB2304789 .
Syntax:	<pre> long ProjectCostPegTransfer.CheckAllowedBetweenPegs (domain tccprj iProjectFrom, domain tcpdm.cspa iElementFrom, domain tcpdm.cact iActivityFrom, domain tccprj iProjectTo, domain tcpdm.cspa iElementTo, domain tcpdm.cact iActivityTo, domain tcorno iCostPegTransfer, domain tcpono iCostPegTransferLine, domain whinh.cpto iOrigin, domain whinh.cptt iTransferType, ref boolean oTransferAllowed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface checks if cost peg transfers are allowed between the given pegs. At least one of the arguments iProjectFrom or iProjectTo has to be filled when calling this Public Interface. Pre: N.a. Post: N.a. Input: iProjectFrom - Project from iElementFrom - Element from iActivityFrom - Activity from iProjectTo - Project to iElementTo - Element to iActivityTo - Activity to iCostPegTransfer - Cost Peg Transfer iCostPegTransferLine - Cost Peg Transfer Line iOrigin - Origin iTransferType - Transfer Type Output: oTransferAllowed - Transfer Allowed Indicator oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Cost Peg transfer is allowed between the pegs

	<> 0 - Error

ProjectCostPegTransfer.CheckInventoryAvailable

DLL:	whextwmdapi
	This function is available from KB2304786 .
Syntax:	<pre> long ProjectCostPegTransfer.CheckInventoryAvailable(domain tcitem iItem, domain tcuef,effn iEffectivityUnit, domain tccwar iWarehouse, domain tccprj iProject, domain tcpdm.cspa iElement, domain tcpdm.cact iActivity, domain tcowns iOwnership, domain tcqivl iTransferQuantity, domain whinh.cptt iTransferType, ref domain whinh.cptt oTransferType, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface checks if there is enough inventory available to transfer a cost peg. Pre: N.a. Post: N.a. Input: iItem - Item (mandatory) iEffectivityUnit - Effectivity Unit iWarehouse - Warehouse (mandatory) iProject - Project iElement - Element iActivity - Activity iOwnership - Ownership iTransferQuantity - Transfer Quantity iTransferType - Cost peg transfer type Output: oTransferType - Cost peg transfer type, switched if only 'Permanent' is allowed. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Inventory is available to be transferred <> 0 - Error

ProjectCostPegTransfer.Generate

DLL:	whextinhapi

	This function is available from KB2286749 .
Syntax:	<pre> long ProjectCostPegTransfer.Generate(domain tcseri iSeries, domain tccwar iWarehouse, domain tcitem iItem, domain tcuef.effn iEffectivityUnit, domain tcowns iOwnership, domain tccprj iFromProject, domain tcpdm.cspa iFromElement, domain tcpdm.cact iFromActivity, domain tccprj iToProject, domain tcpdm.cspa iToElement, domain tcpdm.cact iToActivity, domain whinh.cptt iTransferType, domain tcqivl iQuantity, domain tcdate iRequirementDate, domain tccdis iReason, domain tcyesno iDirectProcess, ref domain tcorno oProjectCostPegTransfer, ref domain tcpono oProjectCostPegTransferLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function generates a Project Cost Peg Transfer header and line. Optionally, the cost peg transfer can be processed directly. Only allowed when the Project Pegging concept is enabled in Implemented Software Components (tccom0500m000). Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iSeries Series used to generate the Project Cost Peg Transfer. Must exist for the Number Group defined for Project Cost Peg Transfers in Inventory Handling Parameters. If left empty, the series defined in the User Profiles, Warehouse Settings by Site or Inventory Handling Parameters are used iWarehouse (mandatory) The warehouse where the project cost peg is transferred iItem (mandatory) The item of the project cost peg transfer line iEffectivityUnit The Effectivity Unit. Only applicable if the item is lot controlled (in inventory). iOwnership (mandatory) The ownership of the project pegged inventory to be transferred Only value Company Owned or Customer Owned is allowed. In case of Customer Owned, the owner of the iFromProject (or iToProject) is put on the cost peg transfer line. iFromProject The Project from which the cost pegs are transferred. iFromElement The Element linked to the iFromProject. Is mandatory if cost control level by element is active for the iFromProject iFromActivity The Activity linked to the iFromProject Is mandatory if cost control level by activity is active for the iFromProject iToProject The Project to which the project cost peg is transferred

	Is mandatory if the iItem is mandatory pegged iToElement The Element linked to the iToProject Is mandatory if cost control level by element is active for the iToProject iToActivity The Activity linked to the iToProject Is mandatory if cost control level by activity is active for the iToProject iTransferType (mandatory) The project cost peg transfer type. Allowed values: - Permanent: The loaning project is compensated for the cost of the material. New demand orders are created to replenish it - Borrow/Loan: The borrowing project must pay back the borrowed inventory to the lending project before the next billing cycle. No cost is transferred between the projects. This value is only allowed if: - iOwnership is Company Owned and - Both iFromProject as iToProject is filled and - Manual Borrow/Loan Transfers are allowed according to the Project Pegging Parameters (tcpeg0100m000) iQuantity (mandatory) The quantity to be transferred (in inventory unit) iRequirementDate The date on which the target peg requires the goods. If left empty, the current date is used. iReason (mandatory) The reason code of the project cost peg transfer. Must be of type 'Cost Peg Transfer' iDirectProcess (mandatory) Directly process the generated cost peg transfer(Yes/No) Output: oProjectCostPegTransfer The generated project cost peg transfer number oProjectCostPegTransferLine The generated project cost peg transfer line oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK <> 0: Error

ProjectCostPegTransfer.StartDetail

DLL:	whextinhapi
	This function is available from KB2296399 .
Syntax:	long ProjectCostPegTransfer.StartDetail(long iStartMode,

	<pre> domain tcorno iProjectCostPegTransfer, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the detail session Project Cost Peg Transfer (whinh1140m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variable form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iProjectCostPegTransfer - The primary key fields must refer to an existing Project Cost Peg Transfer Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

ProjectCostPegTransfer.StartMultiMain

DLL:	whextinhapi
	This function is available from KB2296399 .
Syntax:	<pre> long ProjectCostPegTransfer.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iProjectCostPegTransfer, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the multi main table session Project Cost Peg Transfer (whinh1640m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session.

	MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend A specific query to be used when zooming to this session. iProjectCostPegTransfer Optional Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

ProjectCostPegTransfer.StartOverview

DLL:	whextinhapi
	This function is available from KB2296399 .
Syntax:	<pre>long ProjectCostPegTransfer.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iProjectCostPegTransfer, ref domain tcorno oProjectCostPegTransfer, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl This function starts the overview session Project Cost Peg Transfer (whinh1140m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend A specific query to be used when zooming to this session. iProjectCostPegTransfer The Project Cost Peg Transfer to be started

	Mandatory if iStartMode = MODELESS Output: for iStartMode MODAL: oProjectCostPegTransfer - Project Cost Peg Transfer of selected record oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

Public Interfaces for ProjectCostPegTransferLine

The following functions are available:

[ProjectCostPegTransferLine.Process](#)

[ProjectCostPegTransferLine.StartDetail](#)

[ProjectCostPegTransferLine.StartOverview](#)

ProjectCostPegTransferLine.Process

DLL:	whextinhapi
	This function is available from KB2305700 .
Syntax:	<pre>long ProjectCostPegTransferLine.Process(domain tcorno iCostPegTransfer, domain tcpono iCostPegTransferLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function processes the cost peg transfer line. It is only allowed when the Project Pegging concept is enabled in Implemented Software Components (tccom0500m000). Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iCostPegTransfer - Project Cost Peg Transfer (Mandatory) iCostPegTransferLine - Project Cost Peg Transfer Line (Mandatory) Output: NA Return: 0: OK <> 0: Error

ProjectCostPegTransferLine.StartDetail

DLL:	whextinhapi
	This function is available from KB2296385 .
Syntax:	<pre> long ProjectCostPegTransferLine.StartDetail(long iStartMode, domain tcorno iProjectCostPegTransfer, domain tcpono iProjectCostPegTransferLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the detail session Project Cost Peg Transfer Lines (whinh1145m000). Input: <code>iStartMode</code> Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the <code>oExceptionMessage</code> and the session will not be started. Primary Key Fields: <code>iProjectCostPegTransfer</code> <code>iProjectCostPegTransferLine</code> - The primary key fields must refer to an existing Project Cost Peg Transfer Line. Output: <code>oExceptionMessage</code> - The last message if any message is found. If more than one message is given, these are present in the <code>oExceptionID</code>. <code>oExceptionID</code> - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - Session started - Error.

ProjectCostPegTransferLine.StartOverview

DLL:	whextinhapi

	This function is available from KB2296385 .	
Syntax:	<pre> long ProjectCostPegTransferLine.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iProjectCostPegTransfer, domain tcpono iProjectCostPegTransferLine, domain whinh.oorg iOrderOrigin, domain tcorno iOrderNumber, domain tcwset iOrderSet, domain tcpono iOrderLine, domain tcpono iOrderSequence, domain tcpono iAdvice, domain tcitem iItem, domain tccprj iFromProject, domain tcpdm.cspa iFromElement, domain tcpdm.cact iFromActivity, domain tccprj iToProject, domain tcpdm.cspa iToElement, domain tcpdm.cact iToActivity, ref domain tcorno oProjectCostPegTransfer, ref domain tcpono oProjectCostPegTransferLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>	
Usage:	Expl This function starts the overview session Project Cost Peg Transfer Lines (whinh1145m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byOrderLine": Project Cost Peg Transfer Lines are displayed by project cost peg transfer. session will be started on index 1 view fields: Project cost peg transfer "byRelatedOrderLine": Project Cost Peg Transfer Lines are displayed by related order line. session will be started on index 2 view fields: Order Origin Order Number Order Set Order Line "byItem": Project Cost Peg Transfer Lines are displayed by item. session will be started on index 3 view fields: Item "byFromPeg": Project Cost Peg Transfer Lines are displayed by from peg.	

	session will be started on index 4 view fields: From Project "byToPeg": Project Cost Peg Transfer Lines are displayed by to peg. session will be started on index 5 view fields: To Project iSessionIndex Specifies the table index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iProjectCostPegTransfer The Project Cost Peg Transfer to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 1 (or iStartFilter = "byOrderLine") iProjectCostPegTransferLine The Project Cost Peg Transfer Line to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 1 (or iStartFilter = "byOrderLine") iOrderOrigin The Order Origin to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 2 (or iStartFilter = "byRelatedOrderLine") iOrderNumber The Order Number to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 2 (or iStartFilter = "byRelatedOrderLine") iOrderSet The Order Set to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 2 (or iStartFilter = "byRelatedOrderLine") iOrderLine The Order Line to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 2 (or iStartFilter = "byRelatedOrderLine") iOrderSequence Optional iAdvice Optional iItem The Item to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 3 (or iStartFilter = "byItem") iFromProject The From Project to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 4 (or iStartFilter = "byFromPeg") iFromElement Optional iFromActivity Optional iToProject The To Project to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 5 (or iStartFilter = "byToPeg") iToElement Optional iToActivity Optional

	<pre> Output: for iStartMode MODAL: oProjectCostPegTransfer - - Project Cost Peg Transfer of selected record oProjectCostPegTransferLine - Project Cost Peg Transfer Line of selected record oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - Session started - Error. </pre>

Public Interfaces for AllocationChangeOrder

The following functions are available:

[AllocationChangeOrder.Generate](#)

[AllocationChangeOrder.Process](#)

AllocationChangeOrder.Generate

DLL:	whextinhapi
	This function is available from KB2278026 .
Syntax:	<pre> long AllocationChangeOrder.Generate(domain tccwar iWarehouse, domain tccdis iReason, domain tcseri iSeries, domain tcyesno iDirectlyProcess, boolean iSetHeaderText, domain tctxtn iHeaderText, boolean iSetItem, domain tcitem iItem, boolean iSetEffectivityUnit, domain tcuef.effn iEffectivityUnit, boolean iSetHandlingUnit, domain whhuid iHandlingUnit, boolean iSetOrderQuantity, domain tcqstl iOrderQuantity, </pre>

		boolean	iSetOrderUnit,
	domain	tccuni	iOrderUnit,
	boolean		iSetSoldToBusinessPartnerFrom,
	domain	tccom.bpid	iSoldToBusinessPartnerFrom,
	boolean		iSetShipToBusinessPartnerFrom,
	domain	tccom.bpid	iShipToBusinessPartnerFrom,
	boolean		iSetBusinessObjectTypeFrom,
	domain	tcalbt	iBusinessObjectTypeFrom,
	boolean		iSetBusinessObjectFrom,
	domain	tcboid	iBusinessObjectFrom,
	boolean		iSetBusinessObjectReferenceFrom,
	domain	tcborf	iBusinessObjectReferenceFrom,
	boolean		iSetReferenceFrom,
	domain	tcrefa	iReferenceFrom mb,
	boolean		iSetSoldToBusinessPartnerTo,
	domain	tccom.bpid	iSoldToBusinessPartnerTo,
	boolean		iSetShipToBusinessPartnerTo,
	domain	tccom.bpid	iShipToBusinessPartnerTo,
	boolean		iSetBusinessObjectTypeTo,
	domain	tcalbt	iBusinessObjectTypeTo,
	boolean		iSetBusinessObjectTo,
	domain	tcboid	iBusinessObjectTo,
	boolean		iSetBusinessObjectReferenceTo,
	domain	tcborf	iBusinessObjectReferenceTo,
	boolean		iSetReferenceTo,
	domain	tcrefa	iReferenceTo mb,
	boolean		iSetLineText,
	domain	tctxtn	iLineText,
ref	domain	tcorno	oAllocationChangeOrder,
ref	domain	tcmcs.s999m	oExceptionMessage mb,
ref	long		oExceptionID)
Usage:	Expl: This function generates an Allocation Change Order header and line. Optionally, the allocation change order can be processed directly. Only allowed when the Demand Pegging concept is enabled in Implemented Software Components (tccom0500m000). Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iWarehouse (mandatory) The warehouse where the inventory allocation must change iReason (mandatory) The reason code of the allocation change. Must be of type 'Allocation Change' iSeries The series used for allocation change order generation. Must exist for the Number Group defined for allocation change orders in Inventory Handling Parameters. If left empty, the series defined in the User Profiles, Warehouse Settings by Site or Inventory Handling Parameters are used iDirectlyProcess (mandatory) Directly process the generated allocation change (Yes/No) iSetHeaderText Set text for allocation change order header (True/False) iHeaderText Allocation change order header text number iSetItem Set Item (True/False) iItem The Item whose allocation is to change. Is mandatory if the iHandlingUnit is empty If iHandlingUnit is filled, this field is derived from		

	the handling unit. iSetEffectivityUnit Set Effectivity Unit (True/False) iEffectivityUnit Effectivity Unit. Only applicable if the item is lot controlled (in inventory). iSetHandlingUnit Set Handling Unit (True/False) iHandlingUnit The Handling Unit that contains inventory whose allocation is to change. Is mandatory if the Allocation Registration Level of the item by warehouse is by 'Physical Item' iSetOrderQuantity (mandatory if iHandlingUnit is empty) Set Order Quantity (True/False) iOrderQuantity (mandatory if iHandlingUnit is empty) Order Quantity in order unit. Value must be > 0. If handling unit is filled, this field is derived from the handling unit. iSetOrderUnit (mandatory if iHandlingUnit is empty) Set Order Unit (True/False) iOrderUnit (mandatory if iHandlingUnit is empty) Order Unit If iHandlingUnit is filled, this field is derived from the handling unit. iSetSoldToBusinessPartnerFrom Set allocated Sold-to Business partner - from (true/false) iSoldToBusinessPartnerFrom The sold-to business partner to which the inventory is allocated before the allocation change order is processed iSetShipToBusinessPartnerFrom Set allocated Ship-to Business partner - from (true/false) iShipToBusinessPartnerFrom The ship-to business partner to which the inventory is allocated before the allocation change order is processed iSetBusinessObjectTypeFrom Set allocated Business Object Type - from (true/false) iBusinessObjectTypeFrom The business object type (Order type) to which the inventory is allocated before the allocation change order is processed iSetBusinessObjectFrom Set allocated Business Object - from (true/false) iBusinessObjectFrom The business object (Order) to which the inventory is allocated before the allocation change order is processed iSetBusinessObjectReferenceFrom Set allocated business object reference - from (true/false) iBusinessObjectReferenceFrom The business object reference (Order line) to which the inventory is allocated before the allocation change order is processed iSetReferenceFrom Set Reference - from (true/false) iReferenceFrom A reference code to which the inventory was allocated before the allocation change order is processed iSetSoldToBusinessPartnerTo Set allocated Sold-to Business partner - to (true/false) iSoldToBusinessPartnerTo The sold-to business partner to which the inventory is allocated after the allocation change order is processed

	<pre> iSetShipToBusinessPartnerTo Set allocated Ship-to Business partner - to (true/false) iShipToBusinessPartnerTo The ship-to business partner to which the inventory is allocated after the allocation change order is processed iSetBusinessObjectTypeTo Set allocated Business Object Type - to (true/false) iBusinessObjectTypeTo The business object type (Order type) to which the inventory is allocated after the allocation change order is processed iSetBusinessObjectTo Set allocated Business Object - to (true/false) iBusinessObjectTo The business object (Order) to which the inventory is allocated after the allocation change order is processed iSetBusinessObjectReferenceTo Set allocated business object reference - to (true/false) iBusinessObjectReferenceTo The business object reference (Order line) to which the inventory is allocated after the allocation change order is processed iSetReferenceTo Set Reference - to (true/false) iReferenceTo A reference code to which the inventory was allocated after the allocation change order is processed iSetLineText Set allocation change order line text (true/false) iLineText Allocation change order line text number Output: oAllocationChangeOrder The generated allocation change order number. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK <> 0: Error </pre>

AllocationChangeOrder.Process

DLL:	whextinhapi
	This function is available from KB3516844 .
Syntax:	<pre> long AllocationChangeOrder.Process(ref domain tcorno iAllocationChangeOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function processes the given Allocation Change Order. Only allowed when the Demand Pegging concept is enabled in Implemented Software Components (tccom0500m000).

	This function does not print labels. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iAllocationChangeOrder The allocation change order number (mandatory). Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0: OK <> 0: Error

Public Interfaces for Lot

The following functions are available:

[Lot.Generate](#)

[Lot.StartDetail](#)

[Lot.StartGlobalLotBlocking](#)

[Lot.StartOverview](#)

[Lot.StartPrintLotBlocking](#)

Lot.Generate

DLL:	whextltcapi		
	This function is available from KB2303855 .		
Syntax:	<pre>long Lot.Generate (domain tcitem iItem, domain whltc.olot iLotOrigin, domain tcsite iSite, domain whinh.oorg iOrderOrigin, domain tcorno iOrder, domain tcwset iOrderSet, domain tccom.bpid iShipFromBusinessPartner, domain tcuef.effn iEffectivityUnit, domain tcedm.revi iRevision, domain whltc.ltbp iBusinessPartnerLot, domain tcmcs.cmnf iManufacturer,</pre>		

	domain whltc.cert iCertificateNumber, domain tcclot iOriginalLot, ref domain tcclot oLot, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This public interface can be used to generate new lot codes. Based on the lot mask setup, the mask can require the order information for which the lot code is to be generated, which can be provided as input to this function (iOrderOrigin, iOrder, iOrderSet). When the iLotOrigin is Purchase, the iShipFromBusinessPartner must be provided as a lot with this origin requires the reference to the Ship-From Business Partner. Pre: db.retry.point() Post: abort/commit transaction Input: - iItem - Mandatory - iLotOrigin - Mandatory, possible values are: - whltc.olot.maint - Maintenance - whltc.olot.prod - Production - whltc.olot.purch - Purchase - iSite - Optional - iOrderOrigin - Optional - iOrder - Optional - iOrderSet - Optional - iShipFromBusinessPartner - Conditionally Mandatory; When the iLotOrigin is Purchase, this field is mandatory. - iEffectivityUnit - Optional - iRevision - Optional - iBusinessPartnerLot - Optional - iManufacturer - Optional - iCertificateNumber - Optional - iOriginalLot - Optional Output: - oLot - Generated lot - oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Success <> 0 - Error

Lot.StartDetail

DLL:	whextltcapi
Syntax:	long Lot.StartDetail(long iStartMode, domain tcitem iItem, domain tcclot iLot, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function starts the detail session Item - Lot (whltc1100m000). Input: iStartMode

	Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iItem iLot Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

Lot.StartGlobalLotBlocking

DLL:	whextwmdapi
	This function is available from KB2280149 .
Syntax:	<pre>long Lot.StartGlobalLotBlocking(long iStartMode, boolean iIgnoreSelectionFields, domain tcclot iFromLot, domain tcitem iFromItem, domain tcclot iToLot, domain tcitem iToItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl This function starts the session Global Lot Blocking (whwmd6220m000). Depending on the main table of the calling session, Non-Consecutive Record Selection (NCRS) is used. When the main table is: Lots (whltc100) or Lot Blocking (whwmd620) then the range selections will be imported based on the selected records in the parent session and the selection related fields will be ignored. Input: iStartMode Not used Session is always started in MODAL mode. iIgnoreSelectionFields

	If true, (empty) From/To selection fields will not be filled in the session. iFromLot From Lot selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iFromItem From Item selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToLot To Lot selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToItem To Item selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

Lot.StartOverview

DLL:	whextltcapi
Syntax:	<pre>long Lot.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, domain tcclot iLot, ref domain tcitem oItem, ref domain tcclot oLot, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the overview session Item - Lot (whltcl100m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously.

	<pre> iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byItem": data is displayed by: Item session will be started on index 1 view field: Item "byItemBuyFromBP": data is displayed by: Item BuyFromBusinessPartner session will be started on index 2 view fields: Item "byItemBuyFromBPLot": data is displayed by: Item BuyFromBusinessPartner BuyFromBusinessPartnerLot session will be started on index 3 view field: Item "byItemOriginalLot": data is displayed by: Item Original Lot session will be started on index 8 view field: Item iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iItem Mandatory when startfilter "byItem", "byItemBuyFromBP", "byItemBuyFromBPLot" or "byItemOriginalLot" is used and the session is started in overview mode with start mode MODELESS iLot Output: for iStartMode MODAL: oItem - Selected Item. oLot - Selected Lot. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error </pre>

Lot.StartPrintLotBlocking

DLL:	whextwmdapi This function is available from KB2274420 .
Syntax:	<pre> long Lot.StartPrintLotBlocking(long iStartMode, boolean iIgnoreSelectionFields, domain tcitem iFromItem, domain tcclot iFromLot, domain tccdis iFromReason, domain tcitem iToItem, domain tcclot iToLot, domain tccdis iToReason, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl This function starts the session Print Lot Blocking (whwmd6420m000). Depending on the main table of the calling session, Non-Consecutive Record Selection (NCRS) is used. When the main table is: Lot Blocking (whwmd620) then the range selections will be imported based on the selected records in the parent session and the selection related fields will be ignored. Input: iStartMode Not used Session is always started in MODAL mode. iIgnoreSelectionFields If true, (empty) From/To selection fields will not be filled in the session. iFromItem From Item selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iFromLot From Lot selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iFromReason From Reason selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToItem To Item selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToLot To Lot selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToReason To Reason selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error

	information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - Session started - Error.

Public Interfaces for LotInventoryByWarehouseAndItem

The following functions are available:

[LotInventoryByWarehouseAndItem.StartDetail](#)

[LotInventoryByWarehouseAndItem.StartMultiMain](#)

[LotInventoryByWarehouseAndItem.StartOverview](#)

LotInventoryByWarehouseAndItem.StartDetail

DLL:	whextltcapi This function is available from KB2280148.
Syntax:	<pre>long LotInventoryByWarehouseAndItem.StartDetail(long iStartMode, domain tccwar iWarehouse, domain tcitem iItem, domain tcclot iLot, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the detail session Warehouse - Item - Lots Inventory (whltc1505s000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started.

	Primary Key Fields: - iWarehouse - iItem - iLot Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

LotInventoryByWarehouseAndItem.StartMultiMain

DLL:	whextltcapi This function is available from KB2280148.
Syntax:	<pre>long LotInventoryByWarehouseAndItem.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tccwar iWarehouse, domain tcitem iItem, domain tcclot iLot, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the multi main table session Warehouse - Item - Lots Inventory (whltcl605m000). Input: iStartMode Specifies the start mode for the session. Possible values are: - MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. - MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iQueryExtend A specific query to be used when zooming to this session. Primary Key Fields: - iWarehouse - iItem - iLot Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error

	information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - Session started - Error

LotInventoryByWarehouseAndItem.StartOverview

DLL:	whextltcapi This function is available from KB2280148.
Syntax:	<pre>long LotInventoryByWarehouseAndItem.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccwar iWarehouse, domain tcitem iItem, domain tcclot iLot, ref domain tccwar oWarehouse, ref domain tcitem oItem, ref domain tcclot oLot, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the overview session Warehouse - Item - Lots Inventory (whltc1505m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byWarehouseItem": data is displayed by Warehouse, Item session will be started on index 1 view field: Warehouse, Item "byItem": data is displayed by Item, Lot session will be started on index 2 view field: Item, Lot iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iWarehouse

	Mandatory when startfilter "byWarehouseItem" is used and the session is started in overview mode with start mode MODELESS iItem Mandatory when startfilter "byItem" and the session is started in overview mode with start mode MODELESS iLot Mandatory when startfilter "byItem" and the session is started in overview mode with start mode MODELESS Output: for iStartMode MODAL: oWarehouse - Selected Warehouse. oItem - Selected Item. oLot - Selected Lot. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

Public Interfaces for SerialsByWarehouse

The following functions are available:

[SerialsByWarehouse.StartDetail](#)

[SerialsByWarehouse.StartOverview](#)

SerialsByWarehouse.StartDetail

DLL:	whextltcapi
Syntax:	<pre>long SerialsByWarehouse.StartDetail(long iStartMode, domain tcitem iItem, domain tcibd.sern iSerialNumber, domain tccwar iWarehouse, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the detail session Item - Serials and Warehouses (whlrtc5100m000). Input: iStartMode

	Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iItem iSerialNumber iWarehouse Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

SerialsByWarehouse.StartOverview

DLL:	whextltcapi
Syntax:	<pre>long SerialsByWarehouse.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcitem iItem, domain tcibd.sern iSerialNumber, domain tccwar iWarehouse, domain tcclot iLot, domain whloca iLocation, domain tcinv.date iInventoryDate, domain tccom.bpid iBusinessPartner, ref domain tcitem oItem, ref domain tcibd.sern oSerialNumber, ref domain tccwar oWarehouse, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the overview session Item - Serials and Warehouses (whltc5100m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the

	child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byItem": - data is displayed by: - Item - Serial Number - Warehouse - session will be started on index 1 - view field: - Item "byItemLot": - data is displayed by: - Item - Lot - Serial Number - Warehouse - session will be started on index 2 - view fields: - Item - Lot "byItemWarehouse": - data is displayed by: - Item - Warehouse - Serial Number - session will be started on index 3 - view fields: - Item - Warehouse "byWarehouseItemLocation": - data is displayed by: - Warehouse - Item - Location - Serial Number - session will be started on index 4 - view fields: - Warehouse - Item - Location "byWarehouseLocationItem": - data is displayed by: - Warehouse - Location - Item - Lot - Inventory Date - Serial Number - session will be started on index 5 - view fields: - Warehouse - Location - Item - Lot - Inventory Date

	<pre> "byItemBusinessPartner": data is displayed by: Item Business Partner Serial Number Warehouse session will be started on index 6 view fields: Item Business Partner "byItemWarehouseBusinessPartner": data is displayed by: Item Warehouse Business Partner Serial Number session will be started on index 7 view fields: Item Warehouse Business Partner iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. i.item Mandatory when startfilter "byItem", "byItemLot", "byItemWarehouse", "byWarehouseLocationItem", "byItemBusinessPartner" or "byItemWarehouseBusinessPartner" is used and the session is started in overview mode with start mode MODELESS i.serial.number i.warehouse Mandatory when startfilter "byItemWarehouse", "byWarehouseItemLocation", "byWarehouseLocationItem" or "byItemWarehouseBusinessPartner" is used and the session is started in overview mode with start mode MODELESS i.lot i.warehouse Mandatory when startfilter "byItemLot" or "byWarehouseLocationItem" is used and the session is started in overview mode with start mode MODELESS i.location Mandatory when startfilter "byWarehouseItemLocation" or "byWarehouseLocationItem" is used and the session is started in overview mode with start mode MODELESS i.inventory.date Mandatory when startfilter "byWarehouseItemLocation" is used and the session is started in overview mode with start mode MODELESS i.business.partner Mandatory when startfilter "byItemBusinessPartner" or "byItemWarehouseBusinessPartner" is used and the session is started in overview mode with start mode MODELESS Output: for iStartMode MODAL: oItem - Selected Item. oSerialNumber - Selected Serial Number. oWarehouse - Selected Warehouse. oExceptionMessage - The last message if the return </pre>

	value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Session started - Error Return: 0 <> 0

Public Interfaces for ItemLotSerialTransaction

The following functions are available:

[ItemLotSerialTransaction.GetData](#)

ItemLotSerialTransaction.GetData

DLL:	whextltcapi
	This function is available from KB2275936 .
Syntax:	<pre>long ItemLotSerialTransaction.GetData(domain whltc.tord iOrderOrigin, domain tcorno iOrderNumber, domain tcpono iOrderLine, domain tcitem iItem, domain tcclot iLot, domain tcibd.sern iSerial, domain tctrns.date iTransactionDate, domain tcyesno iDirectDelivery, ref domain tcedm.revi oRevision, ref domain tccom.bpid oBusinessPartner, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function reads order line related data for items which are lot controlled or serialized and for which lot/serial tracking applies. The following fields are retrieved: - Revision (Engineering Item Revision) - Business Partner If possible, the Business Partner is retrieved from the Item, Lot and Serial Transactions (whltc310) for the item, lot and/or serial passed. If not found, it is determined from the Ship-from Code (for receipt transactions) or Ship-to Code (for issue transactions) of the warehousing order (history).

	The Revision is read from the inbound/outbound warehouse order line (history) tables. Pre: n.a. Post: n.a. Input: iOrderOrigin - Order Origin used in the LTC Module (mandatory) iOrderNumber - Order Number (mandatory) iOrderLine - Order Line (mandatory) iItem - Item (mandatory) iLot - Lot iSerial - Serial Number iTransactionDate - Transaction Date (UTC Date/Time) iDirectDelivery - Direct Delivery Y/N (mandatory) Output: oRevision - Engineering Item Revision oBusinessPartner - Business Partner oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK <> 0: error

Public Interfaces for SerializedItem

The following functions are available:

[SerializedItem.Convert](#)

SerializedItem.Convert

DLL:	whextltcapi
	This function is available from KB2318156 .
Syntax:	<pre>long SerializedItem.Convert (domain tcitem iItem, domain whltc.conv.ser iConversionTo, domain tcyesno iSerialTracking, domain tcynna iRegisterSerialEntryForDirectDelivery, domain tcynna iRegisterSerialEntryDuringReceipt, domain tcynna iRegisterSerialEntryDuringTransfer, domain tcynna iRegisterSerialIssueDuringAsBuilt,</pre>

	<pre> domain tcynna iRegisterSerialIssueInService, domain whwmd.rgtm iLotSerialRegistrationTemplate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function converts a non-serialized item or low volume serialized item to a high volume serialized item. In other words, an item which is non-serialized or Serialized (in inventory) is converted to 'Serial not in inventory'. Other conversion options are not (yet) supported. The (mandatory) serial registration settings will be updated in session Item - Warehousing (whwmd4500m000). Optionally, the Serial Tracking and/or Serial Registration Template can also be updated. Pre: db.retry.point() is set. Post: commit.transaction() / abort.transaction() Input: iItem - Item (mandatory) iConversionTo - Conversion To (mandatory): -> Serial Not In Inventory (high volume) iSerialTracking - Serial Tracking (mandatory): -> Yes -> No If the value of this field is Yes, the item's serial number is tracked during receipts and issues in warehousing. iRegisterSerialEntryForDirectDelivery (mandatory) -> Yes -> No Only applicable when converting to high volume serials. If the value of this field is Yes, and the order line on the ASN is a direct delivery purchase line, you must register the serial number in the Shipment Notice Line Stock Point Details (whinh3105m000) session. iRegisterSerialEntryDuringReceipt (mandatory) -> Yes -> No Only applicable when converting to high volume serials. If the value of this field is Yes, you must register the serial numbers of received serialized items in the Receipt Line Stock Point Details (whinh3123m000) session. iRegisterSerialEntryDuringTransfer (mandatory) -> Yes -> No Only applicable when converting to high volume serials. If the value of this field is Yes, you must register the serial numbers for both the inbound and outbound lines of transfer orders, but only if registration during Receipt is also Yes. iRegisterSerialIssueDuringAsBuilt (mandatory) -> Yes -> No Only applicable when converting to high volume serials. The value of this field determine at which point you must register serial numbers for component items that are issued from the warehouse to the job shop to manufacture end items listed on production orders in Manufacturing. If the value of this field is Yes, you must register the serials in Serial End Item - As-Built Components (timfc011m000), and NOT during issue from warehouse. iRegisterSerialIssueInService (mandatory) -> Yes

	-> No Only applicable when converting to high volume serials. If the value of this field is Yes, you must register the serials in Serials in Service and NOT during issue from warehouse. iLotSerialRegistrationTemplate (optional) Only applicable when converting to high volume serials. Lot and serial registration templates are used to specify the order origins and transaction types for which serial and/or lot registration must take place. Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Item has been converted successfully DALHOOKERROR - Item has not been converted successfully

Public Interfaces for LotItem

The following functions are available:

Public Interfaces for Load

The following functions are available:

[Load.Freeze](#)

[Load.Reopen](#)

[Load.StartAutomaticOutboundProcessing](#)

[Load.StartDetail](#)

[Load.StartMultiMain](#)

[Load.StartOverview](#)

Load.Freeze

DLL:	whextinhapi
Syntax:	<pre> long Load.Freeze(domain whinh.load iLoad, domain tcyesno iCalculateAdditionalCosts, ref boolean oShipmentLinesFrozen, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will freeze the given Load. When successful, also the automatic outbound process for printing of shipping documents is started for this load. Be aware that transaction management is handled within this function. Pre: N.a. Post: N.a. Input: iLoad - Mandatory iCalculateAdditionalCosts - Mandatory When this is set to Yes, the existing additional cost lines will first be removed, and a new calculation of the additional cost lines will be performed. This excludes the manually created additional cost lines. Output: oShipmentLinesFrozen - One or more Shipment Lines are frozen oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Load.Reopen

DLL:	whextinhapi
Syntax:	<pre> long Load.Reopen(domain whinh.load iLoad, ref boolean oLoadReopened, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface reopens the frozen Load. Pre: db.retry.point() Post: abort/commit transaction Input: iLoad - Mandatory Output: oLoadReopened - Load is reopened.

	<code>oExceptionMessage</code> - The last message if the return value is not equal to 0. If more than one message is given, these are present in the <code>oExceptionID</code> <code>oExceptionID</code> - An ID that refers to all error information. Use the functions in <code>Exception</code> to get all relevant information. Return: 0: OK, <> 0: Error

Load.StartAutomaticOutboundProcessing

DLL:	whextinhapi
Syntax:	<pre>long Load.StartAutomaticOutboundProcessing(domain whinh.load iLoad, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface will perform the next automatic activity of the given Load. This public interface will execute the following steps: 1) Clear any prior notifications done for the automatic outbound process, within the current process. 2) Notify the Load for automatic processing 3) Start the automatic outbound process for the notified load. The way to implement this public interface is to make sure that the next activity is set to automatic. This can be already set properly based on the order type setup, or can be achieved by using public interface <code>OutboundOrderLineActivity.ModifyAutomaticProcessing()</code> for the activity that should be automatically processed. See below code example: <pre>--- db.retry.point() /* Set the activity to automatic ret = OutboundOrderLineActivity.ModifyAutomaticProcessing(.. if ret <> 0 then abort.transaction() return(ret) endif commit.transaction() /* Start the automatic order process ret.val = Load.StartAutomaticOutboundProcessing(.. db.retry.point() /* Set the activity to be not automatic ret = OutboundOrderLineActivity.ModifyAutomaticProcessing(..</pre>

	<pre> if ret <> 0 then abort.transaction() return(ret) endif commit.transaction() return(ret.val) ---</pre> This public interface will start the next automatic activity for the related load. This public interface will stop the automatic process when the next activity is not set to automatic. This function must not be called within a logical transaction as this function will have it's own transaction handling. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iLoad - Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Load.StartDetail

DLL:	whextinhapi
Syntax:	<pre> long Load.StartDetail(long iStartMode, domain whinh.load iLoad, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the detail session Load (whinh4140m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variable is the primary key, this field is mandatory, if the primary key cannot be found an API error

	will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iLoad - The iLoad must refer to an existing Load Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

Load.StartMultiMain

DLL:	whextinhapi
Syntax:	<pre> long Load.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain whinh.load iLoad, domain tccfrw iCarrier, domain tccrte iRoute, domain tcdate iPlannedDeliveryDate, domain tcncmp iShipToCompany, domain tctyps iShipToType, domain tccshp iShipToCode, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the multi main table session Load (whinh4640m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Specifies the table index that is to be used. iQueryExtend A specific query to be used when zooming to this session. iLoad Not Mandatory iCarrier Not Mandatory

	<pre> iRoute Not Mandatory iPlannedDeliveryDate Not Mandatory iShipToCompany Not Mandatory iShipToType Not Mandatory iShipToCode Not Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error </pre>

Load.StartOverview

DLL:	whextinhapi
Syntax:	<pre> long Load.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain whinh.load iLoad, domain tccfrw iCarrier, domain tccrte iRoute, domain tcdte iPlannedDeliveryDate, domain tcncmp iShipToCompany, domain tctyps iShipToType, domain tccshp iShipToCode, ref domain whinh.load oLoad, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl This function starts the overview session Loads (whinh4140m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are:

	<pre> "byLoad": data is displayed by Load session will be started on index 1 view field: None "byCarrier": data is displayed by Carrier session will be started on index 2 view field: Carrier "byShipTo": data is displayed by Ship-To session will be started on index 3 view field: None iSessionIndex Specifies the table index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iLoad The Load to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 1 (or iStartFilter = "byLoad") iCarrier Not Mandatory iRoute Not Mandatory iPlannedDeliveryDate Not Mandatory iShipToCompany Not Mandatory iShipToType Not Mandatory iShipToCode Not Mandatory Output: for iStartMode MODAL: oLoad - selected load oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - Session started - Error. </pre>

Public Interfaces for DeliveryNote

The following functions are available:

[DeliveryNote.Print](#)

DeliveryNote.Print

DLL:	whextinhapi This function is available from KB3540150 .
Syntax:	<pre> long DeliveryNote.Print(domain tcdeIn iDeliveryNote, domain whInh.shpm iPreliminaryDeliveryNote, domain tcmcs.str15 iDevice, long iProcessingOptionSet, ref boolean oDataProcessed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will print the delivery note for the iDeliveryNote/iPreliminaryDeliveryNote using the defaults or options as provided in the iProcessingOptionSet. Opening/Closing of the report is handled by this function. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iDeliveryNote Optional iPreliminaryDeliveryNote Optional iDevice Mandatory iProcessingOptionSet Optional, if 0, the default options are applied. A Processing Option Set can be created via a call to function ProcessingOptionSet.Create(). In case iDeliveryNote is filled then field iPreliminaryDeliveryNote and the following options of the iProcessingOptionSet will be ignored: - selection range fields (From/To) - DeliveryNoteArray - PreliminaryDeliveryNoteArray The Delivery Note will then be printed for the given iDeliveryNote. In case iPreliminaryDeliveryNote is filled then the following options of the iProcessingOptionSet will be ignored: - selection range fields (From/To) - DeliveryNoteArray - PreliminaryDeliveryNoteArray The Delivery Note will then be printed for the given iPreliminaryDeliveryNote. In case option DeliveryNoteArray is set then the selection range fields (From/To) and PreliminaryDeliveryNoteArray of the iProcessingOptionSet will be ignored. The Delivery Note will then be printed for the delivery notes in the array.

In case option PreliminaryDeliveryNoteArray is set then the selection range fields (From/To) of the iProcessingOptionSet will be ignored.

The Delivery Note will then be printed for the preliminary delivery notes in the array.

Processing Options have a direct relationship with the form fields on session Print Delivery Notes (whinh4477m000) and are not explained in further detail here.

Please refer to the session help for additional information.

Processing Options that are set while a required Implemented Software Component is not available are ignored.

Explanation about setting of defaults:

Minimum Value: Minimum value of domain is taken as default value.

Maximum Value: Maximum value of domain is taken as default value.

NAME	TYPE	DEFAULT
DeliveryNoteFrom	domain tcdeln	Minimum Value
DeliveryNoteTo	domain tcdeln	Maximum Value
PreliminaryDeliveryNoteFrom	domain whinh.shpm	Minimum Value
PreliminaryDeliveryNoteTo	domain whinh.shpm	Maximum Value
YearFrom	domain tcyrno	Minimum Value
YearTo	domain tcyrno	Maximum Value
ShipmentFrom	domain whinh.shpm	Minimum Value
ShipmentTo	domain whinh.shpm	Maximum Value
LoadFrom	domain whinh.load	Minimum Value
LoadTo	domain whinh.load	Maximum Value
ShipFromTypeFrom	domain tctyps	Minimum Value
ShipFromTypeTo	domain tctyps	Maximum Value
ShipFromCodeFrom	domain tcshpm	Minimum Value
ShipFromCodeTo	domain tcshpm	Maximum Value
ShipToTypeFrom	domain tctyps	Minimum Value
ShipToTypeTo	domain tctyps	Maximum Value
ShipToCodeFrom	domain tcshpm	Minimum Value
ShipToCodeTo	domain tcshpm	Maximum Value
CarrierFrom	domain tccfrw	Minimum Value
CarrierTo	domain tccfrw	Maximum Value
RouteFrom	domain tccrte	Minimum Value
RouteTo	domain tccrte	Maximum Value
ShipmentConfirmDateFrom	domain tcdate	Minimum Value
ShipmentConfirmDateTo	domain tcdate	Current Date/Time
CashOnDeliveryType	domain tckofn	tckofn.both
OpenDeliveryNotes	domain tcyesno	tcyesno.no
UseOrderLanguage	domain tcyesno	tcyesno.yes
ManualDeliveryNotes	domain tcyesno	tcyesno.yes
DeliveryNoteAction	domain whinh.dnac	whinh.dnac.preview
PrintDetailedReport	domain tcyesno	tcyesno.no
PrintOwnership	domain tcyesno	tcyesno.no
NumberOfCopies	domain tcsrno	0
DeliveryNoteArray	domain ttjson	0
PreliminaryDeliveryNoteArray	domain ttjson	0
ReportName	domain tcmcs.str16	Empty String

JSON Object DeliveryNoteArray has the following structure:

```
"DeliveryNoteArray": [
  {
    "DeliveryNote": "DLN008448"
  },
  {
    "DeliveryNote": "DLN008428"
  }
]
```

```
]
```

This structure can be created with the following code:

```
DeliveryNoteArray = Json.newArray()
DeliveryNote = Json.newObject()
Json.setString(DeliveryNote, "deliverynote", "DLN008448")
Json.add(DeliveryNoteArray, DeliveryNote)
DeliveryNote = Json.newObject()
Json.setString(DeliveryNote, "deliverynote", "DLN008428")
Json.add(DeliveryNoteArray, DeliveryNote)
```

In a similar way the PreliminaryNoteArray can be set.

ReportName only needs to be filled for customized reports, otherwise the standard report is automatically used.

ReportName must start with an "r", e.g. "rwhinh447701001"

Output: oDataProcessed	- true: Delivery Note Printed. false: Nothing Printed.
oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.
oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.

Return: 0: OK, <> 0: Error

Public Interfaces for Shipment

The following functions are available:

[Shipment.Confirm](#)

[Shipment.Freeze](#)

[Shipment.MoveToLoad](#)

[Shipment.NotifyAutomaticOutboundProcess](#)

[Shipment.PrintLabelsByPackageLabelBOD](#)

[Shipment.PrintPackingList](#)

[Shipment.PrintPackingSlip](#)

[Shipment.PrintPickAndLoadSheet](#)

[Shipment.ProcessInvoice](#)

[Shipment.Reopen](#)

[Shipment.StartAutomaticOutboundProcessing](#)

[Shipment.StartDetail](#)[Shipment.StartMultiMain](#)[Shipment.StartOverview](#)

Shipment.Confirm

DLL:	whextinhapi
Syntax:	<pre> long Shipment.Confirm(domain whinh.shpm iShipment, domain tctrns.date iInventoryAdjustmentDate, domain tcyesno iCalculateAdditionalCosts, ref boolean oShipmentLinesConfirmed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function will confirm the given Shipment. When successful, also the automatic outbound process is started for this shipment. Be aware that transaction management is handled within this function. Pre: N.a. Post: N.a. Input: iShipment - Mandatory iInventoryAdjustmentDate - Not mandatory. This field is used for filling of the inventory adjustment date of the shipment line (whinh431.iadr). The iInventoryAdjustmentDate will only be used when whinh431.iadr is zero. When iInventoryAdjustmentDate has the value zero then whinh431.iadr will be filled with the current date (utc.num()). For additional cost lines the date will remain zero. iCalculateAdditionalCosts - Mandatory When this is set to Yes, the existing additional cost lines will first be removed, and a new calculation of the additional cost lines will be performed. This excludes the manually created additional cost lines. Output: oShipmentLinesConfirmed - One ore more Shipment Lines are confirmed oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in

	Exception to get all relevant information. Return: 0: OK, <> 0: Error

Shipment.Freeze

DLL:	whextinhapi
Syntax:	<pre>long Shipment.Freeze(domain whinh.shpm iShipment, domain tcyesno iCalculateAdditionalCosts, ref boolean oShipmentLinesFrozen, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function will freeze the given Shipment. When successful, also the automatic outbound process for printing of shipping documents is started for this shipment. Be aware that transaction management is handled within this function. Pre: N.a. Post: N.a. Input: iShipment - Mandatory iCalculateAdditionalCosts - Mandatory When this is set to Yes, the existing additional cost lines will first be removed, and a new calculation of the additional cost lines will be performed. This excludes the manually created additional cost lines. Output: oShipmentLinesFrozen - One or more Shipment Lines are frozen oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Shipment.MoveToLoad

DLL:	whextinhapi
Syntax:	long Shipment.MoveToLoad(

	domain whinh.shpm iShipment, domain whinh.load iDestinationLoad, domain tccdis iReason, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This public interface supports moving a shipment to the destination load. Pre: db.retry.point() Post: abort/commit transactions. Input: iShipment - Shipment; Mandatory iDestinationLoad - Destination Load; Mandatory iReason - Compose Reason; Mandatory if destination Load is planned by FM. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Shipment.NotifyAutomaticOutboundProcess

DLL:	whextinhapi This function is available from KB3628999 .
Syntax:	long Shipment.NotifyAutomaticOutboundProcess(domain whinh.shpm iShipment, ref boolean oArrayElementAdded, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This Public Interface stores the Shipments that must be processed automatically. Pre: N.A. Post: Start Warehousing.StartAutomaticOutboundProcessing() Input: iShipment - Shipment. Must be in status Confirmed or Frozen Output: oArrayElementAdded - True if the iShipment has been added to array oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0/DALHOOKERROR

Shipment.PrintLabelsByPackageLabelBOD

DLL:	whextinhapi
Syntax:	<pre> long Shipment.PrintLabelsByPackageLabelBOD(domain whinh.shpm iShipment, domain tclabl iLabelLayout, domain tcmcs.long iNumberOfCopies, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will publish the PackageLabelBOD for the handling units that are linked to the iShipment. For each top level handling unit the PackageLabelBOD will be published. Pre: - There should be no pending logical transaction before calling this function. - Handling Units should be present for the iShipment - PackageLabelBOD should be implemented. Post: No need to commit or abort the process, that is handled within the function. Input: iShipment - Mandatory iLabelLayout - Label Layout iNumberOfCopies - Number of copies; By default 1 label is printed per labeled handling unit, so passing this variable as 1 will mean 2 labels are printed, the original one and a copy. Maximum value is 99. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Shipment.PrintPackingList

DLL:	whextinhapi
	This function is available from KB3516507 .
Syntax:	<pre> long Shipment.PrintPackingList(domain whinh.shpm iShipment, domain tcmcs.str15 iDevice, long iProcessingOptionSet, ref boolean oDataProcessed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will print the packing list for the iShipment using the defaults or options as provided in the iProcessingOptionSet. Opening/Closing of the report is handled by this function.

	Possible values of ReportNumber are: 1 - By Shipment JSON Object ShipmentArray has the following structure: "ShipmentArray": [{ "Shipment": "SHP008448" }, { "Shipment": "SHP008428" }] This structure can be created with the following code: ShipmentArray = Json.newArray() Shipment = Json.newObject() Json.setString(Shipment, "shipment", "SHP008448") Json.add(ShipmentArray, Shipment) Shipment = Json.newObject() Json.setString(Shipment, "shipment", "SHP008428") Json.add(ShipmentArray, Shipment) ReportName only needs to be filled for customized reports, otherwise the standard report is used based on the ReportNumber. ReportName must start with an "r", e.g. "rwhinh447601001" Output: oDataProcessed - true: Packing List Printed. false: Nothing Printed. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Shipment.PrintPackingSlip

DLL:	whextinhapi
	This function is available from KB2320617 .
Syntax:	<pre>long Shipment.PrintPackingSlip(domain whinh.shpm iShipment, domain tcmcs.str15 iDevice, long iProcessingOptionSet, ref oDataProcessed, ref domain tcmcs.s999m oExceptionMessage mb, ref oExceptionID)</pre>
Usage:	Expl: This public interface will print the packing slip for the iShipment using the defaults or options as provided in the iProcessingOptionSet. Pre: db.retry.point must be set The report must be opened for the Idevice.

Pre:	The report must be opened for the Idevice. Which report and report group must be opened depends on the SortByOrder: - whinh.plss.ascending : Ascending Report - whinh.plss.descending : Descending Report - whinh.plss.acc.order.type: Both Reports																																																																																											
Post:	The report must be closed for the Idevice.																																																																																											
Input:	iShipment	Optional																																																																																										
	iDevice	Mandatory																																																																																										
	iProcessingOptionSet	Optional, if 0, the default options are applied. A Processing Option Set can be created via a call to function ProcessingOptionSet.Create().																																																																																										
 Processing Options have a direct relationship with the form fields on session Print Pick and Load Sheet (whinh4430m200) and are not explained in further detail here. Please refer to the session help for additional information. Processing Options that are set while a required Implemented Software Component is not available are ignored. Explanation about setting of defaults: Minimum Value: Minimum value of domain is taken as default value. Maximum Value: Maximum value of domain is taken as default value.																																																																																												
<table><tr><td>NAME</td><td>TYPE</td><td>DEFAULT</td></tr><tr><td>LoadFrom</td><td>domain whinh.load</td><td>Minimum Value</td></tr><tr><td>LoadTo</td><td>domain whinh.load</td><td>Maximum Value</td></tr><tr><td>ShipmentFrom</td><td>domain whinh.shpm</td><td>Minimum Value</td></tr><tr><td>ShipmentTo</td><td>domain whinh.shpm</td><td>Maximum Value</td></tr><tr><td>ShipToTypeFrom</td><td>domain tctyps</td><td>Minimum Value</td></tr><tr><td>ShipToTypeTo</td><td>domain tctyps</td><td>Maximum Value</td></tr><tr><td>ShipToCodeFrom</td><td>domain tccshp</td><td>Minimum Value</td></tr><tr><td>ShipToCodeTo</td><td>domain tccshp</td><td>Maximum Value</td></tr><tr><td>RouteFrom</td><td>domain tccrte</td><td>Minimum Value</td></tr><tr><td>RouteTo</td><td>domain tccrte</td><td>Maximum Value</td></tr><tr><td>PlannedDeliveryDateFrom</td><td>domain tcdate</td><td>Minimum Value</td></tr><tr><td>PlannedDeliveryDateTo</td><td>domain tcdate</td><td>Maximum Value</td></tr><tr><td>ShipmentLineStatusFrom</td><td>domain whinh.shst</td><td>Minimum Value</td></tr><tr><td>ShipmentLineStatusTo</td><td>domain whinh.shst</td><td>Maximum Value</td></tr><tr><td>OrderOriginFrom</td><td>domain whinh.oorg</td><td>Minimum Value</td></tr><tr><td>OrderOriginTo</td><td>domain whinh.oorg</td><td>Maximum Value</td></tr><tr><td>OrderNumberFrom</td><td>domain tcorno</td><td>Minimum Value</td></tr><tr><td>OrderNumberTo</td><td>domain tcorno</td><td>Maximum Value</td></tr><tr><td>OrderLineFrom</td><td>domain tcpono</td><td>Minimum Value</td></tr><tr><td>OrderLineTo</td><td>domain tcpono</td><td>Maximum Value</td></tr><tr><td>WarehouseFrom</td><td>domain tccwar</td><td>Minimum Value</td></tr><tr><td>WarehouseTo</td><td>domain tccwar</td><td>Maximum Value</td></tr><tr><td>ItemFrom</td><td>domain tcitem</td><td>Minimum Value</td></tr><tr><td>ItemTo</td><td>domain tcitem</td><td>Maximum Value</td></tr><tr><td>SortByReference</td><td>domain whinh.sssr</td><td>whinh.sssr.reference</td></tr><tr><td>SortByOrder</td><td>domain whinh.plss</td><td>whinh.plss.ascending</td></tr><tr><td>PrintShipmentAsBarcode</td><td>domain tcyesno</td><td>tcyesno.no</td></tr><tr><td>BarcodeType</td><td>domain tcbctype</td><td>tcbctype.not.appl</td></tr><tr><td>BarcodeHeight</td><td>tcmcs.bytel</td><td>0</td></tr></table>			NAME	TYPE	DEFAULT	LoadFrom	domain whinh.load	Minimum Value	LoadTo	domain whinh.load	Maximum Value	ShipmentFrom	domain whinh.shpm	Minimum Value	ShipmentTo	domain whinh.shpm	Maximum Value	ShipToTypeFrom	domain tctyps	Minimum Value	ShipToTypeTo	domain tctyps	Maximum Value	ShipToCodeFrom	domain tccshp	Minimum Value	ShipToCodeTo	domain tccshp	Maximum Value	RouteFrom	domain tccrte	Minimum Value	RouteTo	domain tccrte	Maximum Value	PlannedDeliveryDateFrom	domain tcdate	Minimum Value	PlannedDeliveryDateTo	domain tcdate	Maximum Value	ShipmentLineStatusFrom	domain whinh.shst	Minimum Value	ShipmentLineStatusTo	domain whinh.shst	Maximum Value	OrderOriginFrom	domain whinh.oorg	Minimum Value	OrderOriginTo	domain whinh.oorg	Maximum Value	OrderNumberFrom	domain tcorno	Minimum Value	OrderNumberTo	domain tcorno	Maximum Value	OrderLineFrom	domain tcpono	Minimum Value	OrderLineTo	domain tcpono	Maximum Value	WarehouseFrom	domain tccwar	Minimum Value	WarehouseTo	domain tccwar	Maximum Value	ItemFrom	domain tcitem	Minimum Value	ItemTo	domain tcitem	Maximum Value	SortByReference	domain whinh.sssr	whinh.sssr.reference	SortByOrder	domain whinh.plss	whinh.plss.ascending	PrintShipmentAsBarcode	domain tcyesno	tcyesno.no	BarcodeType	domain tcbctype	tcbctype.not.appl	BarcodeHeight	tcmcs.bytel	0
NAME	TYPE	DEFAULT																																																																																										
LoadFrom	domain whinh.load	Minimum Value																																																																																										
LoadTo	domain whinh.load	Maximum Value																																																																																										
ShipmentFrom	domain whinh.shpm	Minimum Value																																																																																										
ShipmentTo	domain whinh.shpm	Maximum Value																																																																																										
ShipToTypeFrom	domain tctyps	Minimum Value																																																																																										
ShipToTypeTo	domain tctyps	Maximum Value																																																																																										
ShipToCodeFrom	domain tccshp	Minimum Value																																																																																										
ShipToCodeTo	domain tccshp	Maximum Value																																																																																										
RouteFrom	domain tccrte	Minimum Value																																																																																										
RouteTo	domain tccrte	Maximum Value																																																																																										
PlannedDeliveryDateFrom	domain tcdate	Minimum Value																																																																																										
PlannedDeliveryDateTo	domain tcdate	Maximum Value																																																																																										
ShipmentLineStatusFrom	domain whinh.shst	Minimum Value																																																																																										
ShipmentLineStatusTo	domain whinh.shst	Maximum Value																																																																																										
OrderOriginFrom	domain whinh.oorg	Minimum Value																																																																																										
OrderOriginTo	domain whinh.oorg	Maximum Value																																																																																										
OrderNumberFrom	domain tcorno	Minimum Value																																																																																										
OrderNumberTo	domain tcorno	Maximum Value																																																																																										
OrderLineFrom	domain tcpono	Minimum Value																																																																																										
OrderLineTo	domain tcpono	Maximum Value																																																																																										
WarehouseFrom	domain tccwar	Minimum Value																																																																																										
WarehouseTo	domain tccwar	Maximum Value																																																																																										
ItemFrom	domain tcitem	Minimum Value																																																																																										
ItemTo	domain tcitem	Maximum Value																																																																																										
SortByReference	domain whinh.sssr	whinh.sssr.reference																																																																																										
SortByOrder	domain whinh.plss	whinh.plss.ascending																																																																																										
PrintShipmentAsBarcode	domain tcyesno	tcyesno.no																																																																																										
BarcodeType	domain tcbctype	tcbctype.not.appl																																																																																										
BarcodeHeight	tcmcs.bytel	0																																																																																										
Output:	oDataProcessed	- true: Pick and Load Sheet Printed. false: Nothing Printed.																																																																																										
	oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.																																																																																										

	<code>oExceptionID</code> - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Shipment.ProcessInvoice

DLL:	whextinhapi This function is available from KB3621390.												
Syntax:	<pre> long Shipment.ProcessInvoice(domain whinh.shpm iShipment, domain tcslitinv iTypeOfInvoice, domain tcpitp iInvoiceType, boolean iProcessInvoice, domain tcmcs.str15 iInvoiceDevice, domain tcmcs.str15 iErrorDevice, long iProcessingOptionSet, ref boolean oSetNewRetryPoint, ref boolean oErrorFound, ref boolean oDataProcessed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>												
Usage:	Expl: This public interface will process the invoice for the <code>iShipment</code> using the defaults or options as provided in the <code>i.processing.option.set</code>. If the <code>i.type.of.invoice = Standard</code> a real (definitive) invoice will be created. Roughly this function comprises of two steps: 1) For each billable order origin the corresponding source package is called for creating the billable lines in CI. 2) Finally, CI is called for composing and printing the invoice. Step 2 is only performed when <code>iProcessInvoice</code> is true. Opening/Closing of the report(s) is handled by this function. Pre: Shipment(s) are Frozen / Confirmed. <code>db.retry.point</code> must be set when <code>iProcessInvoice</code> is false. Post: No need to commit or abort the process, that is handled within the function. Input: <table> <tr> <td><code>iShipment</code></td><td>Optional</td></tr> <tr> <td><code>iTypeOfInvoice</code></td><td>Pro Forma Invoice, Customs Invoice, Consignment Invoice or Standard Invoice.</td></tr> <tr> <td><code>iInvoiceType</code></td><td>Mandatory when <code>iTypeOfInvoice</code> is Pro Forma Invoice, Customs Invoice or Consignment Invoice. Ignored for Standard Invoice.</td></tr> <tr> <td><code>iProcessInvoice</code></td><td>Mandatory, true - Step 2 is performed. false - Step 2 is not performed.</td></tr> <tr> <td><code>iInvoiceDevice</code></td><td>Mandatory but will be overruled with predefined device if <code>PrintToPredefinedDevice</code> has value <code>tcyesno.yes</code>.</td></tr> <tr> <td><code>iErrorDevice</code></td><td>Mandatory when <code>PrintErrors</code> has value <code>tcyesno.yes</code>.</td></tr> </table>	<code>iShipment</code>	Optional	<code>iTypeOfInvoice</code>	Pro Forma Invoice, Customs Invoice, Consignment Invoice or Standard Invoice.	<code>iInvoiceType</code>	Mandatory when <code>iTypeOfInvoice</code> is Pro Forma Invoice, Customs Invoice or Consignment Invoice. Ignored for Standard Invoice.	<code>iProcessInvoice</code>	Mandatory, true - Step 2 is performed. false - Step 2 is not performed.	<code>iInvoiceDevice</code>	Mandatory but will be overruled with predefined device if <code>PrintToPredefinedDevice</code> has value <code>tcyesno.yes</code> .	<code>iErrorDevice</code>	Mandatory when <code>PrintErrors</code> has value <code>tcyesno.yes</code> .
<code>iShipment</code>	Optional												
<code>iTypeOfInvoice</code>	Pro Forma Invoice, Customs Invoice, Consignment Invoice or Standard Invoice.												
<code>iInvoiceType</code>	Mandatory when <code>iTypeOfInvoice</code> is Pro Forma Invoice, Customs Invoice or Consignment Invoice. Ignored for Standard Invoice.												
<code>iProcessInvoice</code>	Mandatory, true - Step 2 is performed. false - Step 2 is not performed.												
<code>iInvoiceDevice</code>	Mandatory but will be overruled with predefined device if <code>PrintToPredefinedDevice</code> has value <code>tcyesno.yes</code> .												
<code>iErrorDevice</code>	Mandatory when <code>PrintErrors</code> has value <code>tcyesno.yes</code> .												

	<pre>Shipment = Json.newObject() Json.setString(Shipment, "Shipment", "SHP008428") Json.add(ShipmentArray, Shipment)</pre> ReportName only needs to be filled for customized error report, otherwise the report related to the session is automatically used. ReportName must start with an "r", e.g. "rwhinh427901001" Output: oSetNewRetryPoint - true: New db.retry.point needs to be set. false: No new db.retry.point needs to be set. oErrorFound - Error found oDataProcessed - true: Invoice data is processed. false: Nothing Printed. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Shipment.Reopen

DLL:	whextinhapi
Syntax:	<pre>long Shipment.Reopen(domain whinh.shpm iShipment, ref boolean oShipmentReopened, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface reopens the frozen Shipment. Pre: db.retry.point() Post: abort/commit transaction Input: iShipment - Mandatory Output: oShipmentReopened - Shipment is reopened. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Shipment.StartAutomaticOutboundProcessing

DLL:	whextinhapi
Syntax:	<pre> long Shipment.StartAutomaticOutboundProcessing(domain whinh.shpm iShipment, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will perform the next automatic activity of the given shipment. This public interface will execute the following steps: 1) Clear any prior notifications done for the automatic outbound process, within the current process. 2) Notify the shipment for automatic processing 3) Start the automatic outbound process for the notified shipment The way to implement this public interface is to make sure that the next activity is set to automatic. This can be already set properly based on the order type setup, or can be achieved by using public interface OutboundOrderLineActivity.ModifyAutomaticProcessing() for the activity that should be automatically processed. See below code example: <pre> --- db.retry.point() /* Set the activity to automatic ret = OutboundOrderLineActivity.ModifyAutomaticProcessing(.. if ret <> 0 then abort.transaction() return(ret) endif commit.transaction() /* Start the automatic order process ret.val = Shipment.StartAutomaticOutboundProcessing(.. db.retry.point() /* Set the activity to be not automatic ret = OutboundOrderLineActivity.ModifyAutomaticProcessing(.. if ret <> 0 then abort.transaction() return(ret) endif commit.transaction() return(ret.val) --- </pre> This public interface will start the next automatic activity for the related shipment. This public interface will stop the automatic process when the next activity is not set to automatic.

	This function must not be called within a logical transaction as this function will have it's own transaction handling. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iShipment - Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Shipment.StartDetail

DLL:	whextinhapi
Syntax:	<pre>long Shipment.StartDetail(long iStartMode, domain whinh.shpm iShipment, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the detail session Shipment (whinh4130m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variable is the primary key, this field is mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iShipment - The iShipment must refer to an existing Shipment Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started

	<> 0	- Error.

Shipment.StartMultiMain

DLL:	whextinhapi
Syntax:	<pre> long Shipment.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain whinh.shpm iShipment, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the multi main table session Shipment (whinh4630m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Specifies the table index that is to be used. iQueryExtend A specific query to be used when zooming to this session. iShipment Not Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

Shipment.StartOverview

DLL:	whextinhapi
Syntax:	long Shipment.StartOverview(

	<pre> long iStartMode, domain tcmcs.st30 iStartFilter, const long iSessionIndex, string iQueryExtend(), domain whinh.shpm iShipment, domain whinh.load iLoad, domain tcpono iLoadingListSequence, domain whinh.shpm iDeliveryNote, domain whinh.cntr iShippingContainer, domain tcrefs iShipmentReference mb, domain tccorn iCustomerOrder mb, ref domain whinh.shpm oShipment, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl This function starts the overview session Shipments (whinh4130m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byShipment": data is displayed by Shipment session will be started on index 1 view field: None "byLoad": data is displayed by Load session will be started on index 2 view field: Load "byLoadingList": data is displayed by Loading List session will be started on index 3 view field: Load "byDeliveryNote": data is displayed by Delivery Note session will be started on index 4 view field: Preliminary Delivery Note "byShippingContainer": data is displayed by Shipping Container session will be started on index 5 view field: Load, Shipping Container "byShipmentReference": data is displayed by Shipment Reference session will be started on index 6 view field: Shipment Reference "byCustomerOrder": data is displayed by Customer Order session will be started on index 7 view field: Customer Order iSessionIndex Specifies the table index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable.

	<pre> iQueryExtend A specific query to be used when zooming to this session. iShipment The Shipment to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 1 (or iStartFilter = "byShipment") iLoad Not Mandatory iLoadingListSequence Not Mandatory iDeliveryNote Not Mandatory iShippingContainer Not Mandatory iShipmentReference Not Mandatory iCustomerOrder Not Mandatory Output: for iStartMode MODAL: oShipment oExceptionMessage oExceptionID Return: 0 <> 0 </pre>	<pre> - selected shipment - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Session started - Error. </pre>

Public Interfaces for ShipmentLine

The following functions are available:

[ShipmentLine.Confirm](#)

[ShipmentLine.ConfirmIsAllowed](#)

[ShipmentLine.MoveToLoad](#)

[ShipmentLine.MoveToShipment](#)

[ShipmentLine.PrintLabelsByPackageLabelBOD](#)

[ShipmentLine.Split](#)

[ShipmentLine.StartDetail](#)

[ShipmentLine.StartOverview](#)

ShipmentLine.Confirm

DLL:	whextinhapi
Syntax:	<pre> long ShipmentLine.Confirm(domain whinh.shpm iShipment, domain tcpono iShipmentLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will confirm the given shipment line. When successful, also the automatic outbound process is started for this shipment line. Be aware that transaction management is handled within this function. Pre: Not applicable Post: Not applicable Input: iShipment - Mandatory iShipmentLine - Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

ShipmentLine.ConfirmIsAllowed

DLL:	whextinhapi
	This function is available from KB2329564 .
Syntax:	<pre> long ShipmentLine.ConfirmIsAllowed(domain whinh.shpm iShipment, domain tcpono iShipmentLine, boolean iReadShipmentLine, ref boolean oConfirmationAllowed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will determine if the given shipment line is allowed to be confirmed or not. Pre: Not applicable Post: Not applicable Input: iShipment - Mandatory iShipmentLine - Mandatory iReadShipmentLine - Mandatory True -> Shipment Line will be read. False-> Shipment Line is expected to be current. Output: oConfirmationAllowed oExceptionMessage - The last message if any message is

	found. If more than one message is given, these are present in the oExceptionID. - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

ShipmentLine.MoveToLoad

DLL:	whextinhapi
Syntax:	<pre> long ShipmentLine.MoveToLoad(domain whinh.shpm iShipment, domain tcpono iShipmentLine, domain whinh.load iDestinationLoad, domain tccdis iReason, boolean iAcceptWarnings, ref domain whinh.shpm oDestinationShipment, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface supports moving a shipment line to the destination load. If parameter iAcceptWarnings is set to True a mismatch between the source shipment and the destination load will be ignored; otherwise error message will be returned. Note that in standard LN a mismatch can be ignored by the user in interactive mode. New shipment will be generated and returned to the calling function. Pre: db.retry.point() Post: abort/commit transactions. Input: - iShipment - Shipment; Mandatory - iShipmentLine - Shipment Line; Mandatory - iDestinationLoad - Destination Load; Mandatory - iReason - Compose Reason; Mandatory if destination Load is planned by FM. - iAcceptWarnings - Accept warnings; Mandatory Output: - oDestinationShipment - Generated shipment - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

ShipmentLine.MoveToShipment

DLL:	whextinhapi
Syntax:	<pre> long ShipmentLine.MoveToShipment (domain whinh.shpm iShipment, domain tcpono iShipmentLine, ref domain whinh.shpm ioDestinationShipment, domain tccdis iReason, boolean iAcceptWarnings, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface supports moving a shipment line from one shipment to another. If parameter iAcceptWarnings is set to True a mismatch between the source shipment and the destination shipment will be ignored; otherwise error message will be returned. Note that in standard LN a mismatch can be ignored by the user in interactive mode. If no destination shipment is passed a new shipment will be generated and returned to the calling function. Pre: db.retry.point() Post: abort/commit transaction Input: - iShipment - Shipment; Mandatory - iShipmentLine - Shipment Line; Mandatory - ioDestinationShipment - Destination Shipment; Optional If empty new shipment will be created. - iReason - Compose Reason; Mandatory if Shipment is planned by FM. - iReason - Compose Reason; Mandatory if destination shipment belongs to the load which is planned by FM. - iAcceptWarnings - Accept warnings; Mandatory Output: - ioDestinationShipment - Destination Shipment - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

ShipmentLine.PrintLabelsByPackageLabelBOD

DLL:	whextinhapi
Syntax:	<pre> long ShipmentLine.PrintLabelsByPackageLabelBOD (domain whinh.shpm iShipment, domain tcpono iShipmentLine, domain tclabl iLabelLayout, domain tcmcs.long iNumberOfCopies, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>

Usage:	Expl: This public interface will publish the PackageLabelBOD for the handling units that are linked to the iShipment and iShipmentLine. For each top level handling unit the PackageLabelBOD will be published. Pre: - There should be no pending logical transaction before calling this function. - Handling Units should be present for the iShipment and iShipmentLine - PackageLabelBOD should be implemented. Post: No need to commit or abort the process, that is handled within the function. Input: <table border="0"> <tr> <td>iShipment</td><td>- Shipment; Mandatory</td></tr> <tr> <td>iShipmentLine</td><td>- Shipment Line; Mandatory</td></tr> <tr> <td>iLabelLayout</td><td>- Label Layout; Optional</td></tr> <tr> <td>iNumberOfCopies</td><td>- Number of copies; By default 1 label is printed per labeled handling unit, so passing this variable as 1 will mean 2 labels are printed, the original one and a copy. Maximum value is 99.</td></tr> </table> Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error	iShipment	- Shipment; Mandatory	iShipmentLine	- Shipment Line; Mandatory	iLabelLayout	- Label Layout; Optional	iNumberOfCopies	- Number of copies; By default 1 label is printed per labeled handling unit, so passing this variable as 1 will mean 2 labels are printed, the original one and a copy. Maximum value is 99.
iShipment	- Shipment; Mandatory								
iShipmentLine	- Shipment Line; Mandatory								
iLabelLayout	- Label Layout; Optional								
iNumberOfCopies	- Number of copies; By default 1 label is printed per labeled handling unit, so passing this variable as 1 will mean 2 labels are printed, the original one and a copy. Maximum value is 99.								

ShipmentLine.Split

DLL:	whextinhapi
Syntax:	<pre> long ShipmentLine.Split(domain whinh.shpm iShipment, domain tcpono iShipmentLine, domain tcpono iBomLine, domain tcqivl iSplitQuantity, long long iNumberStockPoints, ref domain tcibd.sern iSerialArray() fixed, ref domain tcclot iLotArray() fixed, ref domain tcinvl.date iInventoryDateArray(), ref domain tcuef.effn iEffectivityUnitArray(), ref domain tcqstl iSplitQuantityArray(), ref domain tccuni iSplitUnitArray() fixed, ref domain whinh.shpm ioDestinationShipment, ref domain whinh.load ioDestinationLoad, domain tccdis iReason, boolean boolean iAcceptWarnings, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface supports splitting of a shipment line and moving of the splitted stock points. Split quantity followed by storage unit must be passed for all

	shipment line - stock point details records, which must be split. Total split quantity iSplitQuantity in inventory units must be equal to the sum of the split quantities in stock point details. Based on combination of ioDestinationShipment and ioDestinationLoad the following actions will be executed: <pre> ----- ioDestShipment ioDestLoad Action Filled Empty Splitted part of shipment line moved to the shipment Filled Filled Check whether shipment matches the load. Splitted part of shipment line moved to the shipment Empty Filled New shipment generated for the Load. Splitted part of shipment line moved to the generated shipment. Empty Empty New shipment and load will be generated. ----- </pre> If parameter iAcceptWarnings is set to True and destination shipment or destination load is passed to the public interface a mismatch between the source shipment and the destination shipment or destination load will be ignored; otherwise error message will be returned. Note that in standard LN a mismatch can be ignored by the user in interactive mode. Example 1: Shipment line contains 10 pieces of item A which are distributed over lots LA1 (4 pieces) and LA2 (6 pieces). The user wants to split 2 pieces of the lot LA1 to the new shipment line. Public Interface must be called as follows: <pre> iSplitQuantity = 2 iNumberStockPoints = 1 iLotArray(1) = LA1 iSplitQuantityArray(1) = 2 Result: 2 pieces of lot LA1 are moved to the new Shipment line. </pre> Example 2: Shipment line contains 10 pieces of item A which are distributed over lots LA1 (4 pieces) and LA2 (6 pieces). The user wants to split 2 pieces of the lot LA1 and 3 pieces of the lot LA2 to the new shipment line. Public Interface must be called as follows: <pre> iSplitQuantity = 5 iNumberStockPoints = 2 iLotArray(1) = LA1 iSplitQuantityArray(1) = 2 iLotArray(2) = LA2 iSplitQuantityArray(2) = 3 Result: 2 pieces of lot LA1 and 3 pieces of lot LA2 are moved to the new Shipment line. </pre> Pre: db.retry.point() Split quantities in the shipment line stock point details are zero. Post: abort/commit transaction Input: iShipment - Shipment (mandatory) iShipmentLine - Shipment Line (mandatory) iBomLine - BOM line (optional)

	iSplitQuantity - Split Quantity in Inventory Units (mandatory) iNumberStockPoints - Number of Shipment Line stock point details to be split (mandatory) iSerialArray - Array of serials (optional) iLotArray - Array of lots (optional) iInventoryDateArray - Array of inventory dates (optional) iEffectivityUnitArray - Array of effectivity units (optional) iSplitQuantityArray - Array of split quantities (mandatory) iSplitUnitArray - Array of storage units (mandatory) ioDestinationShipment - Destination Shipment (optional) ioDestinationLoad - Destination Load (optional) iReason - Compose Reason; Mandatory if destination load is planned by FM. iAcceptWarnings - Accept warnings; (mandatory) Output: ioDestinationShipment - Shipment, Split quantity is moved to. ioDestinationLoad - Load, Split quantity is moved to. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

ShipmentLine.StartDetail

DLL:	whextinhapi
	This function is available from KB2280147 .
Syntax:	<pre>long ShipmentLine.StartDetail(long iStartMode, domain whinh.shpm iShipment, domain tcpono iShipmentLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the detail session Shipment Lines (whinh4131m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variable is the primary key, this field is mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started.

	Primary Key Fields: iShipment iShipmentLine Output: oExceptionMessage oExceptionID Return: 0 <> 0	- The iShipment must refer to an existing Shipment - The iShipmentLine must refer to an existing Shipment Line - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Session started - Error.

ShipmentLine.StartOverview

DLL:	whextinhapi This function is available from KB2280147 .	
Syntax:	<pre>long ShipmentLine.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain whinh.shpm iShipment, domain tcpono iShipmentLine, domain whinh.oorg iOrderOrigin, domain tcorno iOrder, domain tcwset iOrderSet, domain tcpono iOrderLine, domain tcpono iOrderSequence, domain whinh.shst iStatus, domain tcitem iItem, domain tcrefs iShipmentReference mb, domain tccorn iCustomerOrder mb, ref domain whinh.shpm oShipment, ref domain tcpono oShipmentLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl This function starts the overview session Shipment Lines (whinh4131m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the	

	started MULTI_OCC session. Possible values are: "byShipment": data is displayed by Shipment session will be started on index 1 view field: Shipment "byOrder": data is displayed by Order session will be started on index 2 view field: Order Origin, Order "byActivity": data is displayed by Activity session will be started on index 3 view field: None "byOrderSet": data is displayed by Order Set session will be started on index 4 view field: Order Origin, Order, Order Set "byItem": data is displayed by Item session will be started on index 5 view field: Item "byShipmentReference": data is displayed by Shipment Reference session will be started on index 6 view field: Shipment Reference "byCustomerOrder": data is displayed by Customer Order session will be started on index 7 view field: Customer Order iSessionIndex Specifies the table index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iShipment The Shipment to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 1 (or iStartFilter = "byShipment") iShipmentLine Not Mandatory iOrderOrigin Mandatory if iStartMode = MODELESS and iSessionIndex = 2 or 4 (or iStartFilter = "byOrder" or or iStartFilter = "byOrderSet") iOrder Mandatory if iStartMode = MODELESS and iSessionIndex = 2 or 4 (or iStartFilter = "byOrder" or or iStartFilter = "byOrderSet") iOrderSet Mandatory if iStartMode = MODELESS and iSessionIndex = 4 (or iStartFilter = "byOrderSet") iOrderLine Not Mandatory iOrderSequence Not Mandatory iStatus Not Mandatory

	<pre> iItem Mandatory if iStartMode = MODELESS and iSessionIndex = 5 (or iStartFilter = "byItem") iShipmentReference Not Mandatory iCustomerOrder Not Mandatory Output: for iStartMode MODAL: oShipment - selected shipment oShipmentLine - selected shipment line oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error. </pre>

Public Interfaces for ShipmentLineStockPointDetail

The following functions are available:

[ShipmentLineStockPointDetails.StartOverview](#)

ShipmentLineStockPointDetails.StartOverview

DLL:	whextinhapi
	This function is available from KB2280145 .
Syntax:	<pre> long ShipmentLineStockPointDetails.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain whinh.shpm iShipment, domain tcpono iShipmentLine, domain tcpono iBomLine, domain tcmcs.long iDistributionSequence, ref domain whinh.shpm oShipment, ref domain tcpono oShipmentLine, ref domain tcpono oBomLine, ref domain tcmcs.long oDistributionSequence, </pre>

	ref	domain	tcmcs.s999m	oExceptionMessage mb,
	ref		long	oExceptionID)
Usage:	Expl	This function starts the overview session Shipment Line Stock Point Details (whinh4133m000).		
	Input:	iStartMode	Specifies the start mode for the session.	
			Possible values are:	
		MODAL -	The parent session is blocked until the child session exits, the session will be started as a zoom session.	
		MODELESS -	Parent and child are parallel sessions that can be manipulated simultaneously.	
		iStartFilter	Not Used.	
		iSessionIndex	Not Used.	
		iQueryExtend	A specific query to be used when zooming to this session.	
		iShipment	The Shipment to be started	
			Mandatory if iStartMode = MODELESS	
		iShipmentLine	The Shipment Line to be started	
			Mandatory if iStartMode = MODELESS	
		iBomLine	Not Mandatory	
		iDistributionSequence	Not Mandatory	
	Output:	for iStartMode MODAL:		
		oShipment	- selected shipment	
		oShipmentLine	- selected shipment line	
		oBomLine	- selected bom line	
		oDistributionSequence	- selected distribution sequence	
		oExceptionMessage	- The last message if the return value is not equal to 0.	
			If more than one message is given, these are present in the oExceptionID	
		oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.	
	Return:	0	- Session started	
		<> 0	- Error.	

Public Interfaces for ProjectedShipment

The following functions are available:

[ProjectedShipment.Generate](#)

[ProjectedShipment.GenerateOutboundAdvice](#)

[ProjectedShipment.ReleaseOutboundAdvice](#)[ProjectedShipment.StartGenerate](#)

ProjectedShipment.Generate

DLL:	whextinhapi This function is available from KB3613618 .												
Syntax:	<pre> long ProjectedShipment.Generate(domain whinh.oorg iOrderOrigin, domain tcorno iOrderNumber, domain tcpono iOrderLine, domain tcpono iOrderSequence, domain whinh.gnps iAction, domain tcmcs.str15 iDevice, long iProcessingOptionSet, ref boolean oDataProcessed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>												
Usage:	Expl: This public interface will (re-)generate or delete one or more projected shipments using the defaults or options as provided in the iProcessingOptionSet. In case iOrderNumber is filled then the following options of the iProcessingOptionSet will be ignored: - selection range fields (From/To) - OutboundOrderLineArray The projected shipments will then be (re-)generated or deleted for the given iOrderOrigin, iOrderNumber, iOrderLine and iOrderSequence. In case option OutboundOrderLineArray is set then the selection range fields (From/To) of the iProcessingOptionSet will be ignored. The projected shipments will then be (re-)generated or deleted for the outbound order lines in the array. Opening/Closing of the report is handled by this function. Pre: By default the transaction handling is done within this function, so there should be no pending logical transaction before calling this function. This is not the case when IncludeTransactionHandling (option of iProcessingOptionSet) is set to False. Post: By default the transaction handling is done within this function, so there is no need to commit or abort the process. That is handled within the function. When IncludeTransactionHandling (option of iProcessingOptionSet) is set to False then the calling process has to handle abort/commit of the process. Input: <table border="0"> <tr> <td>iOrderOrigin</td><td>Mandatory</td></tr> <tr> <td>iOrderNumber</td><td>Optional</td></tr> <tr> <td>iOrderLine</td><td>Optional</td></tr> <tr> <td>iOrderSequence</td><td>Optional</td></tr> <tr> <td>iAction</td><td>Mandatory</td></tr> <tr> <td>iDevice</td><td>Mandatory, unless options PrintShipments and PrintErrors are set to tcyesno.no.</td></tr> </table>	iOrderOrigin	Mandatory	iOrderNumber	Optional	iOrderLine	Optional	iOrderSequence	Optional	iAction	Mandatory	iDevice	Mandatory, unless options PrintShipments and PrintErrors are set to tcyesno.no.
iOrderOrigin	Mandatory												
iOrderNumber	Optional												
iOrderLine	Optional												
iOrderSequence	Optional												
iAction	Mandatory												
iDevice	Mandatory, unless options PrintShipments and PrintErrors are set to tcyesno.no.												

iProcessingOptionSet	Optional, if 0, the default options are applied. A Processing Option Set can be created via a call to function <code>ProcessingOptionSet.Create()</code>. Processing Options have a direct relationship with the form fields on session Generate Projected Shipments (whinh4230m200) and are not explained in further detail here. Please refer to the session help for additional information. Processing Options that are set while a required Implemented Software Component is not available are ignored. Explanation about setting of defaults: Minimum Value: Minimum value of domain is taken as default value. Maximum Value: Maximum value of domain is taken as default value.	
NAME	TYPE	DEFAULT
OrderOriginFrom	domain whinh.oorg	Minimum Value
OrderOriginTo	domain whinh.oorg	Maximum Value
OrderNumberFrom	domain tcorno	Minimum Value
OrderNumberTo	domain tcorno	Maximum Value
OrderSetFrom	domain tcwset	Minimum Value
OrderSetTo	domain tcwset	Maximum Value
OrderLineFrom	domain tcpono	Minimum Value
OrderLineTo	domain tcpono	Maximum Value
OrderSequenceFrom	domain tcpono	Minimum Value
OrderSequenceTo	domain tcpono	Maximum Value
DistributionSequenceFrom	domain tcmcs.long	Minimum Value
DistributionSequenceTo	domain tcmcs.long	Maximum Value
DeliveryDateFrom	domain tcdate	Minimum Value
DeliveryDateTo	domain tcdate	Current Date/Time
WarehouseFrom	domain tccwar	Minimum Value
WarehouseTo	domain tccwar	Maximum Value
ShipToTypeFrom	domain tctyps	Minimum Value
ShipToTypeTo	domain tctyps	Maximum Value
ShipToCodeFrom	domain tcshpm	Minimum Value
ShipToCodeTo	domain tcshpm	Maximum Value
ShipToAddressFrom	domain tccom.cadr	Minimum Value
ShipToAddressTo	domain tccom.cadr	Maximum Value
RouteFrom	domain tccrte	Minimum Value
RouteTo	domain tccrte	Maximum Value
CarrierFrom	domain tccfrw	Minimum Value
CarrierTo	domain tccfrw	Maximum Value
ShipmentReferenceFrom	domain tcrefs	Minimum Value
ShipmentReferenceTo	domain tcrefs	Maximum Value
JobSequenceFrom	domain tcmcs.str30	Minimum Value
JobSequenceTo	domain tcmcs.str30	Maximum Value
ItemFrom	domain tcitem	Minimum Value
ItemTo	domain tcitem	Maximum Value
PrintShipments	domain tcyesno	tcyesno.yes
PrintErrors	domain tcyesno	tcyesno.yes
IncludeTransactionHandling	boolean	true
OutboundOrderLineArray	domain ttjson	0
ReportName	domain tcmcs.str16	Empty String
JSON Object OutboundOrderLineArray has the following structure: <pre>"OutboundOrderLineArray": [{ "OrderOrigin": 1, "OrderNumber": "SLS000030", "OrderLine": 10,</pre>		

```

        "OrderSequence": 0
    },
    {
        "OrderOrigin": 50,
        "OrderNumber": "JSC000012",
        "OrderLine": 20,
        "OrderSequence": 0
    }
]

This structure can be created with the following code:
OutboundOrderLineArray = Json.newArray()
OutboundOrderLine = Json.newObject()
Json.setNumber(OutboundOrderLine, "OrderOrigin", 1)
Json.setString(OutboundOrderLine, "OrderNumber", "SLS000030")
Json.setNumber(OutboundOrderLine, "OrderLine", 10)
Json.setNumber(OutboundOrderLine, "OrderSequence", 1)
Json.add(OutboundOrderLineArray, OutboundOrderLine)
OutboundOrderLine = Json.newObject()
Json.setNumber(OutboundOrderLine, "OrderOrigin", 50)
Json.setString(OutboundOrderLine, "OrderNumber", "JSC000012")
Json.setNumber(OutboundOrderLine, "OrderLine", 10)
Json.setNumber(OutboundOrderLine, "OrderSequence", 1)
Json.add(OutboundOrderLineArray, OutboundOrderLine)

ReportName only needs to be filled for customized reports, otherwise
the default report is automatically used.
ReportName must start with an "r", e.g. "rwhinh423011001"

Output: oDataProcessed          - true: Projected Shipment(s)
                                   (Re-)Generated/Deleted.
                                   false: Nothing is processed.
oExceptionMessage              - The last message if any message is
                                   found. If more than one message is
                                   given, these are present in the
                                   oExceptionID.
oExceptionID                   - An ID that refers to the exception
                                   information. Use the functions in
                                   Exception to get all relevant
                                   information.

Return: 0                      - Session started
    <> 0                      - Error.

```

ProjectedShipment.GenerateOutboundAdvice

DLL:	whextinhapi		
	This function is available from KB2302540 .		
Syntax:	<pre>long ProjectedShipment.GenerateOutboundAdvice (domain whinh.shpm iShipment, boolean iCrossDock, domain tcyesno iAlternativeItems, domain tcyesno iAdviceDespiteShortage, domain tcyesno iRecalculateExcessATT, domain tcyesno iOverdeliveryAllowed,</pre>		

	ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function will generate an outbound advice for the projected shipment. No data is printed. No automatic outbound process is triggered. Be aware that transaction management is handled within this function. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iShipment - Shipment (Mandatory) iCrossDock - Create cross dock order for shortage (True/False) iAlternativeItems - Alternative items handling (Yes/No) iAdviceDespiteShortage - Advice despite shortage (Yes/No) iRecalculateExcessATT - Recalculate excess ATT (Yes/No) iOverdeliveryAllowed - Overdelivery allowed (Yes/No) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

ProjectedShipment.ReleaseOutboundAdvice

DLL:	whextinhapi This function is available from KB2302540 .
Syntax:	long ProjectedShipment.ReleaseOutboundAdvice(domain whinh.shpm iShipment, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function will release the outbound advice for the projected shipment. No data is printed. No automatic outbound process is triggered. Be aware that transaction management is handled within this function. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iShipment - Shipment (Mandatory) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID.

	<code>oExceptionID</code> - An ID that refers to the exception information. Use the functions in <code>Exception</code> to get all relevant information. Return: 0: OK, <> 0: Error

ProjectedShipment.StartGenerate

DLL:	whextinhapi This function is available from KB2293617.
Syntax:	<pre>long ProjectedShipment.StartGenerate(long iStartMode, boolean iIgnoreSelectionFields, domain whinh.oorg iFromOrderOrigin, domain tcorno iFromOrderNumber, domain tcwset iFromOrderSet, domain tcpono iFromOrderLine, domain tcpono iFromOrderSequence, domain tcmcs.long iFromDistributionSequence, domain whinh.oorg iToOrderOrigin, domain tcorno iToOrderNumber, domain tcwset iToOrderSet, domain tcpono iToOrderLine, domain tcpono iToOrderSequence, domain tcmcs.long iToDistributionSequence, ref tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl This function starts the session Generate Projected Shipments (whinh4230m200). Depending on the main table of the calling session, Non-Consecutive Record Selection (NCRS) is used. When the main table is: - Warehousing Orders (whinh200) or - Outbound Order Lines (whinh220) or - Outbound Order Line - Planned Shipment Requirements (whinh483) then the range selections will be imported based on the selected records in the parent session and the selection related fields will be ignored. Input: <code>iStartMode</code> Not Used. <code>iIgnoreSelectionFields</code> If true, From/To selection fields will not be filled in the session. <code>iFromOrderOrigin</code> From Order Origin selection field is filled with this value. (when <code>iIgnoreSelectionFields</code> is false and NCRS is not applicable) <code>iFromOrderNumber</code> From Order Number selection field is filled with this value. (when <code>iIgnoreSelectionFields</code> is false and NCRS is not applicable) <code>iFromOrderSet</code> From Order Set selection field is filled with this value. (when <code>iIgnoreSelectionFields</code> is false and NCRS

	<pre> is not applicable) iFromOrderLine From Order Line selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iFromOrderSequence From Sequence selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iFromDistributionSequence From Distribution Sequence selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToOrderOrigin To Order Origin selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToOrderNumber To Order Number selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToOrderSet To Order Set selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToOrderLine To Order Line selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToOrderSequence To Sequence selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToDistributionSequence To Distribution Sequence selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error. </pre>

Public Interfaces for OutboundAdvice

The following functions are available:

[OutboundAdvice.StartAutomaticOutboundProcessing](#)
[OutboundAdvice.StartDetail](#)
[OutboundAdvice.StartGenerate](#)
[OutboundAdvice.StartOverview](#)

OutboundAdvice.StartAutomaticOutboundProcessing

DLL:	whextinhapi
Syntax:	<pre> long OutboundAdvice.StartAutomaticOutboundProcessing(domain whinh.oorg iOrderOrigin, domain tcorno iOrderNumber, domain tcpono iOrderLine, domain tcpono iOrderSequence, domain tcpono iAdviceLine, domain whinh.btno iOutboundRun, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will perform the next automatic activity of the given outbound advice line. This public interface will execute the following steps: 1) Clear any prior notifications done for the automatic outbound process, within the current process. 2) Notify the outbound advice line for automatic processing 3) Start the automatic outbound process for the notified outbound advice line. The way to implement this public interface is to make sure that the next activity is set to automatic. This can be already set properly based on the order type setup, or can be achieved by using public interface <code>OutboundOrderLineActivity.ModifyAutomaticProcessing()</code> for the activity that should be automatically processed. See below code example: <pre> --- db.retry.point() /* Set the activity to automatic ret = OutboundOrderLineActivity.ModifyAutomaticProcessing(.. if ret <> 0 then abort.transaction() return(ret) endif commit.transaction() /* Start the automatic order process ret.val = OutboundAdvice.StartAutomaticOutboundProcessing(.. db.retry.point() </pre>

	<pre> * Set the activity to be not automatic ret = OutboundOrderLineActivity.ModifyAutomaticProcessing(..) if ret <> 0 then abort.transaction() return(ret) endif commit.transaction() return(ret.val) --- This public interface will start the next automatic activity for the related outbound advice line. This public interface will stop the automatic process when the next activity is not set to automatic. This function must not be called within a logical transaction as this function will have it's own transaction handling. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iOrderOrigin - Mandatory iOrderNumber - Mandatory iOrderLine - Not Mandatory iOrderSequence - Not Mandatory iAdviceLine - Mandatory iOutboundRun - Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error </pre>

OutboundAdvice.StartDetail

DLL:	whextinhapi
Syntax:	<pre> long OutboundAdvice.StartDetail(long iStartMode, domain whinh.oorg iOrderOrigin, domain tcorno iOrderNumber, domain tcwset iOrderSet, domain tcpono iOrderLine, domain tcpono iOrderSequence, domain tcpono iAdvice, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>

Usage:	Expl: This function starts the detail session Outbound Advice (whinh4525m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variable form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iOrderOrigin iOrderNumber iOrderSet iOrderLine iOrderSequence iAdvice - The primary key fields must refer to an existing Outbound Advice Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - Session started - Error.

OutboundAdvice.StartGenerate

DLL:	whextinhapi
Syntax:	<pre>long OutboundAdvice.StartGenerate(long iStartMode, boolean iIgnoreSelectionFields, domain whinh.oorg iFromOrderOrigin, domain tcorno iFromOrderNumber, domain tcwset iFromOrderSet, domain tcpono iFromOrderLine, domain whinh.oorg iToOrderOrigin, domain tcorno iToOrderNumber, domain tcwset iToOrderSet, domain tcpono iToOrderLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl This function starts the session Generate Outbound Advice (whinh4201m000). Depending on the main table of the calling session, Non-Consecutive Record Selection (NCRS) is used.

	When the main table is: Warehousing Orders (whinh200) or Outbound Order Lines (whinh220) or Planned Loads/Shipments (whinh480) then the range selections will be imported based on the selected records in the parent session and the selection related fields will be ignored. Input: iStartMode Not Used. iIgnoreSelectionFields If true, (empty) From/To selection fields will not be filled in the session. iFromOrderOrigin From Order Origin selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iFromOrderNumber From Order Number selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iFromOrderSet From Order Set selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iFromOrderLine From Order Line selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToOrderOrigin To Order Origin selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToOrderNumber To Order Number selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToOrderSet To Order Set selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) iToOrderLine To Order Line selection field is filled with this value. (when iIgnoreSelectionFields is false and NCRS is not applicable) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

OutboundAdvice.StartOverview

DLL:	whextinhapi
Syntax:	<pre> long OutboundAdvice.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain whinh.oorg iOrderOrigin, domain tcorno iOrderNumber, domain tcwset iOrderSet, domain tcpono iOrderLine, domain tcpono iOrderSequence, domain tcpono iAdvice, domain whinh.btno iRun, domain tcmcs.st20m iActivity mb, domain tcitem iItem, domain tccwar iWarehouse, domain whloca iLocation, domain tcclot iLot, domain tcinvl.date iInventoryDate, domain tcyesno iPicked, domain tccuni iUnit, ref domain whinh.oorg oOrderOrigin, ref domain tcorno oOrderNumber, ref domain tcwset oOrderSet, ref domain tcpono oOrderLine, ref domain tcpono oOrderSequence, ref domain tcpono oAdvice, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl This function starts the overview session Outbound Advice (whinh4525m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byOrder": data is displayed by Order session will be started on index 1 view field: None "byRun": data is displayed by Run session will be started on index 4 view field: Run "byActivity": data is displayed by Activity session will be started on index 6 view field: Activity "byItem": data is displayed by Item

	session will be started on index 5 view field: Item "byWarehouse": data is displayed by Warehouse session will be started on index 2 view field: Warehouse iSessionIndex Specifies the table index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iOrderOrigin The Order Origin to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 1 (or iStartFilter = "byOrder") iOrderNumber The Order Number to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 1 (or iStartFilter = "byOrder") iOrderSet The Order Set to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 1 (or iStartFilter = "byOrder") iOrderLine The Order Line to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 1 (or iStartFilter = "byOrder") iOrderSequence The Order Sequence to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 1 (or iStartFilter = "byOrder") iAdvice The Advice to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 1 (or iStartFilter = "byOrder") iRun The Run to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 4 (or iStartFilter = "byRun") iActivity The Activity to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 6 (or iStartFilter = "byActivity") iItem The Item to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 5 (or iStartFilter = "byItem") iWarehouse The Warehouse to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 2 (or iStartFilter = "byWarehouse") iLocation Not mandatory iLot Not mandatory iInventoryDate Not mandatory iPicked Not mandatory iUnit

	Not mandatory Output: for iStartMode MODAL: - oOrderOrigin - Order Origin of selected record - oOrderNumber - Order Number of selected record - oOrderSet - Order Set of selected record - oOrderLine - Order Line of selected record - oOrderSequence - Order Sequence of selected record - oAdvice - Advice of selected record - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error.

Public Interfaces for OutboundRun

The following functions are available:

[OutboundRun.HandlePickingList](#)

[OutboundRun.StartAutomaticOutboundProcessing](#)

OutboundRun.HandlePickingList

DLL:	whextinhapi
	This function is available from KB2325172 .
Syntax:	<pre>long OutboundRun.HandlePickingList(domain whinh.btno iOutboundRun, domain whinh.spla iAction, domain tcmcs.str15 iDevice, long iProcessingOptionSet, ref boolean oDataProcessed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface will handle the picking list for the iOutboundRun and iAction using the defaults or options as provided in the iProcessingOptionSet. The opening/closing of the related report is handled within this public interface.

Pre: There should be no pending logical transaction before calling this function.

Post: No need to commit or abort the process, that is handled within the function.

Input: iOutboundRun Mandatory
iAction Mandatory
iDevice Mandatory
iProcessingOptionSet Optional, if 0, the default options are applied.
A Processing Option Set can be created via a call to function ProcessingOptionSet.Create().

Processing Options have a direct relationship with the form fields on session Generate Picking List (whinh4415m000) and are not explained in further detail here.

Please refer to the session help for additional information.

Processing Options that are set while a required Implemented Software Component is not available are ignored.

Explanation about setting of defaults:

Run Value: Value of the field stored in the Outbound Run is taken as default value.

Minimum Value: Minimum value of domain is taken as default value.

Maximum Value: Maximum value of domain is taken as default value.

whinh403 Value: If SortOrderBy field is filled then value is retrieved from table Sort Options for Picking List (whinh403), otherwise the default value is tcyesno.no.

NAME	TYPE	DEFAULT
AllAdviceLines	domain tcyesno	tcyesno.no
ExcludePickedAdviceLines	domain tcyesno	tcyesno.no
OrderOriginFrom	domain whinh.oorg	Run Value
OrderOriginTo	domain whinh.oorg	Run Value
OrderGroupFrom	domain tcpdno	Run Value
OrderGroupTo	domain tcpdno	Run Value
OrderNumberFrom	domain tcorno	Run Value
OrderNumberTo	domain tcorno	Run Value
OrderSetFrom	domain tcwset	Run Value
OrderSetTo	domain tcwset	Run Value
OrderLineFrom	domain tcpono	Run Value
OrderLineTo	domain tcpono	Run Value
OrderLineSequenceFrom	domain tcpono	Minimum Value
OrderLineSequenceTo	domain tcpono	Maximum Value
AdviceLineFrom	domain tcpono	Minimum Value
AdviceLineTo	domain tcpono	Maximum Value
SiteFrom	domain tcsite	Run Value
SiteTo	domain tcsite	Run Value
WarehouseFrom	domain tccwar	Run Value
WarehouseTo	domain tccwar	Run Value
ZoneFrom	domain whwmd.zone	Minimum Value
ZoneTo	domain whwmd.zone	Maximum Value
LocationFrom	domain whloca	Minimum Value
LocationTo	domain whloca	Maximum Value
RowFrom	domain whwmd.strt	Minimum Value
RowTo	domain whwmd.strt	Maximum Value
LevelFrom	domain whwmd.coln	Minimum Value
LevelTo	domain whwmd.coln	Maximum Value
BinFrom	domain whwmd.rack	Minimum Value
BinTo	domain whwmd.rack	Maximum Value
PlannedDeliveryDateFrom	domain tcdate	Run Value
PlannedDeliveryDateTo	domain tcdate	Run Value

ShipmentReferenceFrom	domain tcrefs	Minimum Value
ShipmentReferenceTo	domain tcrefs	Maximum Value
CustomerOrderFrom	domain tccorn	Minimum Value
CustomerOrderTo	domain tccorn	Maximum Value
ItemFrom	domain tcitem	Run Value
ItemTo	domain tcitem	Run Value
LoadFrom	domain whinh.load	Minimum Value
LoadTo	domain whinh.load	Maximum Value
ShipmentFrom	domain whinh.shpm	Minimum Value
ShipmentTo	domain whinh.shpm	Maximum Value
HandlingUnitFrom	domain whhuid	Minimum Value
HandlingUnitTo	domain whhuid	Maximum Value
SortOrderBy1	domain tcfdnm.c	Minimum Value
SortDescend1	domain tcyesno	whinh403 Value
SortNewMission1	domain tcyesno	whinh403 Value
SortOrderBy2	domain tcfdnm.c	Minimum Value
SortDescend2	domain tcyesno	whinh403 Value
SortNewMission2	domain tcyesno	whinh403 Value
SortOrderBy3	domain tcfdnm.c	Minimum Value
SortDescend3	domain tcyesno	whinh403 Value
SortNewMission3	domain tcyesno	whinh403 Value
SortOrderBy4	domain tcfdnm.c	Minimum Value
SortDescend4	domain tcyesno	whinh403 Value
SortNewMission4	domain tcyesno	whinh403 Value
SortOrderBy5	domain tcfdnm.c	Minimum Value
SortDescend5	domain tcyesno	whinh403 Value
SortNewMission5	domain tcyesno	whinh403 Value
SortOrderBy6	domain tcfdnm.c	Minimum Value
SortDescend6	domain tcyesno	whinh403 Value
SortNewMission6	domain tcyesno	whinh403 Value
SortOrderBy7	domain tcfdnm.c	Minimum Value
SortDescend7	domain tcyesno	whinh403 Value
SortNewMission7	domain tcyesno	whinh403 Value
SortOrderBy8	domain tcfdnm.c	Minimum Value
SortDescend8	domain tcyesno	whinh403 Value
SortNewMission8	domain tcyesno	whinh403 Value
SortOrderBy9	domain tcfdnm.c	Minimum Value
SortDescend9	domain tcyesno	whinh403 Value
SortNewMission9	domain tcyesno	whinh403 Value
SortOrderBy10	domain tcfdnm.c	Minimum Value
SortDescend10	domain tcyesno	whinh403 Value
SortNewMission10	domain tcyesno	whinh403 Value
PrintNewPagePerOrder	domain tcyesno	tcyesno.no
PrintProject	domain tcyesno	tcyesno.no
PrintLots	domain tcyesno	tcyesno.no
PrintActualInventoryQuantity	domain tcyesno	tcyesno.no
PrintSerials	domain tcyesno	tcyesno.no
PrintOwnership	domain tcyesno	tcyesno.no
PrintItemStorageConditions	domain tcyesno	tcyesno.no
PrintLocationStorageConditions	domain tcyesno	tcyesno.no
SortOption	domain whsb.loolsc	whsb.loolsc.location
ReprintPickingMissionFrom	domain whinh.picm	Minimum Value
ReprintPickingMissionTo	domain whinh.picm	Maximum Value
ReprintPickingSequenceFrom	domain whinh.lino	Minimum Value
ReprintPickingSequenceTo	domain whinh.lino	Maximum Value
NumberOfCopies	domain tcsrno	0
PrintRunAsBarcode	domain tcyesno	tcyesno.no
BarcodeType	domain tcbctype	tcbctype.not.appl
BarcodeHeight	domain tcmcs.byte1	Minimum Value
ReportName	domain tcmcs.str16	Empty String
ReportName only needs to be filled for customized reports, otherwise		

	the report related to the SortOption is automatically used. ReportName must start with an "r", e.g. "rwhinh441511001" Output: oDataProcessed - true: Picking List Generated/Printed. false: Nothing Generated/Printed. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

OutboundRun.StartAutomaticOutboundProcessing

DLL:	whextinhapi
Syntax:	<pre>long OutboundRun.StartAutomaticOutboundProcessing(domain whinh.btno iOutboundRun, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface will perform the next automatic activity of the given outbound run This public interface will execute the following steps: 1) Clear any prior notifications done for the automatic outbound process, within the current process. 2) Notify the outbound run for automatic processing 3) Start the automatic outbound process for the notified outbound run The way to implement this public interface is to make sure that the next activity is set to automatic. This can be already set properly based on the order type setup, or can be achieved by using public interface OutboundOrderLineActivity.ModifyAutomaticProcessing() for the activity that should be automatically processed. See below code example: <pre>--- db.retry.point() * Set the activity to automatic ret = OutboundOrderLineActivity.ModifyAutomaticProcessing(..) if ret <> 0 then abort.transaction() return(ret) endif commit.transaction() * Start the automatic order process ret.val = OutboundRun.StartAutomaticOutboundProcessing(..)</pre>

	<pre> db.retry.point() * Set the activity to be not automatic ret = OutboundOrderLineActivity.ModifyAutomaticProcessing(..) if ret <> 0 then abort.transaction() return(ret) endif commit.transaction() return(ret.val) ---</pre> This public interface will start the next automatic activity for the related outbound run. This public interface will stop the automatic process when the next activity is not set to automatic. This function must not be called within a logical transaction as this function will have it's own transaction handling. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iOutboundRun - Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Public Interfaces for QuarantineInventory

The following functions are available:

[QuarantineInventory.Move](#)

[QuarantineInventory.StartAutomaticInboundProcessing](#)

[QuarantineInventory.StartDetail](#)

[QuarantineInventory.StartDisposition](#)

[QuarantineInventory.StartOverview](#)

QuarantineInventory.Move

DLL:	whextwmdapi
Syntax:	<pre> long QuarantineInventory.Move(domain tcorno iQuarantineIdentifier, domain tccwar iWarehouse, domain tcitem iItem, domain whloca iDestinationLocation, ref boolean oMoved, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface will Move all quarantine inventory of a Quarantine Identifier to another quarantine location. Moving the quarantine inventory is allowed if still unprocessed disposition line(s) exist. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iQuarantineIdentifier - The Quarantine ID (mandatory) iWarehouse - Warehouse (mandatory) iItem - Item (mandatory) iDestinationLocation - The destination quarantine location (mandatory). Output: oMoved - Indicates whether moving succeeded or not true: Inventory successfully moved false: Inventory not successfully moved (or not found) oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Quarantine Inventory has been moved successfully <> 0 - Error

QuarantineInventory.StartAutomaticInboundProcessing

DLL:	whextinhapi
Syntax:	<pre> long QuarantineInventory.StartAutomaticInboundProcessing(domain tcorno iQuarantineInventory, domain whinh.shpm iReceipt, domain tcpono iReceiptLine, domain whhuid iHandlingUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will start the processing of the next automatic activity after the receipt line has been confirmed which is related to Quarantine Inventory.

This public interface will generate inbound advices when this is the next automatic activity of the related inbound order line. When there is already an advice present which must be putaway, please use one of the following public interfaces:
 InboundAdvice.StartAutomaticInboundProcessing
 InboundRun.StartAutomaticInboundProcessing

This public interface will execute the following steps:

- 1) Clear any prior notifications done for the automatic inbound process, within the current process.
- 2) Notify the quarantine inventory, receipt line and the potential handling unit for automatic processing
- 3) Start the automatic inbound process for the notified Quarantine Inventory, Receipt Line and potential handling unit

The way to implement this public interface is to make sure that the next activity is set to automatic. This can be already set properly based on the order type setup, or can be achieved by using public interface
 InboundOrderLineActivity.ModifyAutomaticProcessing() for the activity that should be automatically processed.
 See below code example:

```
---
db.retry.point()

/* Set the activity to automatic
ret = InboundOrderLineActivity.ModifyAutomaticProcessing(..)

if ret <> 0 then
    abort.transaction()
    return(ret)
endif

commit.transaction()

/* Start the automatic order process
ret.val = QuarantineInventory.StartAutomaticInboundProcessing(.)

db.retry.point()

/* Set the activity to be not automatic
ret = InboundOrderLineActivity.ModifyAutomaticProcessing(..)

if ret <> 0 then
    abort.transaction()
    return(ret)
endif

commit.transaction()

return(ret.val)
---
```

This public interface will be able to automatically generate an inbound advice for the given quarantine inventory, receipt line and potential handling unit. This public interface will stop the automatic process when the next activity is not set to automatic.

	This function must not be called within a logical transaction as this function will have it's own transaction handling. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iQuarantineInventory - Mandatory iReceipt - Mandatory iReceiptLine - Mandatory iHandlingUnit - Not Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

QuarantineInventory.StartDetail

DLL:	whextwmdapi
Syntax:	<pre>long QuarantineInventory.StartDetail(long iStartMode, domain tcorno iQuarantineIdentifier, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface starts session Quarantine Inventory (whwmd2171m000) in Detail Mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iQuarantineIdentifier - The Quarantine ID of the disposition line to be processed (mandatory). Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in

	Return: 0 <> 0	Exception to get all relevant information. - Session started - Error

QuarantineInventory.StartDisposition

DLL:	whextwmdapi This function is available from KB3623004.																															
Syntax:	<pre>long QuarantineInventory.StartDisposition(long iStartMode, domain tcorno iQuarantineIdentifier, domain tcmcs.long iDispositionLine, domain whhuid iHandlingUnit, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>																															
Usage:	Expl: This public interface starts session Quarantine Inventory Disposition (whwmd2272m200) in Detail Mode. Input: iStartMode - Not used - Session is always started in MODAL mode. iQuarantineIdentifier - Mandatory iDispositionLine - Optional, if 0, then all disposition lines will be processed. But will be ignored when DispositionLineArray or HandlingUnitArray of iProcessingOptionSet is filled. iHandlingUnit - Optional, will be ignored when iDispositionLine is filled and when DispositionLineArray or HandlingUnitArray of iProcessingOptionSet is filled. iProcessingOptionSet Optional, if 0, the default options are applied. Processing Options have a direct relationship with the form fields on session Disposition (whwmd2272m200) and are not explained in further detail here. Please refer to the session help for additional information. Processing Options that are set while a required Implemented Software Component is not available are ignored. <table> <thead> <tr> <th>NAME</th><th>TYPE</th><th>DEFAULT</th></tr> </thead> <tbody> <tr> <td>Disposition</td><td>whwmd.disp</td><td>whwmd.disp.awaiting</td></tr> <tr> <td>Reason</td><td>tccdis</td><td>Empty String</td></tr> <tr> <td>ToItem</td><td>tcitem</td><td>Empty String</td></tr> <tr> <td>ToLot</td><td>tcclot</td><td>Empty String</td></tr> <tr> <td>Responsibility</td><td>whwmd.resp</td><td>whwmd.resp.not.appl</td></tr> <tr> <td>NonConformanceReference</td><td>tcrefa</td><td>Empty String</td></tr> <tr> <td>DispositionQuantity</td><td>tcqstl</td><td>0.0</td></tr> <tr> <td>DispositionUnit</td><td>tccuni</td><td>Empty String</td></tr> <tr> <td>OrderSeries</td><td>tcseri</td><td>Empty String</td></tr> </tbody> </table>		NAME	TYPE	DEFAULT	Disposition	whwmd.disp	whwmd.disp.awaiting	Reason	tccdis	Empty String	ToItem	tcitem	Empty String	ToLot	tcclot	Empty String	Responsibility	whwmd.resp	whwmd.resp.not.appl	NonConformanceReference	tcrefa	Empty String	DispositionQuantity	tcqstl	0.0	DispositionUnit	tccuni	Empty String	OrderSeries	tcseri	Empty String
NAME	TYPE	DEFAULT																														
Disposition	whwmd.disp	whwmd.disp.awaiting																														
Reason	tccdis	Empty String																														
ToItem	tcitem	Empty String																														
ToLot	tcclot	Empty String																														
Responsibility	whwmd.resp	whwmd.resp.not.appl																														
NonConformanceReference	tcrefa	Empty String																														
DispositionQuantity	tcqstl	0.0																														
DispositionUnit	tccuni	Empty String																														
OrderSeries	tcseri	Empty String																														

	<pre> OrderType tccotp Empty String BuyFromBP tccom.bpid Empty String ShipFromBP tccom.bpid Empty String DispositionLineArray domain ttjson 0 HandlingUnitArray domain ttjson 0 </pre> JSON Object DispositionLineArray has the following structure: <pre> "DispositionLineArray": [{ "Line": 10 }, { "Line": 30 }] </pre> This structure can be created with the following code: <pre> DispositionLineArray = Json.newArray() DispositionLine = Json.newObject() Json.setNumber(DispositionLine, "Line", 10) Json.add(DispositionLineArray, DispositionLine) DispositionLine = Json.newObject() Json.setNumber(DispositionLine, "Line", 30) Json.add(DispositionLineArray, DispositionLine) </pre> HandlingUnitArray will be ignored when DispositionLineArray is filled. JSON Object HandlingUnitArray has the following structure: <pre> "HandlingUnitArray": [{ "HandlingUnit": "HU100" }, { "HandlingUnit": "HU200" }] </pre> This structure can be created with the following code: <pre> HandlingUnitArray = Json.newArray() HandlingUnit = Json.newObject() Json.setString(HandlingUnit, "HandlingUnit", "HU100") Json.add(HandlingUnitArray, HandlingUnit) HandlingUnit = Json.newObject() Json.setString(HandlingUnit, "HandlingUnit", "HU200") Json.add(HandlingUnitArray, HandlingUnit) </pre> Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

QuarantineInventory.StartOverview

DLL:	whextwmdapi
Syntax:	<pre> long QuarantineInventory.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iQuarantineIdentifier, domain tccwar iWarehouse, domain whloca iLocation, domain tcitem iItem, domain whinh.oorg iOrderOrigin, domain tcorno iOrder, domain tcwset iOrderSet, domain tcpono iOrderLine, domain tcpono iOrderSequence, ref domain tcorno oQuarantineIdentifier, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface starts session Quarantine Inventory (whwmd2171m000) in Overview Mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byQuarantineId": data is displayed by Quarantine ID session will be started on index 1 view fields: Quarantine ID "byWarehouseLocation": data is displayed by Warehouse and Location session will be started on index 2 view fields: Warehouse, Location "byItemWarehouseLocation": data is displayed by Item, Warehouse and Location session will be started on index 3 view fields: Item, Warehouse, Location "byOrderLine": data is displayed by Order Line session will be started on index 4 view fields: Order Origin, Order, Order Set, Order Line, Order Sequence iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a iStartFilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session.

	iQuarantineIdentifier Mandatory if iStartMode = MODELESS and iSessionIndex = 1 (or iStartFilter = "byQuarantineId") iWarehouse Not mandatory iLocation Not mandatory iItem Not mandatory iOrderOrigin Mandatory if iStartMode = MODELESS and iSessionIndex = 4 (or iStartFilter = "byOrderLine") iOrder Not mandatory iOrderSet Not mandatory iOrderLine Not mandatory iOrderSequence Not mandatory Output: oQuarantineIdentifier - Quarantine ID of the selected record, if iStartMode = MODAL. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

Public Interfaces for QuarantineInventoryDisposition

The following functions are available:

[QuarantineInventoryDisposition.StartDetail](#)

[QuarantineInventoryDisposition.StartOverview](#)

QuarantineInventoryDisposition.StartDetail

DLL:	whextwmdapi
Syntax:	long QuarantineInventoryDisposition.StartDetail(

	<pre> long iStartMode, domain tcorno iQuarantineIdentifier, domain tcmcs.long iDispositionLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface starts session Quarantine Inventory Disposition(whwmd2172m000) in Detail Mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iQuarantineIdentifier - The Quarantine ID of the disposition line to be processed (mandatory). iDispositionLine - The Disposition Line to be processed (mandatory). Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

QuarantineInventoryDisposition.StartOverview

DLL:	whextwmdapi
Syntax:	<pre> long QuarantineInventoryDisposition.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iQuarantineIdentifier, domain tcmcs.long iDispositionLine, domain tccwar iWarehouse, domain whloca iLocation, domain tcitem iItem, domain whinh.oorg iOrderOrigin, domain tcorno iOrder, domain tcwset iOrderSet, domain tcpono iOrderLine, </pre>

	domain tcpono iOrderSequence, domain whinh.dorg iDispositionOrigin, domain tcorno iDispositionOrder, domain whinh.shpm iReceipt, domain tcpono iReceipt.line, ref domain tcorno oQuarantineIdentifier, ref domain tcmcs.long oDispositionLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This public interface starts session Quarantine Inventory Disposition(whwmd2172m000) in Overview Mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byQuarantineId": data is displayed by Quarantine ID and Disposition Line session will be started on index 1 view fields: Quarantine ID "byWarehouseLocation": data is displayed by Warehouse and Location session will be started on index 2 view fields: Warehouse, Location "byItemWarehouseLocation": data is displayed by Item, Warehouse and Location session will be started on index 3 view fields: Item, Warehouse, Location "byOrderLine": data is displayed by Order Line session will be started on index 4 view fields: Order Origin, Order, Order Set, Order Line, Order Sequence "byDispositionOrder": data is displayed by Disposition Order session will be started on index 5 view fields: Disposition Origin "byReceiptLine": data is displayed by Receipt Line session will be started on index 6 view fields: Receipt, Receipt Line iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a iStartFilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iQuarantineIdentifier Mandatory if iStartMode = MODELESS and iSessionIndex = 1 (or iStartFilter = "byQuarantineId") iDispositionLine Not mandatory

	<pre> iWarehouse Mandatory if iStartMode = MODELESS and iSessionIndex = 2 (or iStartFilter = "byWarehouseLocation" iStartFilter = "byItemWarehouseLocation") iLocation Not mandatory iItem Mandatory if iStartMode = MODELESS and iSessionIndex = 3 (or iStartFilter = "byItemWarehouseLocation") iOrderOrigin Mandatory if iStartMode = MODELESS and iSessionIndex = 4 (or iStartFilter = "byOrderLine") iOrder Not mandatory iOrderSet Not mandatory iOrderLine Not mandatory iOrderSequence, Not mandatory iDispositionOrigin Mandatory if iStartMode = MODELESS and iSessionIndex = 5 (or iStartFilter = "byDispositionOrder") iDispositionOrder Not mandatory iReceipt Not mandatory iReceipt.line Not mandatory Output: oQuarantineIdentifier - Quarantine ID of the selected record oDispositionLine - Disposition Line of the selected record oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error </pre>

Public Interfaces for QuarantineInventoryDispositionLine

The following functions are available:

[QuarantineInventoryDispositionLine.GenerateInboundAdvice](#)

[QuarantineInventoryDispositionLine.Move](#)

[QuarantineInventoryDispositionLine.Process](#)

QuarantineInventoryDispositionLine.GenerateInboundAdvice

DLL:	whextwmdapi This function is available from KB3590515 .
Syntax:	<pre> long QuarantineInventoryDispositionLine.GenerateInboundAdvice(domain tcorno iQuarantineIdentifier, domain tcmcs.long iDispositionLine, domain whloca iPreferenceLocation, boolean iPutAway, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface generates an inbound advice for the quarantine inventory disposition classified as 'Use As Is' or 'No Fault Found'. Optionally, it can be put away directly. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: - iQuarantineIdentifier - Quarantine ID (Mandatory) - iDispositionLine - Disposition Line (Mandatory) - iPreferenceLocation - Advise will preferably be generated to this location. (Optional) - iPutAway - Directly Put Away (Mandatory) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Disposition line has been advised successfully. <> 0 - Error.

QuarantineInventoryDispositionLine.Move

DLL:	whextwmdapi
Syntax:	<pre> long QuarantineInventoryDispositionLine.Move(domain tcorno iQuarantineIdentifier, domain tcmcs.long iDispositionLine, domain tccwar iWarehouse, domain tcitem iItem, </pre>

	domain whloca iDestinationLocation, domain tcqstl iQuantityInStorageUnit, domain tccuni iStorageUnit, ref boolean oMoved, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This Public Interface will Move an individual quarantine disposition line to another quarantine location. Moving the disposition is allowed if still unprocessed and no Handling Unit is linked. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iQuarantineIdentifier - The Quarantine ID (mandatory) iDispositionLine - Disposition Line Number (mandatory) iWarehouse - Warehouse (mandatory) iItem - Item (mandatory) iDestinationLocation - The destination quarantine location. (mandatory). iQuantityInStorageUnit - Disposition line quantity to be moved in storage unit (mandatory). If this quantity is less than the disposition line quantity, the line will be split and only this quantity will be moved. iStorageUnit - Unit of the disposition line (mandatory). Output: oMoved - Indicates whether moving succeeded or not true: Inventory successfully moved false: Inventory not successfully moved (or not found) oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Quarantine Inventory has been moved successfully <> 0 - Error

QuarantineInventoryDispositionLine.Process

DLL:	whextwmdapi
Syntax:	long QuarantineInventoryDispositionLine.Process(domain tcorno iQuarantineIdentifier, domain tcmcs.long iDispositionLine, ref domain tcorno oAdjustmentOrder, ref domain tcorno oDispositionOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This Public Interface will process a given Quarantine Inventory Disposition Line. The line can be processed using this function if: - It is not processed yet - The Disposition field is not 'Awaiting Disposition' - The disposition quantity is not zero - No handling unit is linked Pre: db.retry.point() Post: abort.transaction() or commit.transaction()

	Input: iQuarantineIdentifier - The Quarantine ID of the disposition line to be processed (mandatory). iDispositionLine - The Disposition Line to be processed (mandatory). Output: oAdjustmentOrder - The created adjustment order when processing a disposition line with disposition Scrap. oDispositionOrder - The created disposition order when processing a disposition line with disposition Return To Vendor (RTV), Rework (to new/existing specification) or Reclassify. In case of RTV: the generated PUR return order In case of Rework: the generated production rework order In case of Reclassify: the generated transfer order oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Disposition line has been processed successfully <> 0 - Error

Public Interfaces for QuarantineInventoryHandlingUnit

The following functions are available:

[QuarantineInventoryHandlingUnit.Move](#)

[QuarantineInventoryHandlingUnit.Process](#)

QuarantineInventoryHandlingUnit.Move

DLL:	whextwmdapi
Syntax:	<pre>long QuarantineInventoryHandlingUnit.Move(domain tcorno iQuarantineIdentifier, domain tccwar iWarehouse, domain tcitem iItem, domain whhuid iHandlingUnit, domain whloca iDestinationLocation, ref boolean oMoved, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This Public Interface will Move a given Handling Unit in Quarantine Inventory to another quarantine location.

	Moving the handling unit is allowed if - The handling unit is not processed yet. - Active Handling Unit Process data (whwmd273) exists Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iQuarantineIdentifier - The Quarantine ID of the handling unit to be moved (mandatory). iWarehouse - Warehouse (mandatory) iItem - Item (mandatory) iHandlingUnit - The Handling Unit to be moved (mandatory). iDestinationLocation - The destination quarantine location to move the handling unit to (mandatory). Output: oMoved - Indicates whether moving succeeded or not true: Inventory successfully moved false: Inventory not successfully moved (or not found) oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Handling Unit has been moved successfully <> 0 - Error

QuarantineInventoryHandlingUnit.Process

DLL:	whextwmdapi
Syntax:	<pre>long QuarantineInventoryHandlingUnit.Process(domain tcorno iQuarantineIdentifier, domain whhuid iHandlingUnit, ref domain tcorno oAdjustmentOrder, ref domain tcorno oDispositionOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This Public Interface will process a given Handling Unit in Quarantine Inventory. Processing the handling unit is allowed if - The handling unit is not processed yet. - Active Handling Unit Process data (whwmd273) exists - The disposition (whwmd273.disp) is not 'Awaiting disposition' Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iQuarantineIdentifier - The Quarantine ID of the handling unit to be processed (mandatory). iHandlingUnit - The Handling Unit to be processed (mandatory). Output: oAdjustmentOrder - The created adjustment order when processing a disposition line with disposition Scrap. oDispositionOrder - The created disposition order when processing a disposition line with disposition Return To Vendor (RTV), Rework (to new/existing specification) or Reclassify. In case of RTV: the generated PUR return order In case of Rework: the generated production rework order In case of Reclassify: the generated transfer order oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these

	are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Handling Unit has been processed succesfully <> 0 - Error

Public Interfaces for ProjectPeggedInventory

The following functions are available:

[ProjectPeggedInventory.StartDetail](#)

[ProjectPeggedInventory.StartOverview](#)

ProjectPeggedInventory.StartDetail

DLL:	whextwmdapi This function is available from KB2283036.
Syntax:	<pre>long ProjectPeggedInventory.StartDetail(long iStartMode, domain tccwar iWarehouse, domain tccprj iProject, domain tcitem iItem, domain tcuef.effn iEffectivityUnit, domain tcpdm.cspa iElement, domain tcpdm.cact iActivity, domain tcptc.cstl iExtension, domain tccpcp iCostComponent, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface starts session Project Pegged Inventory (whwmd2560m000) in Detail Mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error

	will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: - iWarehouse - iProject - iItem - iEffectivityUnit - iElement - iActivity - iExtension - iCostComponent Output: oExceptionMessage oExceptionID Return: 0 <> 0	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Session started - Error

ProjectPeggedInventory.StartOverview

DLL:	whextwmdapi
	This function is available from KB2283036 .
Syntax:	<pre>long ProjectPeggedInventory.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccwar iWarehouse, domain tccprj iProject, domain tcitem iItem, domain tcuef.effn iEffectivityUnit, domain tcpdm.cspa iElement, domain tcpdm.cact iActivity, domain tcptc.cstl iExtension, domain tccpcp iCostComponent, ref domain tccwar oWarehouse, ref domain tccprj oProject, ref domain tcitem oItem, ref domain tcuef.effn oEffectivityUnit, ref domain tcpdm.cspa oElement, ref domain tcpdm.cact oActivity, ref domain tcptc.cstl oExtension, ref domain tccpcp oCostComponent, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface starts session Project Pegged Inventory (whwmd2560m000) in Overview Mode. Input: iStartMode Specifies the start mode for the session.

	Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byWarehouseProject": data is displayed by Warehouse and Project session will be started on index 1 view field: Warehouse "byWarehouseItemEffectivityUnit": data is displayed by Warehouse, Item and Effectivity Unit session will be started on index 2 view field: Warehouse "byWarehouseProjectPeg": data is displayed by Warehouse and Project Peg session will be started on index 3 view field: Warehouse "byItemEffectivityUnitProject": data is displayed by Item, Effectivity Unit and Project session will be started on index 4 view field: Item "byItemEffectivityUnitWarehouse": data is displayed by Item, Effectivity Unit and Warehouse session will be started on index 5 view field: Item "byProjectWarehouse": data is displayed by Project and Warehouse session will be started on index 6 view field: Project "byProjectPeg": data is displayed by Project Peg session will be started on index 7 view field: Project iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a iStartFilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iWarehouse Mandatory if iStartMode = MODELESS and i.session.index = 1, 2 or 3 (or startfilter = "byWarehouseProject" or startfilter = "byWarehouseItemEffectivityUnit" or startfilter = "byWarehouseProjectPeg") iProject Mandatory if iStartMode = MODELESS and i.session.index = 6 or 7 (or startfilter = "byProjectWarehouse" or startfilter = "byProjectPeg") iItem Mandatory if iStartMode = MODELESS and

	<pre> i.session.index = 4 or 5 (or startfilter = "byItemEffectivityUnitProject" or startfilter = "byItemEffectivityUnitWarehouse") iEffectivityUnit Optional iElement Optional iActivity Optional iExtension Optional iCostComponent Optional </pre>
Output:	<pre> oWarehouse - Warehouse of the selected record, if iStartMode = MODAL. oProject - Project of the selected record, if iStartMode = MODAL. oItem - Item of the selected record, if iStartMode = MODAL. oEffectivityUnit - Effectivity Unit of the selected record, if iStartMode = MODAL. oElement - Element of the selected record, if iStartMode = MODAL. oActivity - Activity of the selected record, if iStartMode = MODAL. oExtension - Extension of the selected record, if iStartMode = MODAL. oCostComponent - Cost Component of the selected record, if iStartMode = MODAL. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. </pre>
Return: 0	- Session started
<> 0	- Error

Public Interfaces for OutboundOrderLineActivity

The following functions are available:

[OutboundOrderLineActivity.Create](#)

[OutboundOrderLineActivity.ModifyAutomaticProcessing](#)

OutboundOrderLineActivity.Create

DLL:	whextinhapi
	This function is available from KB2271017 .
Syntax:	<pre> long OutboundOrderLineActivity.Create(domain whinh.oorg iOrderOrigin, domain tcorno iOrderNumber, domain tcpono iOrderLine, domain tcpono iOrderSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function will create the outbound order line activities for an outbound order line. Pre: db.retry.point should be set Post: commit or abort transaction depending on the return value. Input: - iOrderOrigin - Mandatory - iOrderNumber - Mandatory - iOrderLine - Not Mandatory (inbound line can be 0) - iOrderSequence - Not Mandatory (inbound sequence can be 0) Output: - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

OutboundOrderLineActivity.ModifyAutomaticProcessing

DLL:	whextinhapi
Syntax:	<pre> long OutboundOrderLineActivity.ModifyAutomaticProcessing(domain whinh.oorg iOrderOrigin, domain tcorno iOrderNumber, domain tcpono iOrderLine, domain tcpono iOrderSequence, domain whinh.prtty iProcedureType, domain tcmcs.st20m iActivity mb, domain tcyesno iAutomatic, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function will update the automatic attribute of the outbound order line activity that is provided. The automatic field will only be updated if the activity is applicable and the value of the automatic attribute is different than the value passed in variable iAutomatic. Depending on the order type certain activities can be set to not applicable, and as a result the automatic flag cannot be updated.

	<pre> Pre: db.retry.point should be set Post: commit or abort transaction depending on the return value. Input: iOrderOrigin - Mandatory iOrderNumber - Mandatory iOrderLine - Not Mandatory (inbound line can be 0) iOrderSequence - Not Mandatory (inbound sequence can be 0) iProcedureType - Mandatory Allowed Values: Outbound Procedure (whinh.prty.issue.advice) Shipment Procedure (whinh.prty.issue.shipment) iActivity - Mandatory Allowed Values: The values are depending on the procedure type. Outbound Procedure: "whinh4201m000" - Generate Outbound Advice "whinh4202m000" - Release Outbound Advice "whinh4415m000" - Print Picking List "whinh4525m100" - Confirm Picking List Shipment Procedure: "whinh4275m001" - Freeze Shipment "whinh4275m000" - Confirm Shipment "whinh4470m000" - Print Bills of Lading "whinh4475m000" - Print Packing Slip "whinh4476m000" - Print Packing List "whinh4477m000" - Print Delivery Notes "whinh4478m000" - Print Shipping Manifest "whinh4279m000" - Print Pro Forma Invoice iAutomatic - Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error </pre>

Public Interfaces for OutboundOrderLine

The following functions are available:

[OutboundOrderLine.StartAutomaticOutboundProcessing](#)

OutboundOrderLine.StartAutomaticOutboundProcessing

DLL:	whextinhapi
Syntax:	<pre> long OutboundOrderLine.StartAutomaticOutboundProcessing(domain whinh.oorg iOrderOrigin, domain tcorno iOrderNumber, domain tcpono iOrderLine, domain tcpono iOrderSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will perform the next automatic activity of the given outbound order line. This public interface will execute the following steps: 1) Clear any prior notifications done for the automatic outbound process, within the current process. 2) Notify the outbound order line for automatic processing 3) Start the automatic outbound process for the notified Outbound Order Line The way to implement this public interface is to make sure that the next activity is set to automatic. This can be already set properly based on the order type setup, or can be achieved by using public interface OutboundOrderLineActivity.ModifyAutomaticProcessing() for the activity that should be automatically processed. See below code example: <pre> --- db.retry.point() * Set the activity to automatic ret = OutboundOrderLineActivity.ModifyAutomaticProcessing(..) if ret <> 0 then abort.transaction() return(ret) endif commit.transaction() * Start the automatic order process ret.val = OutboundOrderLine.StartAutomaticOutboundProcessing(..) db.retry.point() </pre>

	<pre> * Set the activity to be not automatic ret = OutboundOrderLineActivity.ModifyAutomaticProcessing(..) if ret <> 0 then abort.transaction() return(ret) endif commit.transaction() return(ret.val) ---</pre> This public interface will start the next automatic activity for the related outbound order line. This public interface will stop the automatic process when the next activity is not set to automatic. This function must not be called within a logical transaction as this function will have it's own transaction handling. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iOrderOrigin - Mandatory iOrderNumber - Mandatory iOrderLine - Not Mandatory iOrderSequence - Not Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Public Interfaces for ReceiptLine

The following functions are available:

[ReceiptLine.Confirm](#)

[ReceiptLine.CorrectFinalReceipt](#)

[ReceiptLine.GenerateHandlingUnits](#)

[ReceiptLine.GenerateHandlingUnitsV2](#)

[ReceiptLine.GenerateLots](#)

[ReceiptLine.GenerateSerials](#)[ReceiptLine.GetLanguage](#)[ReceiptLine.Reverse](#)[ReceiptLine.Split](#)[ReceiptLine.SplitForLotsBatches](#)[ReceiptLine.SplitForSerials](#)[ReceiptLine.StartAutomaticInboundProcessing](#)

ReceiptLine.Confirm

DLL:	whextinhapi
Syntax:	<pre> long ReceiptLine.Confirm(domain whinh.shpm iReceipt, domain tcpono iReceiptLine, ref boolean oReceiptLinesConfirmed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function will confirm the given Receipt Line. When zero iReceiptLine is passed all receipt lines of iReceipt will be confirmed. When handling units are linked to the receipt line all handling units will be confirmed automatically. When successful, also the automatic inbound process is started for this receipt line. Be aware that transaction management is handled within this function. Pre: N.a. Post: N.a. Input: iReceipt - Mandatory iReceiptLine - Optional 0 means that all receipt lines will be confirmed. Output: oReceiptLinesConfirmed - One ore more Receipt Lines are confirmed oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

ReceiptLine.CorrectFinalReceipt

DLL:	whextinhapi
	This function is available from KB3565746 .
Syntax:	<pre> long ReceiptLine.CorrectFinalReceipt(domain whinh.shpm iReceipt, domain tcpono iReceiptLine, domain tcyesno iFinalReceipt, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function will correct the final receipt for the receipt line. Be aware that transaction management is handled within this function. Pre: N.a. Post: N.a. Input: iReceipt - Mandatory iReceiptLine - Mandatory iFinalReceipt - Mandatory Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

ReceiptLine.GenerateHandlingUnits

DLL:	whextinhapi
Syntax:	<pre> long ReceiptLine.GenerateHandlingUnits(domain whinh.shpm iReceipt, domain tcpono iReceiptLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl : This function generates handling units for the given receipt line. Generating handling units is allowed when: - Receipt line is open or confirmed but not yet advised; - No handling unit is present on the receipt line; - Handling units are in use in the receipt area for the combination of item and warehouse. If receipt line package definition is known, generated handling unit structure will follow this package definition. Generated handling unit(s) will be stored in the entity Receipt Line Handling Units (whinh324). Pre: db.retry.point() Post: abort.transaction() or commit.transaction()

	Input: iReceipt - Mandatory iReceiptLine - Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

ReceiptLine.GenerateHandlingUnitsV2

DLL:	whextinhapi
Syntax:	<pre>long ReceiptLine.GenerateHandlingUnitsV2(domain whinh.shpm iReceipt, domain tcpono iReceiptLine, domain whhuid iReceiptHandlingUnit, domain whhuid iReceiptLineHandlingUnit, domain whwmd.pkdf iPackageDefinition, domain tcitem iPackagingItem, domain tcqivl iPackagingItemQuantity, ref long oNumberReceiptLineHandlingUnits, ref domain whhuid oReceiptLineHandlingUnitsArray() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl : This function generates handling units for the given receipt line. Generating handling units is allowed when: - Receipt line is open or confirmed but not yet advised; - No handling unit is present on the receipt line; - Handling units are in use in the receipt area for the combination of item and warehouse. If iPackageDefinition is not empty handling unit will be generated using this package definition; otherwise receipt line package definition will be used. Generated handling unit(s) will be stored in the entity Receipt Line Handling Units (whinh324). Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iReceipt - Mandatory iReceiptLine - Mandatory iReceiptHandlingUnit - Receipt Header handling unit. Optional iReceiptLineHandlingUnit - Receipt Line handling unit. Optional iPackageDefinition - Package Definition Optional iPackagingItem - Packaging Item Optional iPackagingItemQuantity - Quantity of packaging items Mandatory if packaging item is not empty Output: oNumberReceiptLineHandlingUnits - number of handling units which are generated for Receipt Line. oReceiptLineHandlingUnitsArray - array with generated handling units. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these

	are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

ReceiptLine.GenerateLots

DLL:	whextinhapi
Syntax:	<pre>long ReceiptLine.GenerateLots(domain whinh.shpm iReceipt, domain tcpono iReceiptLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This Public Interface will generate lot code(s) for a given receipt line. Irrespective of 'Consolidate Stock Points in one Receipt Line' parameter, the lot is always present in the Receipt Line Stock Point Details (whinh320), both for items set as 'Lot in Inventory' as well as items not set as 'Lot in Inventory'. If applicable, existing records with empty lot codes will be filled with the generated Lot Code. Unlike serials, the receipt line will never be split as a result of lot generation (because a serial quantity is always 1) Example: Receipt line, 3 pcs (Consolidate = No) Generate Lot: <pre>----- Receipt Line (whinh312) Stock Point Details (whinh320) ----- 10 LOT1 3 pcs LOT1 3 pcs</pre> Example: Receipt line, 3 pcs (Consolidate = Yes) Generate Lot: <pre>----- Receipt Line (whinh312) Stock Point Details (whinh320) ----- 10 LOT1 3 pcs LOT1 3 pcs</pre> Now suppose the item is Lot Controlled and Serialized: Example: Receipt line, 3 pcs (Consolidate = No) Current values: <pre>----- Receipt Line (whinh312) Stock Point Details (whinh320) ----- 10 SERIAL1 1 pcs SERIAL1 1 pcs 20 SERIAL2 1 pcs SERIAL2 1 pcs 30 SERIAL3 1 pcs SERIAL3 1 pcs Generate Lot: (new lot per receipt line!) 10 SERIAL1 LOT1 1 pcs SERIAL1 LOT1 1 pcs 20 SERIAL2 LOT2 1 pcs SERIAL2 LOT2 1 pcs 30 SERIAL3 LOT3 1 pcs SERIAL3 LOT3 1 pcs</pre> Example: Receipt line, 3 pcs (Consolidate = Yes)

	<pre> current values: ----- Receipt Line (whinh312) Stock Point Details (whinh320) ----- 10 (empty) 3 pcs SERIAL1 1 pcs SERIAL2 1 pcs SERIAL3 1 pcs Generate Lot: 10 (empty) LOT1 3 pcs SERIAL1 LOT1 1 pcs SERIAL2 LOT1 1 pcs SERIAL3 LOT1 1 pcs Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iReceipt - Mandatory iReceiptLine - Mandatory oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error </pre>

ReceiptLine.GenerateSerials

DLL:	whextinhapi
Syntax:	<pre> long ReceiptLine.GenerateSerials(domain whinh.shpm iReceipt, domain tcpono iReceiptLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl : This function generates Serial(s) for the given receipt line. Irrespective of 'Consolidate Stock Points in one Receipt Line' parameter, the Serial is always present in the Receipt Line Stock Point Details (whinh320), both for items set as 'Serial In Inventory' as well as items not set as 'Serial in Inventory'. If applicable, existing records with empty serial numbers will be filled. Otherwise, new records will be created. If 'Serial In Inventory' and the Consolidate parameter is not active separate receipt line(s) (whinh312) will be created per Serial. Otherwise, Serial(s) will be generated in the Receipt Line Stock Point Details(whinh320) for the same receipt line. Example: Receipt line, 3 pcs (Consolidate = No) Generate Serial: <pre> ----- Receipt Line (whinh312) Stock Point Details (whinh320) ----- 10 SERIAL1 1 pcs SERIAL1 1 pcs 20 SERIAL2 1 pcs SERIAL2 1 pcs 30 SERIAL3 1 pcs SERIAL3 1 pcs </pre> Example: Receipt line, 3 pcs (Consolidate = Yes) Generate Serial:

	<pre> ----- Receipt Line (whinh312) Stock Point Details (whinh320) ----- 10 (empty) 3 pcs SERIAL1 1 pcs SERIAL2 1 pcs SERIAL3 1 pcs ----- </pre>
	Now suppose the item is Lot Controlled and Serialized:
	Example: Receipt line, 3 pcs (Consolidate = No)
	Current values:
	<pre> ----- Receipt Line (whinh312) Stock Point Details (whinh320) ----- 10 LOT1 3 pcs LOT1 3 pcs ----- </pre>
	Generate Serials:
	<pre> ----- Receipt Line (whinh312) Stock Point Details (whinh320) ----- 10 SERIAL1 LOT1 1 pcs SERIAL1 LOT1 1 pcs 20 SERIAL2 LOT1 1 pcs SERIAL2 LOT1 1 pcs 30 SERIAL3 LOT1 1 pcs SERIAL3 LOT1 1 pcs ----- </pre>
	Example: Receipt line, 3 pcs (Consolidate = Yes)
	Current values:
	<pre> ----- Receipt Line (whinh312) Stock Point Details (whinh320) ----- 10 LOT1 3 pcs LOT1 3 pcs ----- </pre>
	Generate Serial:
	<pre> ----- Receipt Line (whinh312) Stock Point Details (whinh320) ----- 10 LOT1 3 pcs SERIAL1 LOT1 1 pcs SERIAL2 LOT1 1 pcs SERIAL3 LOT1 1 pcs ----- </pre>
	Pre: db.retry.point()
	Post: abort.transaction() or commit.transaction()
	Input: iReceipt - Mandatory
	iReceiptLine - Mandatory
	oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
	oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information.
	Return: 0: OK, <> 0: Error

ReceiptLine.GetLanguage

DLL:	whextinhapi
	This function is available from KB2304631 .
Syntax:	long ReceiptLine.GetLanguage(domain whinh.shpm iReceipt,

	domain tcpono iReceiptLine, boolean iConvertToSystemLanguage, ref domain tcclan oLanguage, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function is used to determine the language for printing receipt text and inspection text. Depending on the order origin, the receipt text is entered by default in the company language, Business Partner language or Order Header Language. This function returns the proper language code in which the receipt text (whinh312.txt) and inspection text (whinh312.itxt) will be printed. The same applies to printing the Inspection Text (whinh211.itxt). Pre: N.a. Post: N.a. Input: iReceipt - Receipt number (Mandatory) iReceipt - Receipt line (Mandatory) iConvertToSystemLanguage - Convert to internal language (true/false) Output: oLanguage - The language code for the receipt Return: 0: OK <> 0: Error

ReceiptLine.Reverse

DLL:	whextinhapi This function is available from KB3561538 .
Syntax:	<pre>long ReceiptLine.Reverse(domain whinh.shpm iReceipt, domain tcpono iReceiptLine, domain whhuid iHandlingUnit, domain whloca iAdjustLocation, boolean iSetPackingSlipQuantityToZero, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function will reverse the given receipt line. Be aware that transaction management is handled within this function. Pre: N.a. Post: N.a. Input: iReceipt - Mandatory iReceiptLine - Mandatory iHandlingUnit - Optional, unless handling units are linked to the receipt line. iAdjustLocation - Optional, value is ignored when iHandlingUnit is filled. When warehouse and item of receipt line are location controlled, then this field becomes mandatory when iHandlingUnit is not filled. iSetPackingSlipQuantityToZero - Mandatory Output: oExceptionMessage - The last message if any message is

	found. If more than one message is given, these are present in the oExceptionID. - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

ReceiptLine.Split

DLL:	whextinhapi
Syntax:	<pre> long ReceiptLine.Split(domain whinh.shpm iReceipt, domain tcpono iReceiptLine, ref long oNumberOfReceiptLines, ref domain tcpono oReceiptLineArray(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl : This public interface splits the given receipt line. The given receipt line will be set to zero received quantity and the existing details (like Stock Point Details and/or Handling Units) will be moved to the next receipt line number according to the step size defined in the master data set up. For executing this public interface the same constraints apply as for the corresponding option in the Infor LN Application. If execution is not allowed a specified error message is returned. Pre: db.retry.point must be set oReceiptLineArray must be declared as a based variable, this function will allocate the memory. Post: Commit the transaction in case of success Abort the transaction in case of failure After the oReceiptLineArray is used, free the memory. Input: iReceipt - Mandatory iReceiptLine - Mandatory Output: oNumberOfReceiptLines - Number of receipt lines in array after split. oReceiptLineArray - Array of created receipt lines during splitting (excluding iReceiptLine). oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

ReceiptLine.SplitForLotsBatches

DLL:	whextinhapi																
Syntax:	<pre> long ReceiptLine.SplitForLotsBatches(domain whinh.shpm iReceipt, domain tcpono iReceiptLine, boolean generateLots, domain tcqivl iSplitSize, domain tcpono iSplitNumber, ref long oNumberOfReceiptLines, ref domain tcpono oReceiptLineArray(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>																
Usage:	Expl : This public interface splits the receipt line (or the stock point details) into separate lines for each lot / batch size. Dependent on the 'Consolidate Stock Points in one Receipt Line' parameter, this public interface has a different result. If the parameter value is No this function splits the receipt line and creates separate receipt lines according to the Split Size and Split Number. If the parameter value is Yes, it does not split the receipt line but splits the related Stock Point Details accordingly. Optionally, if the iGenerateLots is True lot codes will be generated for each created receipt line or detail line. The Split Size and Split Number are optional arguments and either one can be calculated based on the receipt quantity and the value of the other argument. If both Split Size and Split Number are empty, the default Size is determined from the Default Lot Size defined in session "Item-Warehousing" (whwmd4600m000) or "Items-Warehousing by Site" (whwmd4604m000). But in the end both variables are needed for executing the split. If it is not possible to determine the Split Size and Number an error message will be returned. For executing this public interface the same constraints apply as for the corresponding option in the Infor LN Application. If execution is not allowed a specified error message is returned. Example: - Receipt Quantity = 30 - Consolidate Stock Points in one Receipt Line = Yes - iGenerateLots = True, iSplitSize = 10, iSplitNumber = 3 <table border="1"> <thead> <tr> <th>Receipt Line (whinh312)</th><th>Stock Point Details (whinh320)</th></tr> </thead> <tbody> <tr> <td>10 (empty) 30 pcs</td><td>LOT1 10 pcs</td></tr> <tr> <td></td><td>LOT2 10 pcs</td></tr> <tr> <td></td><td>LOT3 10 pcs</td></tr> </tbody> </table> Example: - Receipt Quantity = 30 - Consolidate Stock Points in one Receipt Line = No - iGenerateLots = False, iSplitSize = 10, iSplitNumber = 3 <table border="1"> <thead> <tr> <th>Receipt Line (whinh312)</th><th>Stock Point Details (whinh320)</th></tr> </thead> <tbody> <tr> <td>10 (empty) 10 pcs</td><td>(empty) 10 pcs</td></tr> <tr> <td>20 (empty) 10 pcs</td><td>(empty) 10 pcs</td></tr> <tr> <td>30 (empty) 10 pcs</td><td>(empty) 10 pcs</td></tr> </tbody> </table> Pre: db.retry.point must be set oReceiptLineArray must be declared as a based variable, this	Receipt Line (whinh312)	Stock Point Details (whinh320)	10 (empty) 30 pcs	LOT1 10 pcs		LOT2 10 pcs		LOT3 10 pcs	Receipt Line (whinh312)	Stock Point Details (whinh320)	10 (empty) 10 pcs	(empty) 10 pcs	20 (empty) 10 pcs	(empty) 10 pcs	30 (empty) 10 pcs	(empty) 10 pcs
Receipt Line (whinh312)	Stock Point Details (whinh320)																
10 (empty) 30 pcs	LOT1 10 pcs																
	LOT2 10 pcs																
	LOT3 10 pcs																
Receipt Line (whinh312)	Stock Point Details (whinh320)																
10 (empty) 10 pcs	(empty) 10 pcs																
20 (empty) 10 pcs	(empty) 10 pcs																
30 (empty) 10 pcs	(empty) 10 pcs																

	function will allocate the memory. Post: Commit the transaction in case of success Abort the transaction in case of failure After the oReceiptLineArray is used, free the memory. Input: iReceipt - Mandatory iReceiptLine - Mandatory iGenerateLots - - If true and the item is defined as 'Lot in Inventory', lot codes will be generated per receipt line and/or stock point detail split. iSplitSize - The lot/batch size of the split (optional). If empty, Split Size is determined via receipt quantity and Split Number. If both split size and number are empty, the default Size is determined from the Item-Warehousing. iSplitNumber - The number of lots/batches to be split (optional). If empty, Split Number is determined via receipt quantity and Split Size. Output: oNumberOfReceiptLines - Number of receipt lines in array after split. oReceiptLineArray - Array of created receipt lines during splitting (excluding iReceiptLine). oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

ReceiptLine.SplitForSerials

DLL:	whextinhapi
Syntax:	<pre>long ReceiptLine.SplitForSerials(domain whinh.shpm iReceipt, domain tcpono iReceiptLine, boolean iGenerateSerials, ref long oNumberOfReceiptLines, ref domain tcpono oReceiptLineArray(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl : This public interface splits the receipt line (or the stock point details) into separate lines for each serial. Depending on the parameter 'Consolidate Stock Points in one Receipt Line' this public interface has a different result. If the parameter value is No and the item is 'Serialized in Inventory', the receipt line will be split into separate receipt lines per piece. If the parameter value is Yes, the receipt line stock point

details will be split into separate sequences per piece.
 If the option iGenerateSerials is True, the serial numbers will be generated as well, in both situations.
 If the item is not set as 'Serialized in inventory', this option is not allowed.
 For executing this public interface the same constraints apply as for the corresponding option in the Infor LN Application.
 If execution is not allowed a specified error message is returned.

Example:

- Receipt line, 3 pcs, Serial In Inventory
- Consolidate Stock Points in one Receipt Line = No
- iGenerateSerials = true

Split Line for Serials:

Receipt Line (whinh312)		Stock Point Details (whinh320)	
10	SERIAL1 1 pcs	SERIAL1	1 pcs
20	SERIAL2 1 pcs	SERIAL2	1 pcs
30	SERIAL3 1 pcs	SERIAL3	1 pcs

Example:

- Receipt line, 3 pcs, Serial In Inventory
- Consolidate Stock Points in one Receipt Line = No
- iGenerateSerials = false

Split Line for Serials:

Receipt Line (whinh312)		Stock Point Details (whinh320)	
10	" " 1 pcs	" "	1 pcs
20	" " 1 pcs	" "	1 pcs
30	" " 1 pcs	" "	1 pcs

Example:

- Receipt line, 3 pcs, Serial In Inventory
- Consolidate Stock Points in one Receipt Line = Yes
- iGenerateSerials = true

Split Line for Serials:

Receipt Line (whinh312)		Stock Point Details (whinh320)	
10	(empty) 3 pcs	SERIAL1	1 pcs
		SERIAL2	1 pcs
		SERIAL3	1 pcs

Example:

- Receipt line, 3 pcs, Serial In Inventory
- Consolidate Stock Points in one Receipt Line = Yes
- iGenerateSerials = false

Split Line for Serials:

Receipt Line (whinh312)		Stock Point Details (whinh320)	
10	(empty) 3 pcs	""	1 pcs
		""	1 pcs
		""	1 pcs

Pre: db.retry.point must be set
 oReceiptLineArray must be declared as a based variable, this function will allocate the memory.

Post: Commit the transaction in case of success
 Abort the transaction in case of failure
 After the oReceiptLineArray is used, free the memory.

Input: iReceipt - Mandatory

	iReceiptLine - Mandatory iGenerateSerials - If true, (empty) serial numbers will be generated per receipt line and/or stock point detail split. Output: oNumberOfReceiptLines - Number of receipt lines in array after split. oReceiptLineArray - Array of created receipt lines during splitting (excluding iReceiptLine). oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

ReceiptLine.StartAutomaticInboundProcessing

DLL:	whextinhapi
Syntax:	<pre>long ReceiptLine.StartAutomaticInboundProcessing(domain whinh.shpm iReceipt, domain tcpono iReceiptLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface will start the processing of the next automatic activity after the receipt line has been confirmed. This public interface will generate inbound advices when this is the next automatic activity of the related inbound order line. When there is already an advice present which must be putaway, please use one of the following public interfaces: InboundAdvice.StartAutomaticInboundProcessing InboundRun.StartAutomaticInboundProcessing This public interface will execute the following steps: 1) Clear any prior notifications done for the automatic inbound process, within the current process. 2) Notify the receipt line for automatic processing 3) Start the automatic inbound process for the notified Receipt Line The way to implement this public interface is to make sure that the next activity is set to automatic. This can be already set properly based on the order type setup, or can be achieved by using public interface InboundOrderLineActivity.ModifyAutomaticProcessing() for the activity that should be automatically processed. See below code example: <pre>--- db.retry.point()</pre>

```

|* Set the activity to automatic
ret = InboundOrderLineActivity.ModifyAutomaticProcessing(..)

if ret <> 0 then
    abort.transaction()
    return(ret)
endif

commit.transaction()

|* Start the automatic order process
ret.val = ReceiptLine.StartAutomaticInboundProcessing(..)

db.retry.point()

|* Set the activity to be not automatic
ret = InboundOrderLineActivity.ModifyAutomaticProcessing(..)

if ret <> 0 then
    abort.transaction()
    return(ret)
endif

commit.transaction()

return(ret.val)
---
```

This public interface will be able to automatically generate an inbound advice for the given receipt line. This public interface will stop the automatic process when the next activity is not set to automatic.

This function must not be called within a logical transaction as this function will have it's own transaction handling.

Pre: There should be no pending logical transaction before calling this function.

Post: No need to commit or abort the process, that is handled within the function.

Input: iReceipt - Mandatory
iReceiptLine - Mandatory

Output: oExceptionMessage - The last message if the return value is not equal to 0.
If more than one message is given, these are present in the oExceptionID

oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information.

Return: 0: OK, <> 0: Error

Public Interfaces for ReceiptLineStockPointDetail

The following functions are available:

[ReceiptLineStockPointDetails.StartOverview](#)

ReceiptLineStockPointDetails.StartOverview

DLL:	whextinhapi										
	This function is available from KB2304920 .										
Syntax:	<pre> long ReceiptLineStockPointDetails.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain whinh.shpm iReceipt, domain tcpono iReceiptLine, domain tcmcs.long iDistributionSequence, ref domain whinh.shpm oReceipt, ref domain tcpono oReceiptLine, ref domain tcmcs.long oDistributionSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>										
Usage:	Expl This function starts the overview session Receipt Line Stock Point Details (whinh3123m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend A specific query to be used when zooming to this session. Receipt The Receipt to be started Mandatory if iStartMode = MODELESS iReceiptLine The Receipt Line to be started Mandatory if iStartMode = MODELESS iDistributionSequence Optional Output: for iStartMode MODAL: <table border="0"> <tr> <td>oReceipt</td> <td>- selected receipt</td> </tr> <tr> <td>oReceiptLine</td> <td>- selected receipt line</td> </tr> <tr> <td>oDistributionSequence</td> <td>- selected distribution sequence</td> </tr> <tr> <td>oExceptionMessage</td> <td>- The last message if any message is found. If more than one message is given, these are present in the</td> </tr> <tr> <td></td> <td>oExceptionID.</td> </tr> </table>	oReceipt	- selected receipt	oReceiptLine	- selected receipt line	oDistributionSequence	- selected distribution sequence	oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the		oExceptionID.
oReceipt	- selected receipt										
oReceiptLine	- selected receipt line										
oDistributionSequence	- selected distribution sequence										
oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the										
	oExceptionID.										

	<pre> oExceptionID Return: 0 <> 0 </pre>	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information. - Session started - Error.

Public Interfaces for StorageList

The following functions are available:

[StorageList.SplitAdvice](#)

[StorageList.StartAutomaticInboundProcessing](#)

StorageList.SplitAdvice

DLL:	whextinhapi
	This function is available from KB2288171 .
Syntax:	<pre> long StorageList.SplitAdvice(domain tcorno iInboundAdvice, domain tcpono iInboundAdviceLine, ref domain tcpono oNewInboundAdviceLine, ref domain whinh.lino oNewStorageSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will split the given storage list. Pre: db.retry.point must be set. Post: Commit the transaction in case of success. Abort the transaction in case of failure. Input: iInboundAdvice - Mandatory iInboundAdviceLine - Mandatory Output: oNewInboundAdviceLine - Created Inbound Advice Line. oNewStorageSequence - Created Storage Sequence oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

StorageList.StartAutomaticInboundProcessing

DLL:	whextinhapi
Syntax:	<pre> long StorageList.StartAutomaticInboundProcessing(domain whinh.btno iInboundRun, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will start the processing of the next automatic activity for the inbound run, starting with the next activity, putaway storage list. This public interface will putaway all storage lists which are to be putaway automatically for the storage lists related to the given run. This public interface will execute the following steps: 1) Clear any prior notifications done for the automatic inbound process, within the current process. 2) Notify the run for automatic processing of the storage list 3) Start the automatic inbound process for the notified run, starting at the Storage List. The way to implement this public interface is to make sure that the next activity is set to automatic. This can be already set properly based on the order type setup, or can be achieved by using public interface InboundOrderLineActivity.ModifyAutomaticProcessing() for the activity that should be automatically processed. See below code example: <pre> --- db.retry.point() /* Set the activity to automatic ret = InboundOrderLineActivity.ModifyAutomaticProcessing(..) if ret <> 0 then abort.transaction() return(ret) endif commit.transaction() /* Start the automatic order process ret.val = StorageList.StartAutomaticInboundProcessing(..) db.retry.point() /* Set the activity to be not automatic ret = InboundOrderLineActivity.ModifyAutomaticProcessing(..) if ret <> 0 then abort.transaction() return(ret) endif </pre>

	<pre> commit.transaction() return(ret.val) ---</pre> This public interface will be able to automatically putaway the Storage Lists which are to be putaway for the given run. This public interface will stop the automatic process when the next activity is not set to automatic. This function must not be called within a logical transaction as this function will have it's own transaction handling. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iInboundRun - Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Public Interfaces for PickingList

The following functions are available:

[PickingList.SplitAdvice](#)

PickingList.SplitAdvice

DLL:	whextinhapi
	This function is available from KB2288175 .
Syntax:	<pre> long PickingList.SplitAdvice(domain whinh.oorg iOrderOrigin, domain tcorno iOrderNumber, domain tcwset iOrderSet, domain tcpono iOrderLine, domain tcpono iOrderSequence,</pre>

	domain tcpono iAdvice, ref domain tcpono oNewAdvice, ref domain whinh.lino oNewPickingSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This public interface will split the given picking list. Pre: db.retry.point must be set Post: Commit the transaction in case of success. Abort the transaction in case of failure. Input: iOrderOrigin - Mandatory iOrderNumber - Mandatory iOrderSet - Optional iOrderLine - Optional iOrderSequence - Optional iAdvice - Mandatory Output: oNewAdvice - Created Outbound Advice. oNewPickingSequence - Created Picking Sequence. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Public Interfaces for WarehouseInspection

The following functions are available:

[WarehouseInspection.CopyImage](#)

[WarehouseInspection.LinkHandlingUnit](#)

[WarehouseInspection.Print](#)

[WarehouseInspection.Process](#)

WarehouseInspection.CopyImage

DLL:	whextinhapi
	This function is available from KB3540411 .
Syntax:	long WarehouseInspection.CopyImage (

	<pre> domain tcorno iInspection, domain tcpono iInspectionSequenceSource, domain tcpono iInspectionSequenceTarget, domain tcguid iInspectionImageSource, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl: This function copies the image from the source inspection sequence to the target inspection sequence. Pre: db.retry.point() Post: abort/commit transaction Input: iInspection - Mandatory iInspectionSequenceSource - Mandatory iInspectionSequenceTarget - Mandatory iInspectionImageSource - Mandatory Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error </pre>

WarehouseInspection.LinkHandlingUnit

DLL:	whextinhapi
Syntax:	<pre> long WarehouseInspection.LinkHandlingUnit(domain tcorno iInspection, domain tcpono iInspectionSequence, domain whhuid iHandlingUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl: This public interface supports linking of a handling unit in stock to an existing Inventory Inspection. Warehouse Inspection must be of type Inventory Inspection. Handling unit must have status In Stock. The result of this public interface will be that the iHandlingUnit is linked to the Warehouse Inspection. The iHandlingUnit will get status To be Inspected. Stock point details of the handling unit will be added as inspection lines to the Inventory Inspection. Pre: db.retry.point() is set. Post: commit.transaction() / abort.transaction() Input: iInspection - Inspection Number (mandatory) iInspectionSequence - Inspection Sequence (mandatory) iHandlingUnit - Handling Unit (mandatory) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant </pre>

	information. Return: 0: OK, <> 0: Error

WarehouseInspection.Print

DLL:	whextinhapi This function is available from KB3566665 .
Syntax:	<pre> long WarehouseInspection.Print(domain tcorno iInspection, domain tcpono iInspectionSequence, domain tcmcs.str15 iDevice, long iProcessingOptionSet, ref boolean oDataProcessed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will print the Warehouse Inspections for the iInspection and iInspectionSequence using the defaults or options as provided in the iProcessingOptionSet. Opening/Closing of the report is handled by this function. Pre: N.a. Post: N.a. Input: iInspection Optional iInspectionSequence Optional, ignored if iInspection is empty. iDevice Mandatory iProcessingOptionSet Optional, if 0, the default options are applied. A Processing Option Set can be created via a call to function ProcessingOptionSet.Create(). In case iInspection is filled then the following options of the iProcessingOptionSet will be ignored: - selection range fields (From/To) - InspectionArray The warehouse inspections will then be printed for the given iInspection (and specific iInspectionSequence, if filled). In case option InspectionArray is set then the selection range fields (From/To) of the iProcessingOptionSet will be ignored. The warehouse inspections will then be printed for the inspections in the array. Processing Options have a direct relationship with the form fields on session Print Warehouse Inspections (whinh3422m000) and are not explained in further detail here. Please refer to the session help for additional information. Processing Options that are set while a required Implemented Software Component is not available are ignored. Explanation about setting of defaults: Minimum Value: Minimum value of domain is taken as default value. Maximum Value: Maximum value of domain is taken as default value.

NAME	TYPE	DEFAULT
InspectionFrom	domain tcorno	Minimum Value
InspectionTo	domain tcorno	Maximum Value
InspectionSequenceFrom	domain tcpono	Minimum Value
InspectionSequenceTo	domain tcpono	Maximum Value
OrderOriginFrom	domain whinh.oorg	Minimum Value
OrderOriginTo	domain whinh.oorg	Maximum Value
OrderNumberFrom	domain tcorno	Minimum Value
OrderNumberTo	domain tcorno	Maximum Value
OrderSetFrom	domain tcwset	Minimum Value
OrderSetTo	domain tcwset	Maximum Value
OrderLineFrom	domain tcpono	Minimum Value
OrderLineTo	domain tcpono	Maximum Value
WarehouseFrom	domain tccwar	Minimum Value
WarehouseTo	domain tccwar	Maximum Value
LocationFrom	domain whloca	Minimum Value
LocationTo	domain whloca	Maximum Value
InspectionDateFrom	domain tcdate	Minimum Value
InspectionDateTo	domain tcdate	Maximum Value
ItemFrom	domain tcitem	Minimum Value
ItemTo	domain tcitem	Maximum Value
InspectionStatus	domain whinh.prap	whinh.prap.tobe
InspectionType	domain whinh.prat	whinh.prat.both
PrintInspectionLines	domain tcyesno	tcyesno.no
PrintOwnership	domain tcyesno	tcyesno.no
PrintOrderLineTotals	domain tcyesno	tcyesno.no
PrintHandlingUnits	domain tcyesno	tcyesno.no
PrintInspectionText	domain tcyesno	tcyesno.yes
PrintInspectionInstructionText	domain tcyesno	tcyesno.no
PrintInspectionResultsText	domain tcyesno	tcyesno.no
InspectionArray	domain ttjson	0
ReportNumber	domain tcmcs.long	1
ReportName	domain tcmcs.str16	Empty String

Possible values of ReportNumber are:
1 - Warehouse Inspections

JSON Object InspectionArray has the following structure:

```
"InspectionArray": [
  {
    "Inspection": "INS000006"
    "InspectionSequence": 1,
  },
  {
    "Inspection": "INS000003"
    "InspectionSequence": 2,
  }
]
```

This structure can be created with the following code:

```
InspectionArray = Json.newArray()
Inspection = Json.newObject()
Json.setString(Inspection, "Inspection", "INS000006")
Json.setNumber(Inspection, "InspectionSequence", 1)
Json.add(InspectionArray, Inspection)
Inspection = Json.newObject()
Json.setString(Inspection, "Inspection", "INS000003")
Json.setNumber(Inspection, "InspectionSequence", 2)
Json.add(InspectionArray, Inspection)
```

ReportName only needs to be filled for customized reports,

	otherwise the standard report is used based on the ReportNumber. ReportName must start with an "r", e.g. "rwhinh342211001" Output: oDataProcessed - true: Data Printed. false: Nothing Printed. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

WarehouseInspection.Process

DLL:	whextinhapi
Syntax:	<pre> long WarehouseInspection.Process(domain tcorno iInspection, domain tcpono iInspectionSequence, ref domain tcpono oNewInspectionSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function processes the warehouse inspection having status In Process. This implies either approved or rejected quantities have been specified which still need to be processed. If no handling units are present, all inspection lines linked to the inspection will be processed. If handling units are present, the inspection lines related to the handling unit structure are processed. Pre: db.retry.point() is set. Post: commit.transaction() / abort.transaction() Input: iInspection - Inspection Number (mandatory) iInspectionSequence - Inspection Sequence (mandatory) Output: oNewInspectionSequence - New sequence for remainder after processing a partial inspection oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Public Interfaces for WarehouseInspectionLine

The following functions are available:

[WarehouseInspectionLine.Inspect](#)

[WarehouseInspectionLine.StartDetail](#)

[WarehouseInspectionLine.StartOverview](#)

WarehouseInspectionLine.Inspect

DLL:	whextinhapi																
Syntax:	<pre> long WarehouseInspectionLine.Inspect (domain tcorno iInspection, domain tcpono iInspectionSequence, long iInspectionLine, domain tcqstl iApprovedQuantity, domain tccuni iApprovalUnit, domain tcqivl iDestroyedQuantity, domain tccdis iDestroyReason, domain tcqstl iRejectedQuantity, domain tccuni iRejectUnit, domain tccdis iRejectReason, domain tcqstl iScrappedQuantity, domain tccdis iScrapReason, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>																
Usage:	Expl: This function updates the inspection results of a specific Warehouse Inspection Line which is not linked to a handling unit. The Warehouse Inspection should have status Open or In Process and not reside in a WMS-controlled warehouse. This function does not process the inspection results. For processing, use WarehouseInspection.Process Pre: db.retry.point() is set. Post: commit.transaction() / abort.transaction() Input: <table border="0"> <tr> <td>iInspection</td><td>- Inspection Number (mandatory)</td></tr> <tr> <td>iInspectionSequence</td><td>- Inspection Sequence (mandatory)</td></tr> <tr> <td>iInspectionLine</td><td>- Inspection Line (mandatory)</td></tr> <tr> <td>iApprovedQuantity</td><td>- Approved quantity</td></tr> <tr> <td>iApprovalUnit</td><td>- Approval Unit (mandatory if approved quantity is greater than zero). If some quantity is destroyed, the approval unit must be in inventory unit.</td></tr> <tr> <td>iDestroyedQuantity</td><td>- Destroyed Quantity. Destroyed goods are part of the Approved Quantity and is not allowed for Outbound Inspections.</td></tr> <tr> <td>iDestroyReason</td><td>- Destroy Reason (mandatory if destroyed quantity is greater than zero). Reason must be of type 'Destroyed during inspection'.</td></tr> <tr> <td>iRejectedQuantity</td><td>- Rejected Quantity</td></tr> </table>	iInspection	- Inspection Number (mandatory)	iInspectionSequence	- Inspection Sequence (mandatory)	iInspectionLine	- Inspection Line (mandatory)	iApprovedQuantity	- Approved quantity	iApprovalUnit	- Approval Unit (mandatory if approved quantity is greater than zero). If some quantity is destroyed, the approval unit must be in inventory unit.	iDestroyedQuantity	- Destroyed Quantity. Destroyed goods are part of the Approved Quantity and is not allowed for Outbound Inspections.	iDestroyReason	- Destroy Reason (mandatory if destroyed quantity is greater than zero). Reason must be of type 'Destroyed during inspection'.	iRejectedQuantity	- Rejected Quantity
iInspection	- Inspection Number (mandatory)																
iInspectionSequence	- Inspection Sequence (mandatory)																
iInspectionLine	- Inspection Line (mandatory)																
iApprovedQuantity	- Approved quantity																
iApprovalUnit	- Approval Unit (mandatory if approved quantity is greater than zero). If some quantity is destroyed, the approval unit must be in inventory unit.																
iDestroyedQuantity	- Destroyed Quantity. Destroyed goods are part of the Approved Quantity and is not allowed for Outbound Inspections.																
iDestroyReason	- Destroy Reason (mandatory if destroyed quantity is greater than zero). Reason must be of type 'Destroyed during inspection'.																
iRejectedQuantity	- Rejected Quantity																

	iRejectUnit - Rejection Unit (mandatory if rejected quantity is greater than zero). iRejectReason - Reject Reason (mandatory if rejected quantity is greater than zero). Reason must be of type 'Rejection of Goods'. iScrappedQuantity - Scrapped Quantity (in Rejection Unit). Scrapped quantity is part of Rejected quantity. iScrapReason - Scrap Reason (mandatory if scrapped quantity is greater than zero). Reason must be of type 'Disposition'. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

WarehouseInspectionLine.StartDetail

DLL:	whextinhapi
Syntax:	<pre>long WarehouseInspectionLine.StartDetail(long iStartMode, domain tcorno iInspection, domain tcpono iInspectionSequence, long iInspectionLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface starts session Warehouse Inspection Lines (whinh2131m000) in Detail mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: iInspection - The Inspection Number to be started iInspectionSequence - The Inspection Sequence to be started iInspectionLine - The Inspection Line to be started. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the

	oExceptionID Return: 0 <> 0	oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Session started - Error

WarehouseInspectionLine.StartOverview

DLL:	whextinhapi
Syntax:	<pre> long WarehouseInspectionLine.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iInspection, domain tcpono iInspectionSequence, long iInspectionLine, domain whinh.shpm iReceipt, domain tcpono iReceiptLine, domain whinh.oorg iOrderOrigin, domain tcorno iOrder, domain tcwset iOrderSet, domain tcitem iItem, domain tccwar iWarehouse, domain whloca iLocation, ref domain tcorno oInspection, ref domain tcpono oInspectionSequence, ref long oInspectionLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface starts session Warehouse Inspection Lines (whinh2131m000) in Overview mode. Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "byInspectionLine" Data is displayed by Inspection Line session will be started on index 1 view fields: Inspection, Inspection Sequence "byReceiptLine" Data is displayed by Receipt Line session will be started on index 2

	<pre> view fields: Receipt, Receipt Line "byOrder" Data is displayed by Warehousing Order session will be started on index 3 view fields: Order Origin, Order, Order Set "byItem" Data is displayed by Item session will be started on index 4 view fields: Item "byWarehouseLocation" Data is displayed by Warehouse Location session will be started on index 5 view fields: Warehouse, Location iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session iInspection The Inspection Number to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 1 (or iStartFilter = "byInspectionLine") iInspectionSequence The Inspection Sequence to be started Mandatory if iStartMode = MODELESS and iSessionIndex = 1 (or iStartFilter = "byInspectionLine") iInspectionLine The Inspection Line to be started. Not mandatory. iReceipt Not Mandatory iReceiptLine Not Mandatory iOrderOrigin Mandatory if iStartMode = MODELESS and iSessionIndex = 3 (or iStartFilter = "byOrder") iOrder Not Mandatory iOrderSet Not Mandatory iItem Not Mandatory iWarehouse Not Mandatory iLocation Not Mandatory Output: oInspection - Inspection Number of selected record oInspectionSequence - Inspection Sequence of selected record oInspectionLine - Inspection Line of selected record oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error </pre>

Public Interfaces for InventoryInspection

The following functions are available:

[InventoryInspection.Generate](#)

InventoryInspection.Generate

DLL:	whextinhapi		
	This function is available from KB2296967 .		
Syntax:	<pre> long InventoryInspection.Generate(domain tcsite iSiteFrom, domain tcsite iSiteTo, domain tccwar iWarehouseFrom, domain tccwar iWarehouseTo, domain whloca iLocationFrom, domain whloca iLocationTo, domain tccom.bpid iLocationOwnerFrom, domain tccom.bpid iLocationOwnerTo, domain tccom.bpid iBuyFromBusinessPartnerFrom, domain tccom.bpid iBuyFromBusinessPartnerTo, domain tcdate iInventoryDateFrom, domain tcdate iInventoryDateTo, domain whwmd.zone iZoneFrom, domain whwmd.zone iZoneTo, domain whwmd.strt iRowFrom, domain whwmd.strt iRowTo, domain whwmd.coln iLevelFrom, domain whwmd.coln iLevelTo, domain whwmd.rack iBinFrom, domain whwmd.rack iBinTo, domain tccitg iItemGroupFrom, domain tccitg iItemGroupTo, domain tcclot iLotFrom, domain tcclot iLotTo, domain tcibd.sern iSerialNumberFrom, domain tcibd.sern iSerialNumberTo, domain whhuid iHandlingUnitFrom, domain whhuid iHandlingUnitTo, domain tcdate iLastInventoryInspectionDateFrom, domain tcdate iLastInventoryInspectionDateTo, domain tcitem iItemFrom, domain tcitem iItemTo, boolean iInspectNonHandlingUnitInventory, boolean iInspectHandlingUnitInventory, </pre>		

	<pre> domain whinh.cchu iHandlingUnitLevel, boolean boolean iForceInventoryInspection, iExcludeLocationAllocatedItems, domain tccltmo iLeadTime, domain tcto iLeadTimeUnit, domain tcseri iSeries, ref long oNumberOfGeneratedInspections, ref domain tcorno oInspectionArray() fixed, ref domain tcpono oInspectionSequenceArray(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl: This function generates inventory inspection lines based on Stock Point Inventory data and based on Handling Units. Be aware that transaction management is handled within this function. Pre: N.A. Post: N.A. Input: iSiteFrom - Optional iSiteTo - Mandatory iWarehouseFrom - Optional iWarehouseTo - Mandatory iLocationFrom - Optional iLocationTo - Mandatory iLocationOwnerFrom - Optional iLocationOwnerTo - Mandatory iBuyFromBusinessPartnerFrom - Optional iBuyFromBusinessPartnerTo - Mandatory iInventoryDateFrom - Optional iInventoryDateTo - Mandatory iZoneFrom - Optional iZoneTo - Mandatory iRowFrom - Optional iRowTo - Mandatory iLevelFrom - Optional iLevelTo - Mandatory iBinFrom - Optional iBinTo - Mandatory iItemGroupFrom - Optional iItemGroupTo - Mandatory iLotFrom - Optional iLotTo - Mandatory iSerialNumberFrom - Optional iSerialNumberTo - Mandatory iHandlingUnitFrom - Optional iHandlingUnitTo - Mandatory iLastInventoryInspectionDateFrom - Optional iLastInventoryInspectionDateTo - Mandatory iItemFrom - Optional iItemTo - Mandatory iInspectNonHandlingUnitInventory - When this is set to True, inventory that is not stored in handling units is inspected. iInspectHandlingUnitInventory - When this is set to True, inventory that is stored in handling units is inspected iHandlingUnitLevel - Mandatory Allowed Values: - Inspect Handling Unit Top Levels (whinh.cchu.top.level) - Inspect Handling Unit Bottom Levels (whinh.cchu.bottom.level) iForceInventoryInspection - When this is set to True, all stock points found in the selection are inspected. </pre>

	When set to False, the stock points due for inspection according to the planned inspection date and frequency of inventory inspection are inspected. iExcludeLocationAllocatedItems - When this is set to True, ERP LN does not generate inventory inspections for stock points with location allocated inventory. iLeadTime - Optional iLeadTimeUnit - Mandatory Allowed Values: - Hours (tctopec.hours) - Days (tctopec.days) iSeries - Optional Output: oNumberOfGeneratedInspections - Number of generated inventory inspections oInspectionArray - Array with generated inventory inspection numbers. This array must be declared as a based variable; this function will allocate the memory oInspectionSequenceArray - Array with generated inventory inventory sequences. This array must be declared as a based variable; this function will allocate the memory oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Success <> 0 - Error

Public Interfaces for HandlingUnitStockPointDetail

The following functions are available:

[HandlingUnitStockPointDetail.Inspect](#)

[HandlingUnitStockPointDetail.SplitForSerial](#)

HandlingUnitStockPointDetail.Inspect

DLL:	whextwmdapi
Syntax:	<pre>long HandlingUnitStockPointDetail.Inspect (domain whhuid iHandlingUnit, domain tcmcs.long iDistributionSequence,</pre>

	domain tccuni iStorageUnit, domain tcqstl iApprovedQuantity, domain tcqivl iDestroyedQuantity, domain tccdis iDestroyReason, domain tcqstl iRejectedQuantity, domain tccdis iRejectReason, domain tcqstl iScrappedQuantity, domain tccdis iScrapReason, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function updates the inspection results of a specific Handling Unit Stock Point Detail - Distribution Sequence. The related warehouse inspection of the Handling Unit should have status Open or In Process and not reside in a WMS-controlled warehouse. This function must only be called for bottom handling units (not allowed for handling unit having child handling unit(s)). This function can also not be used for warehouse inspections which are blocked by a QM Order Inspection. This function does not process the inspection results. For processing, use WarehouseInspection.Process Pre: db.retry.point() is set. Post: commit.transaction() / abort.transaction() Input: iHandlingUnit - Handling Unit (mandatory) iDistributionSequence - Handling Unit Stock Point Detail Distribution Sequence (mandatory) iStorageUnit - Storage Unit (mandatory) iApprovedQuantity - Approved quantity (in storage unit) iDestroyedQuantity - Destroyed Quantity (in inventory unit) Destroyed goods are part of the Approved Quantity and must be expressed in inventory unit and is not allowed for Outbound Inspections. iDestroyReason - Destroy Reason (mandatory if destroyed quantity is greater than zero). Reason must be of type 'Destroyed during inspection'. iRejectedQuantity - Rejected Quantity (in storage unit) iRejectReason - Reject Reason (mandatory if rejected quantity is greater than zero). Reason must be of type 'Rejection of Goods'. iScrappedQuantity - Scrapped Quantity (in storage unit) Scrapped quantity is part of Rejected quantity. iScrapReason - Scrap Reason (mandatory if scrapped quantity is greater than zero). Reason must be of type 'Disposition'. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

HandlingUnitStockPointDetail.SplitForSerial

DLL:	whextwmdapi
	This function is available from KB3541323 .
Syntax:	<pre> long HandlingUnitStockPointDetail.SplitForSerial(domain whhuid iHandlingUnit, domain tcmcs.long iDistributionSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface splits handling unit stock point detail with serialized item in multiple stock point details. Split is allowed when: - Serial is empty and Stock Point Detail quantity is greater than 1.0. - Handling Unit Status is equal to: - ToInspect and inspection is not yet started; - Staged and expected not shipped quantity is 0.0; - Allocated, Partial Allocated, Partial Released or Released and handling unit is linked to the one order line only. - All other statuses except Shipment Frozen, Partial Frozen, Confirming, Shipped, Closed and Inactive. Handling unit with serial in inventory item must be generated according to the package definition template which allows multiple stock points at the bottom level. In case multiple stock points are not allowed a separate handling unit must be generated for each serial number. Pre: db.retry.point() is set. Post: commit.transaction() / abort.transaction() Input: iHandlingUnit - Handling Unit (mandatory) iDistributionSequence - Handling Unit Stock Point Detail Distribution Sequence (mandatory) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Public Interfaces for NegativeInventory

The following functions are available:

[NegativeInventory.Consume](#)

NegativeInventory.Consume

DLL:	whextwmdapi
Syntax:	<pre> long NegativeInventory.Consume(domain tccwar iWarehouse, domain whloca ilocation, domain tcitem iItem, domain tcclot iLot, domain tcinvdate iInventoryDate, domain tcqstl iQuantity, domain tccuni iStorageUnit, domain whwmd.pkdf iPackageDefinition, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This public interface consumes negative inventory. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iWarehouse - Warehouse (Mandatory) iLocation - Location (Mandatory, if warehouse and item are location controlled) iItem - Item (Mandatory) iLot - Lot iInventoryDate - Inventory Date iQuantity - Quantity in Storage Unit (Mandatory) Must have a positive value. iStorageUnit - Storage Unit (Mandatory) iPackageDefinition - Package definition Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Success <> 0 - Error

Public Interfaces for Weight

The following functions are available:

[Weight.Convert](#)

Weight.Convert

DLL:	whextwmdapi
Syntax:	<pre> long Weight.Convert(domain tcwght iWeight, domain tccuni iFromWeightUnit, domain tccuni iToWeightUnit, ref domain tcwght oConvertedWeight, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface converts a weight to a different weight unit. Pre: N.A. Post: N.A. Input: iWeight - Weight to be convert, expressed in iFromWeightUnit iFromWeightUnit - Initial weight unit to convert from (Mandatory) iToWeightUnit - Desired weight unit to convert to (Mandatory) Output: oConvertedWeight - The converted weight, expressed in iToWeightUnit. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: N.A.

Public Interfaces for UnexpectedReceipt

The following functions are available:

[UnexpectedReceipt.Create](#)

UnexpectedReceipt.Create

DLL:	whextinhapi
Syntax:	<pre> long UnexpectedReceipt.Create(</pre>

	<pre> domain tccwar iWarehouse, domain tccom.bpid iShipFromBusinessPartner, domain whinh.shpm iShipmentNotice, domain tcpono iShipmentNoticeSequence, domain whinh.load iFreightLoad, domain whinh.shpm iFreightShipment, domain tccfrw iCarrier, domain tcpksp iPackingSlip mb, domain tcyesno iFinalReceipt, domain tcyesno iConfirmReceipt, domain tcmcs.long iNumberOfItems, ref domain tcitem iItemArray() fixed, ref domain tcclot iLotArray() fixed, ref domain tcibd.sern iSerialArray() fixed, ref domain tcqstl iReceivedQuantityArray(), ref domain tccuni iReceivedUnitArray() fixed, ref domain tcdate iActualReceiptDateArray(), ref domain tcmcs.long iNumberOfStockPointDetailsArray(), ref domain tcclot iStockPointLotArray(,) fixed, ref domain tcibd.sern iStockPointSerialArray(,) fixed, ref domain tcinv.date iStockPointInvDateArray(,), ref domain tcuef.effn iStockPointEffUnitArray(,), ref domain tcqstl iStockPointQuantityArray(,), ref domain tccuni iStockPointUnitArray(,) fixed, ref domain tcpono iStockPointOwnSequenceArray(,), ref domain whinh.shpm oReceipt, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>																		
Usage:	Expl: This public interface supports creation and confirmation of the unexpected receipt lines (receipt lines which are not linked to an inbound order line). The following steps will be executed by this Public Interface: 1. Create unexpected receipt lines for each item in array <code>iItemArray()</code>. 2. Set final receipt for all created unexpected receipt lines if parameter <code>iFinalReceipt = tcyesno.yes</code>. 3. If <code>iConfirmReceipt = tcyesno.yes</code> then - Generate purchase order. To facilitate this step the option Generate Order for unexpected warehousing receipt in the Items - Purchase Business Partner (session <code>tdipu0110m000</code>) must be enabled for items present in <code>iItemArray()</code> in combination with a <code>iShipFromBusinessPartner</code>. - Create new open receipt lines to replace unexpected receipt lines which are created in step 1. - Confirm the created open receipt lines. Note: if for some reason like insufficient stock point data found confirm receipt fails the unexpected receipt lines will be present and can be confirmed manually later on. Be aware that transaction management is handled within this function. Pre: N.a. Post: N.a. Input: <table> <tr> <td><code>iWarehouse</code></td><td>- Warehouse; Mandatory</td></tr> <tr> <td><code>iShipFromBusinessPartner</code></td><td>- Ship From Business Partner; Mandatory</td></tr> <tr> <td><code>iShipmentNotice</code></td><td>- Shipment Notice; Optional</td></tr> <tr> <td><code>iShipmentNoticeSequence</code></td><td>- Shipment Notice Sequence; Optional</td></tr> <tr> <td><code>iFreightLoad</code></td><td>- Freight Load; Optional</td></tr> <tr> <td><code>iFreightShipment</code></td><td>- Freight Shipment; Optional</td></tr> <tr> <td><code>iCarrier</code></td><td>- Carrier; Optional</td></tr> <tr> <td><code>iPackingSlip</code></td><td>- Packing Slip; Optional</td></tr> <tr> <td><code>iFinalReceipt</code></td><td>- Final Receipt; Mandatory</td></tr> </table>	<code>iWarehouse</code>	- Warehouse; Mandatory	<code>iShipFromBusinessPartner</code>	- Ship From Business Partner; Mandatory	<code>iShipmentNotice</code>	- Shipment Notice; Optional	<code>iShipmentNoticeSequence</code>	- Shipment Notice Sequence; Optional	<code>iFreightLoad</code>	- Freight Load; Optional	<code>iFreightShipment</code>	- Freight Shipment; Optional	<code>iCarrier</code>	- Carrier; Optional	<code>iPackingSlip</code>	- Packing Slip; Optional	<code>iFinalReceipt</code>	- Final Receipt; Mandatory
<code>iWarehouse</code>	- Warehouse; Mandatory																		
<code>iShipFromBusinessPartner</code>	- Ship From Business Partner; Mandatory																		
<code>iShipmentNotice</code>	- Shipment Notice; Optional																		
<code>iShipmentNoticeSequence</code>	- Shipment Notice Sequence; Optional																		
<code>iFreightLoad</code>	- Freight Load; Optional																		
<code>iFreightShipment</code>	- Freight Shipment; Optional																		
<code>iCarrier</code>	- Carrier; Optional																		
<code>iPackingSlip</code>	- Packing Slip; Optional																		
<code>iFinalReceipt</code>	- Final Receipt; Mandatory																		

	iConfirmReceipt - Confirm Receipt; Mandatory. iNumberOfItems - Number of Items; Mandatory iItemArray - Array of Items; Mandatory iLotArray - Array of Lots; Optional iSerialArray - Array of Serials; Optional iReceivedQuantityArray - Array of quantities; Mandatory iReceivedUnitArray - Array of units; Mandatory iActualReceiptDateArray - Actual Receipt Date; Optional iNumberOfStockPointDetailsArray - Number stock point details per receipt line; Optional iStockPointLotArray - Stock point detail Lot; Optional iStockPointSerialArray - Stock point detail Serial; Optional iStockPointInvDateArray - Stock point detail Inventory Date; Optional iStockPointEffUnitArray - Stock point detail Effectivity Unit; Optional iStockPointQuantityArray - Stock point detail Quantity; Optional iStockPointUnitArray - Stock point detail Storage Unit; Optional iStockPointOwnSequenceArray - Stock point detail Ownership Sequence; Optional Output: oReceipt - Created Receipt Number oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Public Interfaces for WarehouseOrderLine

The following functions are available:

[WarehouseOrderLine.Activate](#)

[WarehouseOrderLine.ActivityIsApplicable](#)

[WarehouseOrderLine.ActivityIsMandatory](#)

[WarehouseOrderLine.UpdatePlannedDates](#)

WarehouseOrderLine.Activate

DLL:	whextinhapi
Syntax:	<pre> long WarehouseOrderLine.Activate(domain whinh.oorg iWarehouseOrderOrigin, domain tcorno iWarehouseOrder, domain tcpono iWarehouseOrderLine, domain tcpono iWarehouseOrderSequence, domain whinh.ittp iTransactionType, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface will activate a given warehouse order line. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iWarehouseOrderOrigin The Order origin to be activated. This is mandatory to fill. iWarehouseOrder Order number to be activated. This is mandatory to fill. iWarehouseOrderLine Order line number to be activated. Not mandatory to fill. A warehouse order line can have line number 0. iWarehouseOrderSequence Order line sequence to be activated. Not mandatory to fill. A warehouse order sequence can have sequence 0. iTransactionType Indicator for what type of order line has to be activated. This is mandatory to fill. Possible values are: Receipt (1) - An inbound order line will be activated. Issue (2) - An outbound order line will be activated. Transfer (3) - An outbound order line will be activated. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Warehouse Order Line has been activated successfully <> 0 - Error.

WarehouseOrderLine.ActivityIsApplicable

DLL:	whextinhapi
	This function is available from KB2304612 .
Syntax:	<pre> long WarehouseOrderLine.ActivityIsApplicable(domain whinh.oorg iOrderOrigin, domain tcorno iOrder, domain tcpono iOrderLine, domain tcpono iOrderSequence, domain whinh.prty iProcedureType, domain tcmcs.str20m iActivity mb, </pre>

	ref boolean oActivityApplicable, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This Public Interface checks if an activity is applicable for a warehouse order line. Pre: N.a. Post: N.a. Input: iOrderOrigin - Order Origin (Mandatory) iOrder - Order (Mandatory) iOrderLine - Order Line iOrderSequence - Order Sequence iProcedureType - Procedure Type (Mandatory) Allowed Values: Inbound Procedure (whinh.prtty.receipt.line) Inspection Procedure (whinh.prtty.receipt.isnpect) Outbound Procedure (whinh.prtty.issue.advice) Shipment Procedure (whinh.prtty.issue.shipment) iActivity - Activity (Mandatory) Allowed Values: The values are depending on the procedure type. Receipt Procedure: "whinh3412m100" - Print Goods Received Note "whinh3512m000" - Warehouse Receipts "whinh3201m000" - Generate Inbound Advice "whinh3415m000" - Generate Storage List "whinh3525m100" - Confirm Storage List "whinh3203m000" - Putaway Inbound Advice Inspection Procedure: "whinh3201m000" - Generate Inbound Advice "whinh3415m000" - Generate Storage List "whinh3525m100" - Confirm Storage List "whinh3203m000" - Putaway Inbound Advice Outbound Procedure: "whinh4201m000" - Generate Outbound Advice "whinh4202m000" - Release Outbound Advice "whinh4415m000" - Print Picking List "whinh4525m100" - Confirm Picking List Shipment Procedure: "whinh4275m001" - Freeze Shipment "whinh4275m000" - Confirm Shipment "whinh4470m000" - Print Bills of Lading "whinh4475m000" - Print Packing Slip "whinh4476m000" - Print Packing List "whinh4477m000" - Print Delivery Notes "whinh4478m000" - Print Shipping Manifest "whinh4279m000" - Print Pro Forma Invoice Output: oActivityApplicable - Activity Applicable Indicator oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - No Error <> 0 - Error

WarehouseOrderLine.ActivityIsMandatory

DLL:	whextinhapi This function is available from KB2304607 .
Syntax:	<pre> long WarehouseOrderLine.ActivityIsMandatory(domain whinh.oorg iOrderOrigin, domain tcorno iOrder, domain tcpono iOrderLine, domain tcpono iOrderSequence, domain whinh.prtty iProcedureType, domain tcmcs.str20m iActivity mb, ref boolean oActivityMandatory, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface checks if an activity is mandatory for a warehouse order line. Pre: N.a. Post: N.a. Input: iOrderOrigin - Order Origin (Mandatory) iOrder - Order (Mandatory) iOrderLine - Order Line iOrderSequence - Order Sequence iProcedureType - Procedure Type (Mandatory) Allowed Values: - Inbound Procedure (whinh.prtty.receipt.line) - Inspection Procedure (whinh.prtty.receipt.isnpect) - Outbound Procedure (whinh.prtty.issue.advice) - Shipment Procedure (whinh.prtty.issue.shipment) iActivity - Activity (Mandatory) Allowed Values: The values are depending on the procedure type. Receipt Procedure: "whinh3412m100" - Print Goods Received Note "whinh3512m000" - Warehouse Receipts "whinh3201m000" - Generate Inbound Advice "whinh3415m000" - Generate Storage List "whinh3525m100" - Confirm Storage List "whinh3203m000" - Putaway Inbound Advice Inspection Procedure: "whinh3201m000" - Generate Inbound Advice "whinh3415m000" - Generate Storage List "whinh3525m100" - Confirm Storage List "whinh3203m000" - Putaway Inbound Advice Outbound Procedure: "whinh4201m000" - Generate Outbound Advice "whinh4202m000" - Release Outbound Advice "whinh4415m000" - Print Picking List "whinh4525m100" - Confirm Picking List Shipment Procedure: "whinh4275m001" - Freeze Shipment "whinh4275m000" - Confirm Shipment "whinh4470m000" - Print Bills of Lading "whinh4475m000" - Print Packing Slip "whinh4476m000" - Print Packing List "whinh4477m000" - Print Delivery Notes "whinh4478m000" - Print Shipping Manifest "whinh4279m000" - Print Pro Forma Invoice Output: oActivityMandatory - Activity Mandatory Indicator

	oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - No Error <> 0 - Error

WarehouseOrderLine.UpdatePlannedDates

DLL:	whextinhapi This function is available from KB2308224.
Syntax:	<pre>long WarehouseOrderLine.UpdatePlannedDates (domain whinh.oorg iOrderOrigin, domain tcorno iOrderNumber, domain tcpono iOrderLine, domain tcpono iOrderSequence, domain tcdte iPlannedDeliveryDate, domain tcdte iPlannedReceiptDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function updates the planned dates of a warehouse order line Pre: db.retry.point must be set Post: Commit the transaction in case of success Abort the transaction in case of failure Input: iOrderOrigin - Order Origin; Mandatory iOrderNumber - Order Number; Mandatory iOrderLine - Order Line iOrderSequence - Order Sequence iPlannedDeliveryDate - Planned Delivery Date; Mandatory iPlannedReceiptDate - Planned Receipt Date; Mandatory Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Planned Dates updated succesfully DALHOOKERROR - An error occurred when updating Planned Dates

Public Interfaces for Kanban

The following functions are available:

[Kanban.GenerateOrders](#)[Kanban.StartGenerate](#)

Kanban.GenerateOrders

DLL:	whextinhapi																		
	This function is available from KB3647754 .																		
Syntax:	<pre> long Kanban.GenerateOrders (domain tckbid iKanbanSignal, domain whloca iLocationTo, domain tcqivl iQuantity, domain tcncmp iCompanyFrom, domain tcsupt iSupplyTypeFrom, domain tcsupc iSupplyCodeFrom, domain tccom.bpid iShipFromBusinessPartner, domain tcseri iWarehouseOrderSeries, domain tccotp iWarehouseOrderType, domain tcseri iProductionOrderSeries, domain tcseri iPurchaseOrderSeries, domain tccotp iPurchaseOrderType, domain tcseri iSalesOrderSeries, domain tccotp iSalesOrderType, domain tcseri iWarehouseAdviceSeries, domain tcseri iProductionAdviceSeries, domain tcseri iPurchaseAdviceSeries, domain tcyesno iActivateWarehouseTransfer, domain tcyesno iProcessWarehouseTransfer, domain tcyesno iReleaseProductionOrder, domain tcyesno iCombineOrderAdvice, domain tcyesno iConfirmOrderAdvice, ref domain tcorno oAdvice, ref domain whinh.oorg oOrderOrigin, ref domain tcorno oOrderNumber, ref domain tcpono oOrderLine, ref domain tcpono oOrderSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>																		
Usage:	Expl: This function creates a warehouse order, a production order, a purchase order, a warehouse advice, a production advice or a purchase advice for a specified kanban signal. Be aware that transaction management is handled within this function. Pre: N.a. Post: N.a. Input: <table> <tr><td>iKanbanSignal</td><td>Mandatory</td></tr> <tr><td>iLocationTo</td><td>Optional</td></tr> <tr><td>iQuantity</td><td>Optional</td></tr> <tr><td>iCompanyFrom</td><td>Optional</td></tr> <tr><td>iSupplyTypeFrom</td><td>Optional</td></tr> <tr><td>iSupplyCodeFrom</td><td>Optional</td></tr> <tr><td>iShipFromBusinessPartner</td><td>Optional</td></tr> <tr><td>iWarehouseOrderSeries</td><td>Optional</td></tr> <tr><td>iWarehouseOrderType</td><td>Optional</td></tr> </table>	iKanbanSignal	Mandatory	iLocationTo	Optional	iQuantity	Optional	iCompanyFrom	Optional	iSupplyTypeFrom	Optional	iSupplyCodeFrom	Optional	iShipFromBusinessPartner	Optional	iWarehouseOrderSeries	Optional	iWarehouseOrderType	Optional
iKanbanSignal	Mandatory																		
iLocationTo	Optional																		
iQuantity	Optional																		
iCompanyFrom	Optional																		
iSupplyTypeFrom	Optional																		
iSupplyCodeFrom	Optional																		
iShipFromBusinessPartner	Optional																		
iWarehouseOrderSeries	Optional																		
iWarehouseOrderType	Optional																		

	iProductionOrderSeries Optional iPurchaseOrderSeries Optional iPurchaseOrderType Optional iSalesOrderSeries Optional iSalesOrderType Optional iWarehouseAdviceSeries Optional iProductionAdviceSeries Optional iPurchaseAdviceSeries Optional iActivateWarehouseTransfer Optional iProcessWarehouseTransfer Optional iReleaseProductionOrder Optional iCombineOrderAdvice Optional iConfirmOrderAdvice Optional Output: oAdvice oOrderOrigin oOrderNumber oOrderLine oOrderSequence Return: 0 - Advice or Order has been created successfully. DALHOOKERROR - Advice/Order could not be created.

Kanban.StartGenerate

DLL:	whextinhapi This function is available from KB3629008 .
Syntax:	<pre>long Kanban.StartGenerate(domain tckbid iKanbanSignal, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl This function starts the session Generate Orders (Kanban) (whinh2200m00). Input: i.kanban.signal Kanban Signal for which an order or advice needs to be generated. Output: N.a. Return: 0 / DALHOOKERROR

Public Interfaces for TimePhasedOrderPoint

The following functions are available:

[TimePhasedOrderPoint.GenerateOrders](#)

TimePhasedOrderPoint.GenerateOrders

DLL:	whextinhapi																																																																														
	This function is available from KB3533941 .																																																																														
Syntax:	<pre>long TimePhasedOrderPoint.GenerateOrders(domain tcmcs.str15 iDevice, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref domain long oExceptionID)</pre>																																																																														
Usage:	Expl: This function creates warehouse, production, purchase and/or sales orders based on the Time Phased Order Point principle. Be aware that transaction management is handled within this function. Pre: N.a. Post: N.a. Input: iDevice Mandatory iProcessingOptionSet Optional, if 0, the default options are applied. A Processing Option Set can be created via a call to function ProcessingOptionSet.Create(). Processing Options have a direct relationship with the form fields on session Generate Orders (TPOP) (whinh2201m000) and are not explained in further detail here. Please refer to the session help for additional information. Processing Options that are set while a required Implemented Software Component is not available are ignored. Explanation about setting of defaults: Minimum Value: Minimum value of domain is taken as default value. Maximum Value: Maximum value of domain is taken as default value. <table><tr><th>NAME</th><th>TYPE</th><th>DEFAULT</th></tr><tr><td>ItemFrom</td><td>domain tcitem</td><td>Minimum Value</td></tr><tr><td>ItemTo</td><td>domain tcitem</td><td>Maximum Value</td></tr><tr><td>ItemTypeFrom</td><td>domain tckitm</td><td>Minimum Value</td></tr><tr><td>ItemTypeTo</td><td>domain tckitm</td><td>Maximum Value</td></tr><tr><td>ItemGroupFrom</td><td>domain tccitg</td><td>Minimum Value</td></tr><tr><td>ItemGroupTo</td><td>domain tccitg</td><td>Maximum Value</td></tr><tr><td>SiteFrom</td><td>domain tcsite</td><td>Minimum Value</td></tr><tr><td>SiteTo</td><td>domain tcsite</td><td>Maximum Value</td></tr><tr><td>WarehouseFrom</td><td>domain tccwar</td><td>Minimum Value</td></tr><tr><td>WarehouseTo</td><td>domain tccwar</td><td>Maximum Value</td></tr><tr><td>LocationFrom</td><td>domain whloca</td><td>Minimum Value</td></tr><tr><td>LocationTo</td><td>domain whloca</td><td>Maximum Value</td></tr><tr><td>WarehouseTypeFrom</td><td>domain tctypw</td><td>Minimum Value</td></tr><tr><td>WarehouseTypeTo</td><td>domain tctypw</td><td>Maximum Value</td></tr><tr><td>ABCFrom</td><td>domain tcabcc</td><td>Minimum Value</td></tr><tr><td>ABCTo</td><td>domain tcabcc</td><td>Maximum Value</td></tr><tr><td>SlowMovingPercFrom</td><td>domain tcprct</td><td>Minimum Value</td></tr><tr><td>SlowMovingPercTo</td><td>domain tcprct</td><td>Maximum Value</td></tr><tr><td>BuyFromPartnerFrom</td><td>domain tccom.bpid</td><td>Minimum Value</td></tr><tr><td>BuyFromPartnerTo</td><td>domain tccom.bpid</td><td>Maximum Value</td></tr><tr><td>OrderSystemFrom</td><td>domain tcosys</td><td>Minimum Value</td></tr><tr><td>OrderSystemTo</td><td>domain tcosys</td><td>Maximum Value</td></tr><tr><td>OrderMethodFrom</td><td>domain tcomth</td><td>Minimum Value</td></tr><tr><td>OrderMethodTo</td><td>domain tcomth</td><td>Maximum Value</td></tr><tr><td>Factor</td><td>domain tcfacr</td><td>0</td></tr></table>	NAME	TYPE	DEFAULT	ItemFrom	domain tcitem	Minimum Value	ItemTo	domain tcitem	Maximum Value	ItemTypeFrom	domain tckitm	Minimum Value	ItemTypeTo	domain tckitm	Maximum Value	ItemGroupFrom	domain tccitg	Minimum Value	ItemGroupTo	domain tccitg	Maximum Value	SiteFrom	domain tcsite	Minimum Value	SiteTo	domain tcsite	Maximum Value	WarehouseFrom	domain tccwar	Minimum Value	WarehouseTo	domain tccwar	Maximum Value	LocationFrom	domain whloca	Minimum Value	LocationTo	domain whloca	Maximum Value	WarehouseTypeFrom	domain tctypw	Minimum Value	WarehouseTypeTo	domain tctypw	Maximum Value	ABCFrom	domain tcabcc	Minimum Value	ABCTo	domain tcabcc	Maximum Value	SlowMovingPercFrom	domain tcprct	Minimum Value	SlowMovingPercTo	domain tcprct	Maximum Value	BuyFromPartnerFrom	domain tccom.bpid	Minimum Value	BuyFromPartnerTo	domain tccom.bpid	Maximum Value	OrderSystemFrom	domain tcosys	Minimum Value	OrderSystemTo	domain tcosys	Maximum Value	OrderMethodFrom	domain tcomth	Minimum Value	OrderMethodTo	domain tcomth	Maximum Value	Factor	domain tcfacr	0
NAME	TYPE	DEFAULT																																																																													
ItemFrom	domain tcitem	Minimum Value																																																																													
ItemTo	domain tcitem	Maximum Value																																																																													
ItemTypeFrom	domain tckitm	Minimum Value																																																																													
ItemTypeTo	domain tckitm	Maximum Value																																																																													
ItemGroupFrom	domain tccitg	Minimum Value																																																																													
ItemGroupTo	domain tccitg	Maximum Value																																																																													
SiteFrom	domain tcsite	Minimum Value																																																																													
SiteTo	domain tcsite	Maximum Value																																																																													
WarehouseFrom	domain tccwar	Minimum Value																																																																													
WarehouseTo	domain tccwar	Maximum Value																																																																													
LocationFrom	domain whloca	Minimum Value																																																																													
LocationTo	domain whloca	Maximum Value																																																																													
WarehouseTypeFrom	domain tctypw	Minimum Value																																																																													
WarehouseTypeTo	domain tctypw	Maximum Value																																																																													
ABCFrom	domain tcabcc	Minimum Value																																																																													
ABCTo	domain tcabcc	Maximum Value																																																																													
SlowMovingPercFrom	domain tcprct	Minimum Value																																																																													
SlowMovingPercTo	domain tcprct	Maximum Value																																																																													
BuyFromPartnerFrom	domain tccom.bpid	Minimum Value																																																																													
BuyFromPartnerTo	domain tccom.bpid	Maximum Value																																																																													
OrderSystemFrom	domain tcosys	Minimum Value																																																																													
OrderSystemTo	domain tcosys	Maximum Value																																																																													
OrderMethodFrom	domain tcomth	Minimum Value																																																																													
OrderMethodTo	domain tcomth	Maximum Value																																																																													
Factor	domain tcfacr	0																																																																													

Constant	domain	tcwttm	0
Unit	domain	tctoep	tctoep.days
IncludeSpecification	domain	tcyesno	tcyesno.no
MaxOrders	domain	tcffno	1
CombineInterval	domain	tcyesno	tcyesno.no
CombineOrders	domain	tcyesno	tcyesno.no
ActivateOrders	domain	tcyesno	tcyesno.no
DirectProcess	domain	tcyesno	tcyesno.no
SeriesWarehousing	domain	tcseri	Empty String
OrderTypeWarehousing	domain	tccotp	Empty String
SeriesProduction	domain	tcseri	Empty String
SeriesPurchase	domain	tcseri	Empty String
OrderTypePurchase	domain	tccotp	Empty String
SeriesSales	domain	tcseri	Empty String
OrderTypeSales	domain	tccotp	Empty String
Output: N.a.			
Return: 0 - Order(s) have been created successfully.			
DALHOOKERROR - Order could not be created.			

Public Interfaces for ProductionOrder

The following functions are available:

[ProductionOrder.GenerateHandlingUnitForLineSideLabeling](#)

[ProductionOrder.GenerateHandlingUnitForLineSideLabelingV2](#)

ProductionOrder.GenerateHandlingUnitForLineSideLabeling

DLL:	whextinhapi
	This function is available from KB2304620 .
Syntax:	<pre>long ProductionOrder.GenerateHandlingUnitForLineSideLabeling(domain tcorno iOrderNumber, domain tcuef.effn iEffectivityUnit, domain tcqstl iLabelQuantity, domain tccuni iLabelQuantityUnit, domain tcclot iLotCode, boolean iGenerateLot, domain whwmd.pkdf iPackageDefinition, domain tcitem iPackagingItem, domain tcqivl iPackagingItemQuantity, domain tcuef.mask iHandlingUnitMask,</pre>

	<pre> domain tctyesno iSSCCMask, domain tctyesno iShipmentMask, boolean iPublishLabel, domain tclabl iLabelLayout, domain tcmcs.long iNumberOfCopies, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function generates handling unit structure for production order in Line Side Labeling scenario. When generating lot is required Lot Code will be generated or derived from iLotCode and added in the Item - Lot table. Pre: db.retry.point() Post: commit/abort transaction. Input: - iOrderNumber - production order number (Mandatory) - iEffectivityUnit - Effectivity Unit (Optional) - iLabelQuantity - Quantity to be labeled (Mandatory) - iLabelUnit - Storage unit (Mandatory) - iLotCode - Lot Code (Optional) - iGenerateLot - Generate Lot (True/False) - iPackageDefinition - Package Definition (Optional) - iPackagingItem - Packaging Item (Mandatory if iPackageDefinition is empty) - iPackagingItemQuantity - Packaging Item Quantity (Mandatory if iPackagingItem is filled) - iHandlingUnitMask - Handling Unit Mask (Optional, default mask will be used if empty) - iSSCCMask - SSCC Handling Unit Mask (Mandatory) - iShipmentMask - Use shipment or internal mask (Mandatory) - iPublishLabel - Publish Package Label BOD (True/False) - iLabelLayout - Label Layout (Optional) - iNumberOfCopies - Number of Copies (Mandatory if iPublishLabel is True) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Handling Units are generated <> 0 - Generating Handling Units failed

ProductionOrder.GenerateHandlingUnitForLineSideLabelingV2

DLL:	whextinhapi
	This function is available from KB3565601 .
Syntax:	<pre> long ProductionOrder.GenerateHandlingUnitForLineSideLabelingV2 (domain tcorno iOrderNumber, domain tcuef.effn iEffectivityUnit, </pre>

	domain tcqstl iLabelQuantity, domain tccuni iLabelQuantityUnit, domain tcclot iLotCode, boolean iGenerateLot, domain whwmd.pkdf iPackageDefinition, domain tcitem iPackagingItem, domain tcqivl iPackagingItemQuantity, domain tcuef.mask iHandlingUnitMask, domain tcyesno iSSCCMask, domain tcyesno iShipmentMask, boolean iPrintLabels, domain whwmd.lbpb iLabelPrintedBy, domain tcmcs.str15 iDevice, domain tclabl iLabelLayout, domain tcmcs.long iNumberOfCopies, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function generates handling unit structure for production order in Line Side Labeling scenario. When generating lot is required Lot Code will be generated or derived from iLotCode and added in the Item - Lot table. Labels of generated handling units can be printed or published. Pre: db.retry.point() Post: commit/abort transaction. Input: - iOrderNumber - Production Order Number (Mandatory) - iEffectivityUnit - Effectivity Unit (Optional) - iLabelQuantity - Quantity to be labeled (Mandatory) - iLabelUnit - Storage unit (Mandatory) - iLotCode - Lot Code (Optional) - iGenerateLot - Generate Lot (True/False) - iPackageDefinition - Package Definition (Optional) - iPackagingItem - Packaging Item (Mandatory if iPackageDefinition is empty) - iPackagingItemQuantity - Packaging Item Quantity (Mandatory if iPackagingItem is filled) - iHandlingUnitMask - Handling Unit Mask (Optional, default mask will be used if empty) - iSSCCMask - SSCC Handling Unit Mask (Mandatory) - iShipmentMask - Use shipment or internal mask (Mandatory) - iPrintLabels - Print Labels (True/False) - iLabelPrintedBy - Label Printed By (Mandatory if iPrintLabels = True) Possible values are: - whwmd.lbpb.internal - Report will be printed - whwmd.lbpb.external - BOD will be published - iDevice - Report Device (Mandatory if iPrintLabels = True and iLabelPrintedBy = whwmd.lbpb.internal) - iLabelLayout - Label Layout (Mandatory if iPrintLabels = True) - iNumberOfCopies - Number of Copies (Mandatory if iPrintLabels = True) Output: - oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Handling Units are generated

	<> 0 - Generating Handling Units failed

Public Interfaces for CrossDockOrderLine

The following functions are available:

[CrossDockOrderLines.Print](#)

CrossDockOrderLines.Print

DLL:	whextinhapi								
	This function is available from KB3513678 .								
Syntax:	<pre>long CrossDockOrderLines.Print(domain tcorno iCrossDockOrder, domain tcpono iCrossDockOrderLine, domain tcmcs.str15 iDevice, long iProcessingOptionSet, ref boolean oDataProcessed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>								
Usage:	Expl: This public interface will print the cross-dock order lines for the iCrossDockOrder and iCrossDockOrderLine using the defaults or options as provided in the iProcessingOptionSet. Opening/Closing of the report is handled by this function. Pre: N.a. Post: N.a. Input: <table border="0"> <tr> <td>iCrossDockOrder</td><td>Optional</td></tr> <tr> <td>iCrossDockOrderLine</td><td>Optional, ignored if iCrossDockOrder is empty.</td></tr> <tr> <td>iDevice</td><td>Mandatory</td></tr> <tr> <td>iProcessingOptionSet</td><td>Optional, if 0, the default options are applied.</td></tr> </table> A Processing Option Set can be created via a call to function ProcessingOptionSet.Create(). In case iCrossDockOrder is filled then the following options of the iProcessingOptionSet will be ignored: - selection range fields (From/To) - CrossDockOrderLineArray The cross-dock order lines will then be printed for the given iCrossDockOrder (and specific iCrossDockOrderLine, if filled). In case option CrossDockOrderLineArray is set then the selection range fields (From/To) of the iProcessingOptionSet will be	iCrossDockOrder	Optional	iCrossDockOrderLine	Optional, ignored if iCrossDockOrder is empty.	iDevice	Mandatory	iProcessingOptionSet	Optional, if 0, the default options are applied.
iCrossDockOrder	Optional								
iCrossDockOrderLine	Optional, ignored if iCrossDockOrder is empty.								
iDevice	Mandatory								
iProcessingOptionSet	Optional, if 0, the default options are applied.								

ignored.

The cross-dock order lines will then be printed for the cross-dock order lines in the array.

Processing Options have a direct relationship with the form fields on session Print Cross-dock Order Lines (whinh6410m000) and are not explained in further detail here.
Please refer to the session help for additional information.
Processing Options that are set while a required Implemented Software Component is not available are ignored.

Explanation about setting of defaults:

Minimum Value: Minimum value of domain is taken as default value.

Maximum Value: Maximum value of domain is taken as default value.

NAME	TYPE	DEFAULT
WarehouseFrom	domain tccwar	Minimum Value
WarehouseTo	domain tccwar	Maximum Value
StagingLocationFrom	domain whloca	Minimum Value
StagingLocationTo	domain whloca	Maximum Value
ItemFrom	domain tcitem	Minimum Value
ItemTo	domain tcitem	Maximum Value
CrossDockOrderFrom	domain tcorno	Minimum Value
CrossDockOrderTo	domain tcorno	Maximum Value
CrossDockOrderTypeFrom	domain whinh.cdt	Minimum Value
CrossDockOrderTypeTo	domain whinh.cdt	Maximum Value
CrossDockOrderStatusFrom	domain whinh.cd	Minimum Value
CrossDockOrderStatusTo	domain whinh.cd	Maximum Value
OutboundOrderOriginFrom	domain whinh.oorg	Minimum Value
OutboundOrderOriginTo	domain whinh.oorg	Maximum Value
OutboundOrderNumberFrom	domain tcorno	Minimum Value
OutboundOrderNumberTo	domain tcorno	Maximum Value
OutboundOrderLineFrom	domain tcpono	Minimum Value
OutboundOrderLineTo	domain tcpono	Maximum Value
PlannedDeliveryDateFrom	domain tcdate	Minimum Value
PlannedDeliveryDateTo	domain tcdate	Current Date/Time
CrossDockOrderLineFrom	domain tcpono	Minimum Value
CrossDockOrderLineTo	domain tcpono	Maximum Value
InboundOrderOriginFrom	domain whinh.oorg	Minimum Value
InboundOrderOriginTo	domain whinh.oorg	Maximum Value
InboundOrderNumberFrom	domain tcorno	Minimum Value
InboundOrderNumberTo	domain tcorno	Maximum Value
InboundOrderLineFrom	domain tcpono	Minimum Value
InboundOrderLineTo	domain tcpono	Maximum Value
PlannedReceiptDateFrom	domain tcdate	Minimum Value
PlannedReceiptDateTo	domain tcdate	Maximum Value
CrossDockOrderLineStatusFrom	domain whinh.cdls	Minimum Value
CrossDockOrderLineStatusTo	domain whinh.cdls	Maximum Value
ReportNumber	domain tcmcs.long	1
CrossDockOrderLineArray	domain ttjson	0
ReportName	domain tcmcs.str16	Empty String

Possible values of ReportNumber are:

1 - By Cross-dock Line

JSON Object CrossDockOrderLineArray has the following structure:

```
"CrossDockOrderLineArray": [
  {
    "OrderNumber": "COD000006"
    "OrderLine": 10,
  },
  {
```

Run.DetermineNumber

DLL:	whextinhapi
	This function is available from KB3541252 .
Syntax:	<pre>long Run.DetermineNumber(domain whinh.kofr iRunType, domain tclogn iUser.</pre>

	<pre> ref domain whinh.btno oRunNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface will determine the first available run number. Pre: N.A. Post: N.A. Input: iRunType - Run Type (Mandatory) 1 - Inbound 2 - Outbound iUser - User Login (Optional) Output: oRunNumber - Determined Run Number oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0/DALHOOKERROR

Chapter 26 Public Interfaces for Freight

Public Interfaces for FreightOrderCluster

The following functions are available:

[FreightOrderCluster.ConfirmDelivery](#)

FreightOrderCluster.ConfirmDelivery

DLL:	fmextlbdapi						
Syntax:	<pre> long FreightOrderCluster.ConfirmDelivery(domain tcorno iFreightOrderCluster, domain tcdate iActualLoadDate, domain tcdate iActualUnloadDate, domain fmlbd.conv iShippedOrCompleted, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>						
Usage:	Expl: This public interface can be used to set the status of the Freight Order Cluster and it's Cluster Lines to Shipped or Completed, based on the iShippedOrCompleted indicator. This public interface will have an internal transaction handling, so calling this Public Interface inside a transaction is not allowed, to prevent this from happening, this public interface will not work when called inside a logical transaction Pre: This Public Interface will update records and as such must be called within a transaction. When implementing this Public Interface, please be aware of the performance aspect. In standard, this functionality will have it's own transaction management, while the Public Interface does not!. Therefore it is strongly advisable to call this Public Interface inside a separate transaction and after returning directly perform the abort in case of errors and the commit in case this function returns 0. Long logical transaction must be prevented as records will be locked and are not available for other transactions! Post: Exception handling can be done, when needed. Input: <table border="0"> <tr> <td>iFreightOrderCluster</td><td>- Freight Order Cluster: Mandatory</td></tr> <tr> <td>iActualLoadDate</td><td>- Actual Load Date: Mandatory when iShippedOrComplete is set to Shipped</td></tr> <tr> <td>iActualUnloadDate</td><td>- Actual Unload Date: Mandatory when</td></tr> </table>	iFreightOrderCluster	- Freight Order Cluster: Mandatory	iActualLoadDate	- Actual Load Date: Mandatory when iShippedOrComplete is set to Shipped	iActualUnloadDate	- Actual Unload Date: Mandatory when
iFreightOrderCluster	- Freight Order Cluster: Mandatory						
iActualLoadDate	- Actual Load Date: Mandatory when iShippedOrComplete is set to Shipped						
iActualUnloadDate	- Actual Unload Date: Mandatory when						

	iShippedOrCompleted Output: oExceptionMessage oExceptionID Return: 0 <> 0	iShippedOrComplete is set to Complete - Confirm the delivery of the Freight Order Cluster to either Shipped (fmlbd.conv.shipped) or Completed (fmlbd.conv.completed) - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Confirm Delivery succeeded - Error

Public Interfaces for FreightPlanning

The following functions are available:

[FreightPlanning.ActualizePlan](#)

[FreightPlanning.GenerateForSpecificOrderSelection](#)

[FreightPlanning.MoveShipmentToLoad](#)

FreightPlanning.ActualizePlan

DLL:	fmextlbdapi	
Syntax:	<pre>long FreightPlanning.ActualizePlan(domain tcorno iPlan, boolean iReplanAllowed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: This Public Interface will actualize the given plan Pre: db.retry.point is set Post: transaction handling (abort/commit) exception handling can be done when applicable Input: iPlan iReplanAllowed Output: oExceptionMessage - Plan which must be actualized: Mandatory - Should potential freight order lines in the plan which must be replanned be replanned before actualization takes place, true or false. - The last message if the return	

	value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

FreightPlanning.GenerateForSpecificOrderSelection

DLL:	fmextlbdapi
Syntax:	<pre> long FreightPlanning.GenerateForSpecificOrderSelection(domain tccwoc iShippingOffice, domain fmfoc.cplg iPlanningGroup, domain fmlbd.algo iPlanningAlgorithm, domain fmlbd.dtdr iShipmentDatesBasedOn, domain fmlbd.rsel iCarrierSelectionCriterion, domain tcmcs.long iNumberOfSpecificOrders, ref domain tcorno iSpecificOrders() fixed, ref domain boolean iSpecificOrderLineSelection, ref domain tcpono iSpecificOrderLines(), domain tcdsca iPlanningDescription mb, domain tcyesno iPlanningAlgorithmBinding, domain tcyesno iFreightManagementLeadingPlan, domain tcyesno iAllowMeansOfTransportInMultiplePlans, domain tcyesno iCalculateAdditionalCosts, domain tcyesno iReplan, domain tcorno iPlanForReplan, domain tcyesno iOnlyReplanLines, domain fmfoc.lnst iReplanAdditionThroughStatus, domain fmlbd.replan iReplanningMethod, domain tcyesno iCommittedInventoryOnly, domain tcyesno iDetailedPlanningLog, domain boolean iShowProgressIndicator, ref domain tcorno oPlan, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface allows to generate a freight plan for a selection of freight orders and lines. It is assumed the freight orders that are present in the order arrays are having the status 'Expected' or 'Planned'. When planning is not possible due to unreachable addresses or low capacity of transport means, the plan will still be generated till the stage it fails to plan, this will lead to an unusable plan, but will allow for user analysis via the planning log. This public interface will not print the planning reports which are printed normally in the session "Generate Plan" (fmlbd0280m000). Errors / information messages will be available in the freight planning log, session "Planning Log" (fmlbd0530m000). Pre: This Public Interface has it's own transaction management, as an exception to the standard Public Interfaces. As generate plan

	can be a long running transaction, the Public Interface should not lock records too long. Exception handling can be done when applicable Post: N.a. Input: - iShippingOffice - Shipping Office: Mandatory - iPlanningGroup - Planning Group: Mandatory - iPlanningAlgorithm - Planning Algorithm: Mandatory - Possible values: - Direct Shipping (fmlbd.algo.sing) - Consolidation (fmlbd.algo.cons) - Pooling (fmlbd.algo.pool) - iShipmentDatesBasedOn - Shipping Dates based On: Mandatory - Possible values: - Earliest of Possible Dates (fmlbd.dtdr.earliest) - Latest of Possible Dates (fmlbd.dtdr.latest) - Minimum of Planned Unload Dates (fmlbd.dtdr.minimum) - Average of Planned Unload Dates (fmlbd.dtdr.average) - iCarrierSelectionCriterion - Carrier Selection: Mandatory - Possible values: - Cheapest (fmlbd.rsel.cheap) - Fastest (fmlbd.rsel.fast) - Shortest (fmlbd.rsel.short) - iNumberOfSpecificOrders - Number of specific orders in the specific order array: Mandatory - iSpecificOrders - Array containing the freight order headers for which planning must be generated. - iSpecificOrderLineSelection - Apply specific line selection - This is to be used when specific freight order lines must be planned, when complete freight orders are to be planned it is not required to pass all the freight order lines. - iSpecificOrderLines - Array containing the freight order lines for which planning must be generated. - iPlanningDescription - Description of the freight plan - iPlanningAlgorithmBinding - Planning Algorithm Binding, if planning is not possible for a certain algorithm, the logic will plan for other algorithms as well. This flag can disable that behavior. - iFreightManagementLeadingPlan - Create a freight management leading load plan. - iAllowMeansOfTransportInMultiplePlans - Allow the means of transport to be planned in multiple plans. - iCalculateAdditionalCosts - Calculate Additional Costs for the generated freight plan - iReplan - Replanning - iPlanForReplan - Plan for Replanning: Mandatory when iReplan is set to Yes - iOnlyReplanLines - Only Replan Freight Order Lines to be Replanned.

	<code>iReplanAdditionThroughStatus</code> - Replanning, allow updating of existing Loads/Shipments till this status. <code>iReplanningMethod</code> - Replanning Method Possible Values - Replan Freight Order Lines Separatly (<code>fmlbd.replan.separate</code>) - Combine with Planned Freight Orders (<code>fmlbd.replan.combine</code>) <code>iCommittedInventoryOnly</code> - Plan for Committed Inventory Only <code>iDetailedPlanningLog</code> - Plan with the detailed planning log <code>iShowProgressIndicator</code> - Show the progress indicator to the user, this will give the user an indication on the process flow and the time required to complete the generation of the freight plan. Output: <code>oPlan</code> - The generated plan <code>oExceptionMessage</code> - The last message if the return value is not equal to 0. If more than one message is given, these are present in the <code>oExceptionID</code> <code>oExceptionID</code> - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

FreightPlanning.MoveShipmentToLoad

DLL:	fmextlbdapi
Syntax:	<pre>long FreightPlanning.MoveShipmentToLoad(domain tcorno iShipment, domain tcorno iDestinationLoad, boolean iAllowMergeShipmentLines, boolean iDeleteEmptyLoad, boolean iCalculateFreightCostsIfInteractive, boolean iCalculateAdditionalCosts, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This Public Interface will move a Shipment to a Load. Pre: db.retry.point is set Post: transaction handling (abort/commit) exception handling can be done when applicable Input: <code>iShipment</code> - Shipment which must be moved: Mandatory <code>iDestinationLoad</code> - Load to which the shipment must be moved: Mandatory <code>iAllowMergeShipmentLines</code> - When <code>iDestinationLoad</code> already has a shipment line which has the same freight order line of any of the <code>iSourceShipment</code> lines, the lines must be merged. This flag determines if that should be allowed (true/false) <code>iDeleteEmptyLoad</code> - If the load from which the

	iSourceShipment is moved becomes empty then it will be possible to remove the empty load with the setting of this flag. iCalculateFreightCostsIfInteractive - If freight costs calculation is setup to be recalculated interactively in the parameters, in the standard a question will be raised, with this flag the calculation in such cases can be set to true or false. iCalculateAdditionalCosts - If freight costs are to be calculated should the additional costs also be calculated, true or false. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0: OK, <> 0: Error

Public Interfaces for Load

The following functions are available:

[Load.ConfirmDelivery](#)

Load.ConfirmDelivery

DLL:	fmextlbdapi		
Syntax:	long	Load.ConfirmDelivery(	
		domain tcorno	iLoad,
		domain tcdate	iActualLoadDate,
		domain tcdate	iActualUnloadDate,
		domain fmlbd.conv	iShippedOrCompleted,
	ref	domain tcmcs.s999m	oExceptionMessage mb,
	ref	long	oExceptionID)
Usage:	Expl:	This public interface can be used to set the status of the load and it's shipments and shipment lines to Shipped or Completed, based on the iShippedOrCompleted indicator.	
	Pre:	This Public Interface will update records and as such must be	

	called within a transaction! When implementing this Public Interface, please be aware of the performance aspect. In standard, this functionality will have it's own transaction management, while the Public Interface does not!. Therefore it is strongly advisable to call this Public Interface inside a seperate transaction and after returning directly perform the abort in case of errors and the commit in case this function returns 0. Long logical transaction must be prevented as records will be locked and are not available for other transactions! Post: Exception handling can be done, when needed. Input: iLoad - Load: Mandatory iActualLoadDate - Actual Load Date: Mandatory when iShippedOrComplete is set to Shipped iActualUnloadDate - Actual Unload Date: Mandatory when iShippedOrComplete is set to Complete iShippedOrCompleted - Confirm the delivery of the Load to either Shipped (fmlbd.conv.shipped) or Completed (fmlbd.conv.completed) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Confirm Delivery succeeded <> 0 - Error

Public Interfaces for Shipment

The following functions are available:

[Shipment.ConfirmDelivery](#)

Shipment.ConfirmDelivery

DLL:	fmextlbdapi		
Syntax:	<pre>long Shipment.ConfirmDelivery(domain tcorno iShipment, domain tcdate iActualLoadDate, domain tcdate iActualUnloadDate, domain fmlbd.conv iShippedOrCompleted, ref domain tcmcs.s999m oExceptionMessage mb,</pre>		

	ref	long	oExceptionID)
Usage:	Expl:	This public interface can be used to set the status of the shipment and it's shipment lines to Shipped or Completed, based on the iShippedOrCompleted indicator.	
	Pre:	This Public Interface will update records and as such must be called within a transaction. When implementing this Public Interface, please be aware of the performance aspect. In standard, this functionality will have it's own transaction management, while the Public Interface does not!. Therefore it is strongly advisable to call this Public Interface inside a seperate transaction and after returning directly perform the abort in case of errors and the commit in case this function returns 0. Long logical transaction must be prevented as records will be locked and are not available for other transactions!	
	Post:	Exception handling can be done, when needed.	
	Input:	iShipment	- Shipment: Mandatory
		iActualLoadDate	- Actual Load Date: Mandatory when iShippedOrComplete is set to Shipped
		iActualUnloadDate	- Actual Unload Date: Mandatory when iShippedOrComplete is set to Complete
		iShippedOrCompleted	- Confirm the delivery of the Shipment to either Shipped (fmlbd.conv.shipped) or Completed (fmlbd.conv.completed)
	Output:	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
		oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
	Return:	0	- Confirm Delivery succeeded
		<> 0	- Error

Chapter 27 Public Interfaces for Service

Public Interfaces for Service

The following functions are available:

[Service.DetermineStartOrFinishTime](#)

[Service.GetClaimSettings](#)

[Service.GetGeneralSettings](#)

[Service.GetMaintenanceSalesOrderSettings](#)

[Service.GetMobileServiceSettings](#)

[Service.GetPlanningSettings](#)

[Service.GetQuoteSettings](#)

[Service.GetResourceCalendarCode](#)

[Service.GetServiceOrderSettings](#)

[Service.GetWorkOrderSettings](#)

Service.DetermineStartOrFinishTime

DLL:	tsextsocapi		
Syntax:	<pre>long Service.DetermineStartOrFinishTime(domain tcncmp iLogisticCompany, domain tsbsc.clst iInstallationGroup, domain tcitem iItem, domain tcibd.sern iSerialNumber, domain tcemno iAssignedServiceEngineer, domain tccmno iPreferredServiceEngineer, domain tccwoc iServiceDepartment, domain tcccpc.ccap iDuration, domain tcccpc.ract iAvailabilityType, domain tcplcd iPlanMethod, domain tsmdm.pldt iTime,</pre>		

	ref	domain	tsmdm.pldt	oTime,
	ref	domain	tcmscs.s999m	oExceptionMessage mb,
	ref		long	oExceptionID)
Usage:	Expl. : This function determines a finish time for a given start time or a start time for a given finish time: - When plan method is 'plan forward' the start time must be given and the finish time is calculated. - When plan method is 'plan backward' the finish time must be given and the start time is calculated. The calendar code and time zone used for planning are retrieved in the following way. When the parameter 'Use Engineer Calendar for Planning' is No, the calendar/timezone is searched in the following order: 1. Serialized Item 2. Installation Group 3. Assigned Service Engineer 4. Preferred Service Engineer 5. Service Center 6. Company This means that the Serialized Item calendar is retrieved first. When no Serialized Item calendar is available the calendar of the Installation Group is retrieved. And so on. When for the first 5 levels no calendar is found, finally the Company calendar/timezone is retrieved; this calendar/timezone is mandatory, so is always available. When the parameter 'Use Engineer Calendar for Planning' is Yes, the calendar/timezone is searched in the following order: 1. Assigned Service Engineer 2. Preferred Service Engineer 3. Service Center 4. Serialized item 5. Installation Group 6. Company Pre : None Post : None Input : - iLogisticCompany - Logistic Company: Mandatory - iInstallationGroup - Installation Group: Not Mandatory - iItem - Item: Not Mandatory - iSerialNumber - Serial Number: Not Mandatory - iAssignedServiceEngineer - Assigned Service Engineer: Not Mandatory - iPreferredServiceEngineer - Preferred Service Engineer: Not Mandatory - iServiceDepartment - Service Department: Not Mandatory - iDuration - Duration of Activity: Not Mandatory The Duration is expressed in the Time Duration Unit from the General Service Parameters - iAvailabilityType - Availability Type: Not Mandatory - iPlanMethod - Planning Method (forward/backward): Mandatory - iTime - Base Time (planned start/			

Output : oTime oExceptionMessage oExceptionID Return: 0 <> 0	finish date): Not Mandatory - Result Time (planned start/finish date) - The last message if the return value unequals 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Data read - An error occurred

Service.GetClaimSettings

DLL:	tsxtcmmap
Syntax:	<pre> long Service.GetClaimSettings(domain tcncmp iLogisticCompany, domain tccwoc iServiceOffice, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves claim settings. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tscmm000), or office or site level (settings table tscmm003) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when ATP check enabled and default task are needed, the function must be called as follows: <pre> if service.GetClaimSettings(* Fixed arguments: company, --> input service office, --> input site, --> input force.read, --> input context is master data --> input </pre>

	<pre> exception.message, --> output exception.id, --> output /* Variable arguments: "atpr", --> input atp.check.enabled, --> output "ctsk", --> input default.task) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif Pre: None Post: None Input: iLogisticCompany - Logistic Company: Mandatory iSite - Site: Not Mandatory iServiceOffice - Service Office: Not Mandatory iForceRead - Option to force new query in stead of using cached information iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - An error occurred </pre>

Service.GetGeneralSettings

DLL:	tsxtmdmapi
Syntax:	<pre> long Service.GetGeneralSettings(domain tcncmp iLogisticCompany, domain tccwoc i.ServiceOffice, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves general settings for service. Individual fields are retrieved, multiple fields can be retrieved

	in one call. This function is Multi Site aware, and data is read from general level (parameter table tsmdm000), or office or site level (settings table tsmdm003) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when distance unit and discount policy are needed, the function must be called as follows: <pre> if service.GetGeneralSettings(/* Fixed arguments: company, --> input service office, --> input site, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output /* Variable arguments: "undi", --> input distance.unit, --> output "dipy", --> input discount.policy) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif </pre> Pre: None Post: None Input: <table> <tr> <td>iLogisticCompany</td><td>- Logistic Company: Mandatory</td></tr> <tr> <td>iSite</td><td>- Site: Not Mandatory</td></tr> <tr> <td>i.ServiceOffice</td><td>- Service Office: Not Mandatory</td></tr> <tr> <td>iForceRead</td><td>- Option to force new query in stead of using cached information</td></tr> <tr> <td>iContextIsMasterData</td><td>- True/False: Indicates if call is done to retrieve data for master data or transactional data.</td></tr> <tr> <td>...</td><td>- The field mnemonic of the required field.</td></tr> </table> Output: <table> <tr> <td>oExceptionMessage</td><td>- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID</td></tr> <tr> <td>oExceptionID</td><td>- An ID that refers to all error information. Use the functions in Exception to get all relevant</td></tr> </table>	iLogisticCompany	- Logistic Company: Mandatory	iSite	- Site: Not Mandatory	i.ServiceOffice	- Service Office: Not Mandatory	iForceRead	- Option to force new query in stead of using cached information	iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.	...	- The field mnemonic of the required field.	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant
iLogisticCompany	- Logistic Company: Mandatory																
iSite	- Site: Not Mandatory																
i.ServiceOffice	- Service Office: Not Mandatory																
iForceRead	- Option to force new query in stead of using cached information																
iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.																
...	- The field mnemonic of the required field.																
oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID																
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant																

	<pre> ... Return: 0 <> 0 </pre>	<pre> information. - The value of the required field. - Data read - An error occurred </pre>

Service.GetMaintenanceSalesOrderSettings

DLL:	tsxtmscapi
Syntax:	<pre> long Service.GetMaintenanceSalesOrderSettings(domain tcncmp iLogisticCompany, domain tccwoc iServiceOffice, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves maintenance sales order settings. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tsmc000), or office or site level (settings table tsmc003) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: - Fields that are only defined at parameter level (not defined per office or site) are also retrieved. - This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when Sales Margin and Total Invoice are needed, the function must be called as follows: <pre> if service.GetMaintenanceSalesOrderSettings(* Fixed arguments: company, --> input service office, --> input site, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output * Variable arguments: "slmr", --> input sales.margin, --> output "tinv", --> input total.invoice) <> 0 then --> output * Error, do something </pre>

	<pre> Exception.Delete(exception.id) endif Pre: None Post: None Input: iLogisticCompany - Logistic Company: Mandatory iSite - Site: Not Mandatory iServiceOffice - Service Office: Not Mandatory iForceRead - Option to force new query in stead of using cached information iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - An error occurred </pre>

Service.GetMobileServiceSettings

DLL:	tsextsocapi
Syntax:	<pre> long Service.GetMobileServiceSettings(domain tcncmp iLogisticCompany, domain tccwoc iServiceOffice, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves mobile service settings. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tssoc002), or office or site level (settings table tssoc004) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: Fields that are only defined at parameter level (not

defined per office or site) are also retrieved.
 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value.

Example: when issue material and distribute visist report are needed, the function must be called as follows:

```
if service.GetMobileServiceSettings(
    /* Fixed arguments:

        company,                --> input
        service office,         --> input
        site,                   --> input
        force.read,             --> input
        context is master data  --> input
        exception.message,      --> output
        exception.id,           --> output

    /* Variable arguments:

        "issm",                  --> input
        issue.material,          --> output
        "dmvr",                  --> input
        distribute.visit.report) <> 0 then --> output
    /* Error, do something
        Exception.Delete(exception.id)
endif
```

Pre: None

Post: None

Input:

iLogisticCompany	- Logistic Company: Mandatory
iSite	- Site: Not Mandatory
iServiceOffice	- Service Office: Not Mandatory
iForceRead	- Option to force new query in stead of using cached information
iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.
...	- The field mnemonic of the required field.

Output:

oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
...	- The value of the required field.

Return: 0

<> 0	- An error occurred

Service.GetPlanningSettings

DLL:	tsextspcapi								
Syntax:	<pre> long Service.GetPlanningSettings(domain tcncmp iLogisticCompany, domain tccwoc iServiceOffice, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>								
Usage:	Expl: This function retrieves planning settings. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tsspc000), or office or site level (settings table tsspc003) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when step size for activities and unit for day are needed, the function must be called as follows: <pre> if service.GetPlanningSettings(/* Fixed arguments: company, --> input service office, --> input site, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output /* Variable arguments: "ssac", --> input step.size.for.activities, --> output "undy", --> input unit.for.day) <> 0 then --> output /* Error, do something Exception.Delete(exception.id) endif </pre> Pre: None Post: None Input: <table> <tr> <td>iLogisticCompany</td><td>- Logistic Company: Mandatory</td></tr> <tr> <td>iSite</td><td>- Site: Not Mandatory</td></tr> <tr> <td>iServiceOffice</td><td>- Service Office: Not Mandatory</td></tr> <tr> <td>iForceRead</td><td>- Option to force new query in stead of</td></tr> </table>	iLogisticCompany	- Logistic Company: Mandatory	iSite	- Site: Not Mandatory	iServiceOffice	- Service Office: Not Mandatory	iForceRead	- Option to force new query in stead of
iLogisticCompany	- Logistic Company: Mandatory								
iSite	- Site: Not Mandatory								
iServiceOffice	- Service Office: Not Mandatory								
iForceRead	- Option to force new query in stead of								

	using cached information iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - An error occurred

Service.GetQuoteSettings

DLL:	tsexteppapi
Syntax:	<pre> long Service.GetQuoteSettings(domain tcncmp iLogisticCompany, domain tccwoc iServiceOffice, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves Quote settings. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tsepp000), or office or site level (settings table tsepp003) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when quote duration and proposal method are needed, the function must be called as follows: <pre> if service.GetQuoteSettings(* Fixed arguments: </pre>

	<pre> company, --> input service office, --> input site, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output * Variable arguments: "qttm", --> input quote.duration, --> output "prmd", --> input proposal.method) <> 0 then --> output * Error, do something Exception.Delete(exception.id) endif Pre: None Post: None Input: iLogisticCompany - Logistic Company: Mandatory iSite - Site: Not Mandatory iServiceOffice - Service Office: Not Mandatory iForceRead - Option to force new query in stead of using cached information iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - An error occurred </pre>

Service.GetResourceCalendarCode

DLL:	tsxtsocapi
Syntax:	<pre> long Service.GetResourceCalendarCode(domain tcncmp iLogisticCompany, domain tsclm.crtc iResponseType, domain tsbsc.clst iInstallationGroup, domain tcitem iItem, </pre>

	domain tcibd.sern iSerialNumber, domain tcemno iAssignedServiceEngineer, domain tcemno iPreferredServiceEngineer, domain tccwoc iServiceDepartment, ref domain tcccp.ccal oCalendarCode, ref domain tczone oTimeZone, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl. : This function determines a calendar code and the time zone that is related to the calendar. The calendar code and time zone are retrieved in the following way. When the parameter 'Use Engineer Calendar for Planning' is No, the calendar/timezone is searched in the following order: 1. Response Type 2. Serialized Item 3. Installation Group 4. Assigned Service Engineer 5. Preferred Service Engineer 6. Service Center 7. Company This means that the Response Type calendar is retrieved first. When no Response Type calendar is available the calendar of the Serialized Item is retrieved. And so on. When for the first 6 levels no calendar is found, finally the Company calendar/timezone is retrieved; this calendar/timezone is mandatory, so is always available. When the parameter 'Use Engineer Calendar for Planning' is Yes, the calendar/timezone is searched in the following order: 1. Assigned Service Engineer 2. Preferred Service Engineer 3. Service Center 4. Response Type 5. Serialized item 6. Installation Group 7. Company Pre : None Post : None Input : iLogisticCompany - Logistic Company: Mandatory iResponseType - Response Type: Not Mandatory iInstallationGroup - Installation Group: Not Mandatory iItem - Item: Not Mandatory iSerialNumber - Serial Number: Not Mandatory iAssignedServiceEngineer - Assigned Service Engineer: Not Mandatory iPreferredServiceEngineer - Preferred Service Engineer: Not Mandatory iServiceDepartment - Service Department: Not Mandatory Output : oCalendarCode - Calendar Code oTimeZone - Time Zone oExceptionMessage - The last message if the return value unequals 0. If more than one message is given, these are present in

	oExceptionID Return: 0 <> 0	the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Data read - An error occurred

Service.GetServiceOrderSettings

DLL:	tsextsocapi	
Syntax:	<pre>long Service.GetServiceOrderSettings(domain tcncmp iLogisticCompany, domain tccwoc iServiceOffice, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...)</pre>	
Usage:	Expl: This function retrieves service orders settings. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tssoc000), or office or site level (settings table tssoc003) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site. Notes: 1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved. 2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value. Example: when use margin control unit and lower margin are needed, the function must be called as follows: <pre>if service.GetServiceOrderSettings(* Fixed arguments: company, --> input service office, --> input site, --> input force.read, --> input context is master data --> input exception.message, --> output exception.id, --> output * Variable arguments: "sqmc", --> input use.margin.control, --> output</pre>	

	<pre> "lwgm", --> input lower.margin) <> 0 then --> output * Error, do something Exception.Delete(exception.id) endif Pre: None Post: None Input: iLogisticCompany - Logistic Company: Mandatory iSite - Site: Not Mandatory iServiceOffice - Service Office: Not Mandatory iForceRead - Option to force new query in stead of using cached information iContextIsMasterData - True/False: Indicates if call is done to retrieve data for master data or transactional data. ... - The field mnemonic of the required field. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. ... - The value of the required field. Return: 0 - Data read <> 0 - An error occurred </pre>

Service.GetWorkOrderSettings

DLL:	tsextwcsapi
Syntax:	<pre> long Service.GetWorkOrderSettings(domain tcncmp iLogisticCompany, domain tccwoc iServiceOffice, domain tcsite iSite, boolean iForceRead, boolean iContextIsMasterData, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function retrieves work order settings. Individual fields are retrieved, multiple fields can be retrieved in one call. This function is Multi Site aware, and data is read from general level (parameter table tswcs000), or office or site level (settings table tswcs003) depending on the specified input field(s) (mnemonic), input arguments Site and Office and implementation phase of Multi Site.

Notes:

1. Fields that are only defined at parameter level (not defined per office or site) are also retrieved.
2. This function uses variable arguments, for input and output. Data is retrieved via name value pairs: specify the field and the field value.

Example: when commitment required and default labor type are needed, the function must be called as follows:

```

if service.GetWorkOrderSettings(
    /* Fixed arguments:

        company,                --> input
        service office,         --> input
        site,                   --> input
        force.read,             --> input
        context is master data  --> input
        exception.message,      --> output
        exception.id,           --> output

    /* Variable arguments:

        "ccrl",                  --> input
        commitment.required,     --> output
        "chlt",                  --> input
        default.labor.type) <> 0 then --> output
    /* Error, do something
        Exception.Delete(exception.id)
endif

```

Pre: None

Post: None

Input:

iLogisticCompany	- Logistic Company: Mandatory
iSite	- Site: Not Mandatory
iServiceOffice	- Service Office: Not Mandatory
iForceRead	- Option to force new query in stead of using cached information
iContextIsMasterData	- True/False: Indicates if call is done to retrieve data for master data or transactional data.
...	- The field mnemonic of the required field.

Output:

oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
...	- The value of the required field.
Return: 0	- Data read
<> 0	- Otherwise.

Public Interfaces for SerializedItem

The following functions are available:

[SerializedItem.ChangeStatus](#)

[SerializedItem.GetActiveContracts](#)

[SerializedItem.GetSupplierWarrantyEndDate](#)

[SerializedItem.GetWarrantyEndDate](#)

[SerializedItem.StartMultiMain](#)

[SerializedItem.StartOverview](#)

[SerializedItem.Supersede](#)

SerializedItem.ChangeStatus

DLL:	tsxtcfapi
	This function is available from KB3624238 .
Syntax:	<pre> long SerializedItem.ChangeStatus (domain tcitem iItem, domain tcibd.sern iSerialNumber, domain tscfg.cfst iChangeToStatus, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl. : This function changes the status of a Serialized Item and therefore provides the same functionality as session tscfg2230m000. The following new statuses are supported: - Active - Defective - Working Condition - To be Recycled - Removed This function can only be used if the current status of the serialized item is not Startup, Revision or Superseded, because this functionality is only used to set a physical status value. The administrative status can be set in tscfg2100m000. When changing the status of a Serialized Item to Removed, the item will automatically be removed from the Installation Group. Pre : A db.retry.point() must have been specified. Post : An abort.transaction() or commit.transaction() must be executed. Input : - iItem - The Item of which the status must be changed.

	- iSerialNumber - The Serial Number of which the status must be changed; Mandatory - iChangeToStatus - The Status that the serialized item must be changed to; Mandatory Allowed statuses: - Active - Defective - Working Condition - To be Recycled - Removed Output : - ExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return : 0 - No error <> 0 - An error occurred

SerializedItem.GetActiveContracts

DLL:	tsxtctmapi
	This function is available from KB2274297 .
Syntax:	<pre>long SerializedItem.GetActiveContracts(domain tcitem iItem, domain tcibd.sern iSerialNumber, ref long oNumberOfCoverageContracts, ref domain tcorno oCoverageContracts() fixed, ref domain tsmdm.desc oCoverageContractDescriptions() fixed mb, ref domain tccom.bpid oCoverageBusinessPartners() fixed, ref long oNumberOfPricingContracts, ref domain tcorno oPricingContracts() fixed, ref domain tsmdm.desc oPricingContractDescriptions() fixed mb, ref domain tccom.bpid oPricingBusinessPartners() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl. : This function will check if the specified Configuration is covered by any Coverage or Pricing Contract. Details of the most specific contract are returned. Searching for a contract is done in the following order: 1. Search for a Contract for the Serialized Item 2. Search for a Contract for the Parent Serialized Items in the Physical Breakdown Tree if any. 3. Search for a Contract for the Installation Group of the Serialized Item. 4. Search for a Contract for just the Item code. 5. Search for a Contract for just the Item codes of the Parent Serialized Items in the Physical Breakdown Tree if any.

	A Serialized Item can be covered by a contract for the Owner of the Serialized Item. When this is the case, this contract is returned. As a result, the output arrays are filled with 0, or 1 contract, indicated by the output arguments o.nr.of.coverage.contracts and o.nr.of.pricing.contracts respectively. The set up with output arrays is for a future enhancement. Then it becomes possible to have a contract for the Owner and for the In-use by Business Partner, simultaneously. When a contract is found, it is for the Owner/Sold-to Business Partner of the Serialized Item. The Contract Descriptions, and the related Business Partners, are also returned. Pre : Serialized Item should be passed as input. Arrays must be declared as based. Post : Free memory Input : iItem Item; Mandatory iSerialNumber Serial Number; Mandatory Output : oNumberOfCoverageContracts Number of Coverage Contracts oCoverageContracts Coverage Contracts oCoverageContractDescriptions Coverage Contract Descriptions oCoverageBusinessPartners Sold to/In Use by Business Partner for Coverage Contracts oNumberOfPricingContracts Number of Pricing Contracts oPricingContracts Pricing Contracts oPricingContractDescriptions Pricing Contract Descriptions oPricingBusinessPartners Sold to/In Use by Business Partner for Pricing Contracts oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return : 0 - Search routine successful <> 0 - An error occurred

SerializedItem.GetSupplierWarrantyEndDate

DLL:	tsxtcfapi

	This function is available from KB2290728 .
Syntax:	<pre> long SerializedItem.GetSupplierWarrantyEndDate(domain tcitem iItem, domain tcibd.sern iSerialNumber, ref domain tsmdm.date oEndDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function determines the Supplier Warranty End Date for the given Serialized Item. Pre: - Post: - Input: iItem Item iSerialNumber Serial Number; Mandatory Output: oEndDate The Supplier Warranty End Date in number of days from 01-01-0001. 0 if no Supplier Warranty is present. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Determining end date successful <> 0 - An error occurred

SerializedItem.GetWarrantyEndDate

DLL:	tsxtctfgapi
	This function is available from KB3613621 .
Syntax:	<pre> long SerializedItem.GetWarrantyEndDate(domain tcitem iItem, domain tcibd.sern iSerialNumber, ref domain tsmdm.date oEndDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function determines the Warranty End Date based on the Warranty Template or the Generic Warranty. The latest of these two days is used as Expiry Term. The End Date that is determined by this function is the same as the date of the field Terms Expire On on the Warranty tab in session tscfg2100m000. Pre: - Post: - Input: iItem Item iSerialNumber Serial Number; Mandatory. Output: oEndDate

	The Warranty End Date in number of days from 01-01-0001. 0 if no Supplier Warranty is present. <code>oExceptionMessage</code> The last message if any message is found. If more than one message is given, these are present in the <code>oExceptionID</code>. <code>oExceptionID</code> An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Determining end date successful. <> 0 - An error occurred.

SerializedItem.StartMultiMain

DLL:	tsxtcfgapi
Syntax:	<pre>long SerializedItem.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcitem iItem, domain tcibd.sern iSerial, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the Multi-Main session Serialized Item (tscfg2600m000). Input: <code>iStartMode</code> Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. <code>iStartFilter</code> Not Used. <code>iQueryExtend</code> A specific query to be used when zooming to this session. <code>iItem</code> Item <code>iSerial</code> Serial Number Output: <code>oExceptionMessage</code> The last message if the return value is not equal to 0. If more than one message is given, these are present in the <code>oExceptionID</code> <code>oExceptionID</code> An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

SerializedItem.StartOverview

DLL:	tsextcfgapi
	This function is available from KB2274288 .
Syntax:	<pre> long SerializedItem.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcitem iItem, domain tcibd.sern iSerial, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the overview session Serialized Items (tscfg2100m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iQueryExtend A specific query to be used when zooming to this session. iItem Item iSerial Serial Number Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

SerializedItem.Supersede

DLL:	tsextcfgapi
	This function is available from KB2281129 .
Syntax:	<pre> long SerializedItem.Supersede(domain tcitem iSourceItem fixed, domain tcibd.sern iSourceSerialNumber fixed, domain tcitem iTargetItem fixed, domain tcibd.sern iTargetSerialNumber fixed, domain tcedm.revi iRevision fixed, </pre>

	domain tcclot iLotCode fixed, domain tcseri iWarehouseOrderSeries fixed, domain tcyesno iSetPrice, domain tcpric iNewPrice, domain tcccur iCurrencyNewPrice fixed, domain tcyesno iCopySourceSerializedItemData, domain tcyesno iCopyLogisticalAndOperationalData, domain tcyesno iCopyText, domain tcyesno iCopyCounterReadings, domain tcyesno iCopyWarranty, domain tstdm.wrtpt iWarrantyType, domain tscfg.cwte iWarrantyTemplate fixed, domain tsmdm.date iWarrantyStartDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl. : This function handles the process of superseding a Serialized Item. It offers the same functionality as session Supersede Serialized Item (tscfg2240m000). Superseding is only possible when there is a Physical Breakdown Change (PBC) present for the Source Item. When no PBC exists it will be created first. To have Physical Breakdown Changes, the Configuration Management Parameter Physical Breakdown Changes must be Yes. First it is checked if superseding of the Source Item is allowed. Superseding is allowed when: - the Source Item has not been superseded already; - there are no open Service Orders, Maintenance Sales Orders, Maintenance Work Orders, Field Change Orders, Calls, Quotes, Planned Activities, Contracts, Inspections, Maintenance Notifications, Subcontract Agreements, or Claims for the Source Item; - there is no Part Maintenance Line present for the Source Item based on which the new Serialized Item will be returned to the customer, or the related warehouse issue order can be changed. When superseding is allowed the following actions are performed: - a new Serialized Item is created when it does not exist yet; when Copy Source Serialized Item Data is Yes, and Copy Logistical and Operational Data is Yes, logistical and operational data is copied as well. When Copy Logistical and Operational Data is No, the field values will be defaulted from the item master data. When Copy Text is Yes, the three Text fields are copied; - warranty data is copied when Copy Warranty indicates to do so. Otherwise warranty is controlled by the warranty settings which are passed as input arguments; - warranty is not copied when the warranty of the Source Item has been expired or terminated; - warranty is not copied when the Target Item already exists, having its own warranty;

	- when Copy Counter Readings is Yes, the Counter Readings and Reset Rules are copied between the Source Item and Target Item; - counter readings are not copied when the Target Item already exists, having its own counter readings; - when the Source Item is in a Warehouse, an Item Transfer is created and processed automatically; - the Physical Breakdown of the Source Item is copied to the Target Item; - the Serialized Item Status of the Source Item is set to Superseded, and the relation between the two items is stored in table Superseded Items (tscfg204); - finally, the existing Physical Breakdown Change for the Source Item is updated. The Item Superseded flag is set and the Target Item and Target Serial Number are updated. Pre : No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post : - Input : iSourceItem Serialized Item to supersede: Mandatory iSourceSerialNumber Serial Number of Serialized Item to supersede: Mandatory iTargetItem New Serialized Item: Mandatory iTargetSerialNumber Serial Number of New Serialized Item: Mandatory iRevision Revision of the New Serialized Item: Not mandatory iLotCode Lot Code of New Serialized Item: Not mandatory iWarehouseOrderSeries Warehouse Order Series: Not mandatory iSetPrice Set Price: Mandatory; Yes/No iNewPrice New Price: Not mandatory iCurrencyNewPrice Currency New Price: Mandatory when iSetPrice = Yes iCopySourceSerializedItemData Copy Source Serialized Item Data: Mandatory; Yes/No iCopyLogisticalAndOperationalData Copy Logistical and Operational Data: Mandatory; Yes/No iCopyText Copy Text: Mandatory; Yes/No iCopyCounterReadings Copy Counter Readings: Mandatory; Yes/No iCopyWarranty Copy Warranty: Mandatory; Yes/No iWarrantyType Warranty Type: Not mandatory iWarrantyTemplate Warranty Template: Not mandatory iWarrantyStartDate Warranty Start Date: Not mandatory

	<pre> Output : oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return : 0 - No error <> 0 - An error occurred </pre>

Public Interfaces for PhysicalBreakdown

The following functions are available:

[PhysicalBreakdown.Change](#)

[PhysicalBreakdown.CreateFromProjectBreakdownStructure](#)

[PhysicalBreakdown.CreateStructure](#)

[PhysicalBreakdown.ItemExistsInBreakdownOfParent](#)

PhysicalBreakdown.Change

DLL:	tsxtcfgapi
Syntax:	<pre> long PhysicalBreakdown.Change(domain tscfg.actn iAction, domain tcitem iItem, domain tcibd.sern iSerialNumber, domain tcitem iReplaceWithItem, domain tcibd.sern iReplaceWithSerialNumber, domain tcclot iReplaceWithLot, domain tsmdm.qmat iReplaceWithQuantity, domain tcpono iReplaceAtPosition, domain tcclot iInstallWithLot, domain tcitem iParentReplaceItem, domain tcibd.sern iParentReplaceSerialNumber, domain tscfg.cfst iTopItemStatus, domain tsbsc.clst iInstallationGroup, domain tsbsc.lino iInstallationGroupLineNumber, domain tsmdm.qmat iQuantity, domain tsmdm.utct iChangeDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl. : This function executes changes within the Physical Breakdown. The following changes are supported: </pre>

	(1) REPLACE: The replacement of Serialized Items or the replacement of Anonymous Items in a Physical Breakdown. When replacing an Item in a Physical Breakdown, the Top Item Status is changed from Active to Revision. After That the changes to the Physical Breakdown are done. Finally the Top Item Status is set back to Active again. Quantity, and Position play a key role in case of Anonymous Items replacement. The replacement can be either full or partial. (2) INSTALL: For the Serialized Item that is installed a check is done if it is present in any Installation Group. If this is the case then the Installation Group line will be set to expired. Next the Serialized Item can be installed (again) as Installation Group line. This will be done if installation group is filled. If the installation Group is empty it can be installed under any other Serialized Item within its Physical Breakdown. If the Serialized Item, the Physical Breakdown in which it is present in, or the Installation Group in which it is present is covered by an active Service Contract the Serialized Item will not be directly installed, but preparation for the installation will be done. This is done by: - Setting Status of Installation Group line to 'To be Installed' - Or setting the Status of the Physical Breakdown relation under which the Serialized Item is present to 'To be Installed'. In both cases the status time will be set to the current time. When installing an Anonymous Item in a Physical Breakdown, the Top Item Status is changed from Active to Revision. After that the changes to the Physical Breakdown are done. Finally the Top Item Status is set back to Active again. Lot, Quantity, and Position play a key role in case of Anonymous Items Installation. (3) REMOVE: For the Serialized Item that is removed a check is done if it is present in any Installation Group. If this is the case then the Installation Group line will be set to expired. If not present in an Installation Group the Serialized Item will be removed from the Physical Breakdown it is part of. If the Serialized Item, the Physical Breakdown in which it is present in, or the Installation Group in which it is present is covered by an active Service Contract the Serialized Item will not be directly removed, but preparation for the removal will be done. This is done by: - Setting status of Installation Group line to 'To be Removed', - Or setting the status of the Physical Breakdown relation under which the Serialized Item is present to 'To be Removed'. In both cases the status time will be set to the current time.

	When removing an Anonymous Item from a Physical Breakdown, the Top Item status is changed from Active to Revision. After that changes to the Physical Breakdown are done. Finally the top most Parent Item status is set back to Active again. Lot, Quantity, and Position play a key role in case of Anonymous Item removal. The removal can be either full or partial. When changing a Physical Breakdown, the status of the Top Item will be set to 'Revision' and then back to 'Active' only in case the status of the Item is 'Active' or 'Working Condition' prior to this operation. NOTE: this function cannot be used if Configuration Management Parameter 'Physical Breakdown Changes' is selected. Pre : a db.retry.point() must have been specified. Post : an abort.transaction() or commit.transaction() must be executed. Input : - iAction - Action to be done. Possible values are: - tscfg.actn.add - Install - tscfg.actn.replace - Replace - tscfg.actn.remove - Remove - iItem - The Item the action is performed at. In case of Installation: -The Item to be installed In case of Replacement: -The Item that is to be replaced In case of Removal: -The Item that is removed - iSerialNumber - The Serial Number of the item action is performed at. - iReplaceWithItem - The Item which replaces the existing Anonymous Item. - iReplaceWithSerialNumber - Serial Number of the Replacing Item. - iReplaceWithLot - Lot of the Replacing Item. - iReplaceWithQuantity - Anonymous Item Quantity. - iReplaceAtPosition - Position in the Breakdown under the Parent Serialized Item where exactly the replacement has to be made. - iInstallWithLot - Lot of Item which needs to be installed or removed or replaced. - iParentReplaceItem - The Replacement Item or the Parent Item the new component is to be installed at (depending on the action). - iParentReplaceSerialNumber - The Serial Number of the Replacement Item or the Parent Item the new component is to be installed at (depending on the action). - iTopItemStatus - The Top Item Status

	- iInstallationGroup- The Installation Group to which the Serialized Item will be linked as Top Item. - iInstallationGroupLineNumber - The Installation Group line number. - iQuantity - The Quantity of the Anonymous item being installed or removed. Output : - oExceptionMessage The last message if the return value unequals 0. If more than one message is given, these are present in the oExceptionID - oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return : 0 - No error <> 0 - An error occurred

PhysicalBreakdown.CreateFromProjectBreakdownStructure

DLL:	tsxtcfgapi
	This function is available from KB2305193 .
Syntax:	<pre>long PhysicalBreakdown.CreateFromProjectBreakdownStructure(domain tccprj iSourceProject, domain tscfg.2212.01 iSourceStructure, domain tscfg.pelt iSourceElementOrActivity, domain tsbsc.clst iTargetInstallationGroup, domain tscfg.sigr iSerialItemGroup, domain tsmdm.utct iDeliveryTime, domain tsmdm.utct iInstallationTime, domain tcyesno iCopyMaterials, domain tcyesno iGenerateDummySerials, domain tcyesno iContinueWhenAlreadyCreated, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl. : This function creates a Physical Breakdown from a Project Breakdown Structure. The same functionality is applicable as available in session Copy Project Breakdown Structure (tscfg2210m200). Pre : a db.retry.point() must have been specified. Post : an abort.transaction() or commit.transaction() must be executed. Input : - iSourceProject Project; Mandatory - iSourceStructure Origin Structure, Element or Activity; Mandatory - iSourceElementOrActivity Project Element/Activity; Mandatory - iTargetInstallationGroup Target Installation Group; Not mandatory - iSerialItemGroup Serialized Item Group; Mandatory when Configurations

	parameter Serialized Item Group Usage is Yes. iDeliveryTime Delivery Time; Mandatory when Warranty is applicable for Element/Activity, and Warranty Start is determined by Delivery. iInstallationTime Installation Time; Mandatory when Warranty is applicable for Element/Activity, and Warranty Start is determined by Installation. iCopyMaterials Copy Materials; Mandatory iGenerateDummySerials Generate Dummy Serial Numbers; Mandatory iContinueWhenAlreadyCreated Continue when Physical Breakdown already has been created for Project; Mandatory Output : ExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return : 0 - No error <> 0 - An error occurred

PhysicalBreakdown.CreateStructure

DLL:	tsxtcfgapi This function is available from KB3567711.
Syntax:	<pre>long PhysicalBreakdown.CreateStructure(long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl.: This function creates a Physical Breakdown Structure. It offers the same functionality as session Create Physical Breakdown Structure (tscfg2210m000). The Help of this session can be used as reference. Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Call ProcessingOptionSet.Create() to obtain iProcessingOptionSet. Post: Delete the option set by calling ProcessingOptionSet.Delete(). Input: iProcessingOptionSet Processing Option Set: a processing option set number referring to a processing option set containing at least one valid option.

Mandatory.		
NAME	TYPE	DEFAULT
=====		
Source	domain tscfg2210.src	item.breakdown
Selects the source of the new physical breakdown. Allowed values are: tscfg2210.src.item.breakdown tscfg2210.src.as.built tscfg2210.src.bom		
UseDeliveries	domain tcyesno	no
Use this option to create a physical breakdown from Sales (after sales). The end item (of the sales order) P-BOM is copied to a physical breakdown. If UseDeliveries is Yes, the following options are applicable: FromInstallationGroup ToInstallationGroup FromCreationTime ToCreationTime FromItem ToItem FromSerialNumber ToSerialNumber		
SkipParents	domain tcyesno	yes
The creation of the new physical breakdown relations between the existing parent serialized items and the child items that are part of an As-built Structure, is skipped.		
Target	domain tscfg2210.link	new.breakdown
Determines the target of the created Physical Breakdown Structure. Allowed values are: tscfg2210.link.install.group tscfg2210.link.breakdown tscfg2210.link.new.breakdown		
LinkToInstallationGroup	domain tsbsc.clst	""
The target Installation Group; applicable when Target is Installation Group.		
LinkToItem	domain tcitem	""
The target Item; applicable when Target is Breakdown.		
LinkToSerialNumber	domain tcibd.sern	""

	The target Serial Number; applicable when Target is Breakdown. FromInstallationGroup domain tsbsc.clst "" From Installation Group; applicable when UseDeliveries is Yes. ToInstallationGroup domain tsbsc.clst "" To Installation Group; applicable when UseDeliveries is Yes. FromCreationTime domain tsmdm.utct 0 From Creation Time; applicable when UseDeliveries is Yes. ToCreationTime domain tsmdm.utct utc.num() To Creation Time; applicable when UseDeliveries is Yes. FromItem domain tcitem "" From Item; applicable when UseDeliveries is Yes. ToItem domain tcitem "" To Item; applicable when UseDeliveries is Yes. FromSerialNumber domain tcibd.sern "" From Serial Number; applicable when UseDeliveries is Yes. ToSerialNumber domain tcibd.sern "" To Serial Number; applicable when UseDeliveries is Yes. DefaultSerializedItemGroup domain tscfg.sigr "" The Serialized Item Group to which the serialized items in the newly-created physical breakdown will belong. DefaultServiceOffice domain tccwoc "" The service department that will carry out service and maintenance activities, on the serialized items in the newly-generated physical breakdown. DefaultOwner

	domain tcccom.bpid "" The Owner to which the serialized items in the newly-created physical breakdown will belong.
SerialStatus	domain tscfg.cfst activ The default Serialized Item Status. This field is only applicable when Configuration Status Usage is checked in the Configuration Management Parameters. Allowed values are: tscfg.cfst.stup tscfg.cfst.activ
DeliveryTime	domain tsmdm.utct 0 The date and time that the item is delivered by the service organization.
InstallationTime	domain tsmdm.utct 0 The date and time when the new serialized item is installed.
CheckEffectivity	domain tcyesno yes Check Effectivity
EffectiveTime	domain tsmdm.utct utc.num() The date on which the service item's breakdown terms are checked for validity.
GenerateDummySerials	domain tcyesno no When GenerateDummySerials is Yes, and the physical breakdown is created from an Item Breakdown, or a Bill of Material, the serial numbers generated for the new serialized items will be marked as dummy serial numbers. These can be replaced later by their real serial numbers.
GenerateServiceItemData	domain tcyesno yes When GenerateServiceItemData is Yes, and the physical breakdown is created from an Item Breakdown, or a Bill of Material, Service Item Data is generated for the new serialized items when they are not present yet.
FallBackOnBillOfMaterial	domain tcyesno yes When the physical breakdown is created from an Item Breakdown, the Bill of Material is used when there is no Item Breakdown.

	CompletedProductionOrdersOnly domain tcyesno no If CompletedProductionOrdersOnly is Yes, only completed production orders are processed to create the physical breakdown structure. CreateOtherBreakdownWhenBreakdownPresent domain tcyesno no When there is already a Physical Breakdown present where the Parent Item/Serial is equal to the LinkToItem / LinkToSerialNumber, another Breakdown is created when CreateOtherBreakdownWhenBreakdownPresent is Yes. Otherwise, nothing is done. Output : ExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return : 0 - No error; however, error messages can have been set. <> 0 - An error occurred

PhysicalBreakdown.ItemExistsInBreakdownOfParent

DLL:	tsxtcfgapi
	This function is available from KB2331323 .
Syntax:	<pre>long PhysicalBreakdown.ItemExistsInBreakdownOfParent (domain tcitem iItem, domain tcibd.sern iSerialNumber, domain tcitem iParentItem, domain tcibd.sern iParentSerialNumber, ref boolean oExists, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function is used to determine if the given child (serialized) item is present in the physical breakdown of the given parent serialized item directly or indirectly. If a child serial number is provided, the physical breakdown under the parent serialized item will only be searched for that specific child serialized item. If no child serial number is provided, the physical breakdown under the parent serialized item will be searched for any child item with the given item code. Note that only installed statuses are taken into account. Pre: Any provided serialized item must exist. Input: iItem Child Item; Mandatory if iSerialNumber is empty iSerialNumber

	Child Serial Number; Mandatory if iItem is empty iParentItem Parent Item; Not mandatory iParentSerialNumber Parent Serial Number; Mandatory Output: oExists True: The given child (serialized) item exists in the physical breakdown tree under the given parent serialized item. False: The given child (serialized) item does not exist in the physical breakdown tree under the given parent serialized item. oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return : 0: No error <> 0: An error occurred

Public Interfaces for ServiceInspections

The following functions are available:

[ServiceInspections.StartOverview](#)

ServiceInspections.StartOverview

DLL:	tsxtcfgapi
	This function is available from KB2293111 .
Syntax:	<pre>long ServiceInspections.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcitem iItem fixed, domain tcibd.sern iSerial fixed, domain tcorno iOrder, domain tcpono iActivity, domain tcorno iClaim, domain tcpono iClaimLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl. : This function starts the overview session Inspections

```

(tscfg3100m000).
Pre      : -
Post     : -
Input    : iStartMode: Not used; Not mandatory
           Session is always started MODELESS. That means that
           parent and child are parallel sessions that can be
           manipulated simultaneously.
iStartFilter: Not mandatory
           iStartFilter determines the View fields when the
           session is started.
Possible values      | View fields
-----|-----
""                  | No view fields
"SerializedItem"    | Item, Serial
"ServiceOrderActivity" | Service Order, Activity
"WorkOrder"         | Work Order, Activity (0)
"WorkOrderActivity" | Work Order, Activity
"CustomerClaim"     | Customer Claim, Line (0)
"CustomerClaimLine" | Customer Claim, Line
"SupplierClaim"     | Supplier Claim, Line (0)
"SupplierClaimLine" | Supplier Claim, Line

iQueryExtend: Not used; Not mandatory
           A specific query to be used when zooming to this
           session.
iItem
           Item. Mandatory when iStartFilter is SerializedItem.
iSerial
           Serial Number. Mandatory when iStartFilter is
           SerializedItem.
iOrder
           Order Number. Mandatory when iStartFilter is
           ServiceOrderActivity, WorkOrder, or WorkOrderActivity.
iActivity
           Activity. Mandatory when iStartFilter is
           ServiceOrderActivity, or WorkOrderActivity.
iClaim
           Claim. Mandatory when iStartFilter is
           CustomerClaim, CustomerClaimLine, SupplierClaim, or
           SupplierClaimLine.
iClaimLine
           Claim Line. Mandatory when iStartFilter is
           CustomerClaimLine, or SupplierClaimLine.
Output   : oExceptionMessage
           The last message if the return value is not equal to 0.
           If more than one message is given, these are present
           in the oExceptionID
           oExceptionID
           An ID that refers to all error information. Use the
           functions in Exception to get all relevant information.
Return   : 0           Session started
           <> 0        An error occurred

```

Public Interfaces for Call

The following functions are available:

[Call.TransferToCustomerClaim](#)

[Call.TransferToPlannedActivity](#)

[Call.TransferToServiceOrder](#)

[Call.TransferToServiceOrderV2](#)

[Call.TransferToServiceQuote](#)

[Call.TransferToWorkOrder](#)

Call.TransferToCustomerClaim

DLL:	ttextclmapi This function is available from KB3542116 .
Syntax:	<pre> long Call.TransferToCustomerClaim(domain tcorno iCall, ref domain tcyesno oCallIsBlocked, ref domain tcorno oCustomerClaim, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function handles the transfer of a call to a customer claim. Before transferring the call, based on the blocking settings defined in the Call Management parameters, it is checked if the call must be blocked. When the call is set to blocked, the call cannot be transferred and output argument oCallIsBlocked will be set to Yes. Pre: None. Post: This function sets a retry-point and will commit and/or abort the transaction. Input: iCall Call Number: Mandatory Output: oCallIsBlocked Indicates if the call is set to blocked. oCustomerClaim The customer claim number to which the call is transferred. oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 : Transfer successful or Call is Blocked

	<> 0 : Error occurred

Call.TransferToPlannedActivity

DLL:	tsxtclmapi
	This function is available from KB3614353 .
Syntax:	<pre> long Call.TransferToPlannedActivity(domain tcorno iCall, ref domain tcyesno oCallIsBlocked, ref domain tcorno oPlannedActivity, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this function to transfer a Call to a new Planned Activity. Note that the Master Routing or the Required Activity must be set on the Call. Before transferring the call, based on the blocking settings defined in the Call Management parameters, it is checked if the call must be blocked. When the call is set to blocked, the call cannot be transferred and output argument oCallIsBlocked will be set to Yes. Pre: - Post: This function sets a retry-point and will commit and/or abort the transaction. Input: iCall Call Number: Mandatory ===== Output: oCallIsBlocked Indicates if the Call is set to blocked. If the Call is blocked, the Call will not be transferred. oPlannedActivity The Planned Activity that the Call was transferred to. oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 : Transfer successful or Call is Blocked <> 0 : Error occurred Return: long

Call.TransferToServiceOrder

DLL:	tsxtclmapi This function is available from KB2270102 .
Syntax:	<pre> long Call.TransferToServiceOrder(domain tcorno iCall, domain tcyesno iContinueWhenStartOrFinishTimesNotSet, domain tcyesno iContinueNoDurationSetForEmergencyCall, domain tcyesno iContinueNoEngineerSetForEmergencyCall, domain tcyesno iGenerateFCOForSerializedItem, domain tcyesno iContinueWhenFCOWillBeUnlinked, domain tcyesno iContinueWhenPlannedTimesCannotBeDetermined, domain tcyesno iContinueWhenPlannedFinishAfterAgreedFinishTime, domain tcyesno iCreateActivityLineUnderPresentServiceOrder, domain tcyesno iCreateNewOrderWhenMultipleOpenOrdersPresent, domain tcyesno iSetStatusToPlanned, ref domain tcyesno oCallIsBlocked, ref domain tcorno oServiceOrder, ref domain tsmdm.acln oActivityLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function handles the transfer of a call to a service order. Based on the provided input arguments this transfer process can be controlled. It can be indicated to continue the transfer process or to stop. These are situations for which in LN UI a question is asked whether the transfer process should continue or stop. When an open service order exists for the same sold-to business partner, invoice-to business partner and service office, and iCreateActivityLineUnderPresentServiceOrder is set to 'yes', a new service order activity line will be created for this service order. When iCreateActivityLineUnderPresentServiceOrder is set to 'no', a new service order and activity line are generated. When multiple open service orders are present to which the call can be transferred, it is not possible to transfer the call to one of these service orders as it is not possible to make a selection. When iCreateNewOrderWhenMultipleOpenOrdersPresent is set to 'yes' a new service order and activity line will be generated. Note that when Call Management parameter 'Matching Service Order Numbers' is set to 'yes', always a new service order and activity line are generated. Before transferring the call, based on the blocking settings defined in the Call Management parameters, it is checked if the call must be blocked. When the call is set to blocked, the call cannot be transferred and output argument oCallIsBlocked will be set to 'yes'. Pre: None. Post: This function sets a retry-point and will commit and/or abort the transaction. Input: iCall Call Number. (mandatory) iContinueWhenStartOrFinishTimesNotSet

	Continue transfer when the Latest Solution Start Time is not set on the call or the Latest Solution Finish Time is later than the Latest Solution Start Time? (mandatory Yes/No) iContinueNoDurationSetForEmergencyCall Continue transfer when the call is an emergency call and no duration is set on the call and Call parameters? (mandatory Yes/No) iContinueNoEngineerSetForEmergencyCall Continue transfer when the call is an emergency call and no service engineer is set on the call? (mandatory Yes/No) iGenerateFCOForSerializedItem When a Field Change Order is linked to the call and a serialized item is set on the call and iGenerateFCOForSerializedItem is set to 'yes', a Field Change Order Line will be generated instead of transferring the call to a service order. (mandatory Yes/No) iContinueWhenFCOWillBeUnlinked Continue transfer when a Field Change Order is linked to the call and an anonymous item is set on the call as this transfer will remove the link between the FCO and the call. (mandatory Yes/No) iContinueWhenPlannedTimesCannotBeDetermined Continue transfer when the service order planned start or planned finish time could not be determined? (mandatory Yes/No) iContinueWhenPlannedFinishAfterAgreedFinishTime Continue transfer when the service order planned finish time will exceed the agreed finish time of the call? (mandatory Yes/No) iCreateActivityLineUnderPresentServiceOrder When there is one open service order present for the same sold-to business partner, invoice-to business partner and service office, and iCreateActivityLineUnderPresentServiceOrder is set to 'yes', a new service order activity line will be created for this service order. When iCreateActivityLineUnderPresentServiceOrder is set to 'no', a new service order and activity line are generated. (mandatory Yes/No) iCreateNewOrderWhenMultipleOpenOrdersPresent When multiple open service orders are present to which the call can be transferred, it is not possible to transfer the call to one of these service orders as it is not possible to make a selection. When iCreateNewOrderWhenMultipleOpenOrdersPresent is set to 'yes' a new service order and activity line will be generated. When iCreateNewOrderWhenMultipleOpenOrdersPresent is set to 'no', the call will not be transferred to service order. (mandatory Yes/No) iSetStatusToPlanned Indicates if the new service order/activity line must be set to status Planned. (mandatory Yes/No) Output: oCallIsBlocked Indicates if the call is set to blocked.

	oServiceOrder The new or existings service order number to which the call is tranferred. oActivityLine The new service order activity line number to which the call is transferred. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - transfer succesfull - Error during transfer occurred

Call.TransferToServiceOrderV2

DLL:	tsxtclmapi This function is available from KB2286782.
Syntax:	<pre>long Call.TransferToServiceOrderV2(domain tcorno iCall, domain tcorno iServiceOrder, domain tcyesno iContinueWhenStartOrFinishTimesNotSet, domain tcyesno iContinueNoDurationSetForEmergencyCall, domain tcyesno iContinueNoEngineerSetForEmergencyCall, domain tcyesno iGenerateFCOForSerializedItem, domain tcyesno iContinueWhenFCOWillBeUnlinked, domain tcyesno iContinueWhenPlannedTimesCannotBeDetermined, domain tcyesno iContinueWhenPlannedFinishAfterAgreedFinishTime, domain tcyesno iCreateActivityLineUnderPresentServiceOrder, domain tcyesno iCreateNewOrderWhenMultipleOpenOrdersPresent, domain tcyesno iSetStatusToPlanned, ref domain tcyesno oCallIsBlocked, ref domain tcorno oServiceOrder, ref domain tsmdm.acln oActivityLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function handles the transfer of a call to a service order. Based on the provided input arguments this transfer process can be controlled. It can be indicated to continue the transfer process or to stop. These are situations for which in LN UI a question is asked whether the transfer process should continue or stop. When one open service order exists for the same sold-to business partner, invoice-to business partner and service office, and iCreateActivityLineUnderPresentServiceOrder is set to Yes, a new service order activity line will be created for this service order. In this case iServiceOrder is ignored. When iCreateActivityLineUnderPresentServiceOrder is set to No, a new service order and activity line are generated.

	When multiple open service orders are present to which the call can be transferred, it is only possible to transfer the call when iServiceOrder is supplied and is one of the open service orders. When iServiceOrder is not supplied it is not possible to transfer the call to one of these service orders as it is not possible to make a selection. When iCreateNewOrderWhenMultipleOpenOrdersPresent is set to Yes a new service order and activity line will be generated. Note that when Call Management parameter 'Matching Service Order Numbers' is set to Yes, always a new service order and activity line are generated. Before transferring the call, based on the blocking settings defined in the Call Management parameters, it is checked if the call must be blocked. When the call is set to blocked, the call cannot be transferred and output argument oCallIsBlocked will be set to Yes. Pre: None. Post: This function sets a retry-point and will commit and/or abort the transaction. Input: iCall Call Number: Mandatory iServiceOrder Service Order: Not mandatory iContinueWhenStartOrFinishTimesNotSet Continue transfer when the Latest Solution Start Time is not set on the call or the Latest Solution Finish Time is later than the Latest Solution Start Time. (mandatory Yes/No) iContinueNoDurationSetForEmergencyCall Continue transfer when the call is an emergency call and no duration is set on the call and Call parameters. (mandatory Yes/No) iContinueNoEngineerSetForEmergencyCall Continue transfer when the call is an emergency call and no service engineer is set on the call. (mandatory Yes/No) iGenerateFCOForSerializedItem When a Field Change Order is linked to the call and a serialized item is set on the call and iGenerateFCOForSerializedItem is set to Yes, a Field Change Order Line will be generated instead of transferring the call to a service order. (mandatory Yes/No) iContinueWhenFCOWillBeUnlinked Continue transfer when a Field Change Order is linked to the call and an anonymous item is set on the call as this transfer will remove the link between the FCO and the call. (mandatory Yes/No) iContinueWhenPlannedTimesCannotBeDetermined Continue transfer when the service order planned start or planned finish time could not be determined. (mandatory Yes/No) iContinueWhenPlannedFinishAfterAgreedFinishTime Continue transfer when the service order planned finish time will exceed the agreed finish time of the call. (mandatory Yes/No)

	iCreateActivityLineUnderPresentServiceOrder When there is one open service order present for the same sold-to business partner, invoice-to business partner and service office, and iCreateActivityLineUnderPresentServiceOrder is set to Yes, a new service order activity line will be created for this service order. When iCreateActivityLineUnderPresentServiceOrder is set to No, a new service order and activity line are generated. (mandatory Yes/No) iCreateNewOrderWhenMultipleOpenOrdersPresent When multiple open service orders are present, the next situations are distinguished: <table><tr><th>Value</th><th>iServiceOrder</th><th>Call transferred to</th></tr><tr><td>-----</td><td>-----</td><td>-----</td></tr><tr><td>Yes</td><td>""</td><td>new service order</td></tr><tr><td>Yes</td><td>valid</td><td>iServiceOrder</td></tr><tr><td>Yes</td><td>invalid</td><td>new service order</td></tr><tr><td>No</td><td>""</td><td>not tranferred --> error</td></tr><tr><td>No</td><td>valid</td><td>iServiceOrder</td></tr><tr><td>No</td><td>invalid</td><td>not tranferred --> error</td></tr></table> (mandatory Yes/No) iSetStatusToPlanned Indicates if the new service order/activity line must be set to status Planned. (mandatory Yes/No) Output: oCallIsBlocked Indicates if the call is set to blocked. oServiceOrder The new or existing service order number to which the call is transferred. oActivityLine The new service order activity line number to which the call is transferred. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Transfer successful <> 0 - Error during transfer occurred	Value	iServiceOrder	Call transferred to	-----	-----	-----	Yes	""	new service order	Yes	valid	iServiceOrder	Yes	invalid	new service order	No	""	not tranferred --> error	No	valid	iServiceOrder	No	invalid	not tranferred --> error
Value	iServiceOrder	Call transferred to																							
-----	-----	-----																							
Yes	""	new service order																							
Yes	valid	iServiceOrder																							
Yes	invalid	new service order																							
No	""	not tranferred --> error																							
No	valid	iServiceOrder																							
No	invalid	not tranferred --> error																							

Call.TransferToServiceQuote

DLL:	tsxtclmapi
	This function is available from KB3614350 .
Syntax:	long Call.TransferToServiceQuote(domain tcorno iCall,

	refdomain tcyesno oCallIsBlocked, refdomain tcorno oQuote, refdomain tcpono oQuoteRevision, refdomain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: Use this function to: - Transfer a Call to a Quote for a Service Order - Transfer a Call to a Quote for Part Maintenance This function handles the transfer of a call to a service quote. Before transferring the call, based on the blocking settings defined in the Call Management parameters, it is checked if the call must be blocked. When the call is set to blocked, the call cannot be transferred and output argument oCallIsBlocked will be set to Yes. Pre: Call ProcessingOptionSet.Create() to obtain iProcessingOptionSet. Post: Delete the option set by calling ProcessingOptionSet.Delete() This function sets a retry-point and will commit and/or abort the transaction. Input: iCall Call Number: Mandatory iProcessingOptionSet Processing Option Set: Mandatory, a processing option set number, referring to a processing option set containing at least one valid option. NAME TYPE DEFAULT ===== QuoteType long 1 This defines to which type of Service Quote the Call should be transferred to. When no correct value is provided, the default value is 1. Allowed values: 1 : Quote for a Service Order 2 : Quote for Part Maintenance Output: oCallIsBlocked Indicates if the call is set to blocked. oQuote The quote number to which the call is transferred. oQuoteRevision The quote revision to which the call is transferred. oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 : Transfer successful or Call is Blocked <> 0 : Error occurred

Call.TransferToWorkOrder

DLL:	tsextclmapi This function is available from KB2286841 .
Syntax:	<pre> long Call.TransferToWorkOrder(domain tcorno iCall, domain tcorno iWorkOrder, domain tcyesno iContinueWhenStartOrFinishTimesNotSet, domain tcyesno iRemoveLinkToFCO, domain tcyesno iCreateActivityLineUnderPresentWorkOrder, domain tcyesno iCreateNewOrderWhenMultipleOpenOrdersPresent, ref domain tcyesno oCallIsBlocked, ref domain tcorno oWorkOrder, ref domain tsmdm.acln oActivityLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function handles the transfer of a call to a work order. Based on the provided input arguments this transfer process can be controlled. It can be indicated to continue the transfer process or to stop. These are situations for which in LN UI a question is asked whether the transfer process should continue or stop. When one open work order exists for the same Item/Serial and same Project/Element/Activity, and iCreateActivityLineUnderPresentWorkOrder is set to Yes, a new work order activity line will be created for this work order. In this case iWorkOrder is ignored. When iCreateActivityLineUnderPresentWorkOrder is set to No, a new work order and activity line is generated. When multiple open work orders are present to which the call can be transferred, it is only possible to transfer the call when iWorkOrder is supplied, and is one of the open work orders. When iWorkOrder is not supplied it is not possible to transfer the call to one of these work orders as it is not possible to make a selection. When iCreateNewOrderWhenMultipleOpenOrdersPresent is set to Yes a new work order and activity line will be generated. Note that when Call Management parameter 'Allow Transfer of Calls to same Order in Depot Repair' is No, always a new work order and activity line is generated. Before transferring the call, based on the blocking settings defined in the Call Management parameters, it is checked if the call must be blocked. When the call is set to blocked, the call cannot be transferred and output argument oCallIsBlocked will be set to Yes. Pre: None. Post: This function sets a retry-point and will commit and/or abort the transaction. Input: iCall Call Number: Mandatory

```

iWorkOrder
    Work Order: Not mandatory
iContinueWhenStartOrFinishTimesNotSet
    Continue transfer when the Latest Solution Start Time
    is not set on the call or the Latest Solution Finish
    Time is later than the Latest Solution Start Time.
    (mandatory Yes/No)
iRemoveLinkToFCO
    When Call is linked to a Field Change Order, and
    iRemoveLinkToFCO is No, the Call will not be
    transferred. Otherwise, the Call is transferred, and
    the Field Change Order will be emptied.
    (mandatory Yes/No)
iCreateActivityLineUnderPresentWorkOrder
    When there is one open work order present for the
    same Item/Serial, and same Project/Element/Activity,
    and iCreateActivityLineUnderPresentWorkOrder is set to
    Yes, a new work order activity line will be
    created for this work order.
    When iCreateActivityLineUnderPresentWorkOrder is set
    to No, a new work order and activity line is
    generated.
    (mandatory Yes/No)
iCreateNewOrderWhenMultipleOpenOrdersPresent
    When multiple open work orders are present, the next
    situations are distinguished:

    Value   | iWorkOrder   | Call transferred to
    -----|-----|-----
    Yes     | ignored      | new work order
    No      | ""           | not tranferred --> error
    No      | valid        | iWorkOrder
    No      | invalid      | not tranferred --> error

    (mandatory Yes/No)
Output: oCallIsBlocked
    Indicates if the call is set to blocked.
oWorkOrder
    The new or existing work order number to which the
    call is transferred.
oActivityLine
    The new work order activity line number to which the
    call is transferred.
oExceptionMessage
    The last message if the return value is not equal to 0.
    If more than one message is given, these are present
    in the oExceptionID.
oExceptionID
    An ID that refers to all error information. Use the
    functions in Exception to get all relevant information.
Return: 0
    - Transfer successful
    <> 0  - Error during transfer occurred

```

Public Interfaces for ServiceContract

The following functions are available:

[ServiceContract.Activate](#)

[ServiceContract.ActivateV2](#)

[ServiceContract.Create](#)

[ServiceContract.GenerateConfigurationLines](#)

[ServiceContract.GetTotalSalesAmount](#)

[ServiceContract.StartMultiMain](#)

[ServiceContract.UnpackObjectReference](#)

ServiceContract.Activate

DLL:	tsxtctmapi This function is available from KB2285960 .
Syntax:	<pre> long ServiceContract.Activate(domain tcorno iServiceContract, domain tsctm.cchn iServiceContractChange, domain tcyesno iContinueWhenNoSalesAmountOnLines, domain tcyesno iContinueWhenInvoicingByChanges, domain tcyesno iContinueActivatingChangeWithExpiredContractLines, domain tsctm.upt.cntr iActionOnOrdersCallsWhenExpired, domain tsctm.link iLinkOrApplyCoverageOrIgnoreWhenOpenOrdersOrCalls, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function makes the passed Service Contract, and/or Service Contract Change, active. First general checks are executed whether activation is allowed or not. Next, for contract renewal, or an incidental change, possible prepared changes in the Installation Group and Physical Breakdowns of the configuration lines are processed. Then the Contract Change is made active. Finally, the table Totals (tsctm100) is filled based on the aggregation process, and the contract installments are handled. Based on the provided input arguments this process can be controlled. It can be indicated to continue the activation

	process or to stop. These are situations for which in LN UI questions are asked whether the process should continue or stop. When only iServiceContract is passed, the Service Contract, and the Service Contract Change of type Original Contract, is activated. When iServiceContract, and iServiceContractChange are passed, and the Service Contract Change is of type Original Contract, both Service Contract and Service Contract Change are activated. When the Service Contract Change is not of type Original Contract, then the Service Contract Change is activated. Pre: - Post: This function sets a retry-point and will commit and/or abort the transaction. Input: iServiceContract Service Contract: Mandatory iServiceContractChange Service Contract Change. When the Service Contract must be activated, pass zero (0). Otherwise, pass a valid Service Contract Change number. (Not mandatory) iContinueWhenNoSalesAmountOnLines Continue activation when sales amount has not been filled on all contract configuration lines that are used for contract coverage. (mandatory Yes/No) iContinueWhenInvoicingByChanges Continue when there are Calls, Service Orders and/or Maintenance Sales Orders present for configurations covered by this contract with Invoicing By Project. Invoicing By will be changed to Service. (mandatory Yes/No) iContinueActivatingChangeWithExpiredContractLines Continue when there are Calls, Service Orders and/or Maintenance Sales Orders present for configurations that are expired by this contract change. (mandatory Yes/No) iActionOnOrdersCallsWhenExpired When there are open Calls, Service Orders and/or Maintenance Sales Orders and the effective or expiry date of the contract line will result in an invalid contract on the order/call, then there are the following options: - Unlink Contract Unlinking is done by updating the orders/calls by setting the coverage time to 0. The contract will be unlinked and coverage is recalculated for the cost lines. - Relink Contract Relinking is done by setting the coverage time on the orders/calls to the contract line expiry date, which will redetermine the applicable service contract. (mandatory unlink/relink) iLinkOrApplyCoverageOrIgnoreWhenOpenOrdersOrCalls When there are Calls, Service Orders and/or Maintenance Sales Orders present for configurations covered by this contract, then there are the following options:

	- Establish Link For each contract configuration line which is present on a Call, Service Order (Activities), or Maintenance Sales Order Line, the Service Contract is filled. - Apply Coverage Contract coverage is applied to orders and calls present for contract configurations. First the Service Contract is updated on each of these Calls and Orders. Next the contract coverage calculation is performed. - Ignore Nothing is updated on open Calls, Service Order (Activities), or Maintenance Sales Order Lines. (mandatory Link/Cover/Ignore) Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Activation successful <> 0 - Error during Activation occurred

ServiceContract.ActivateV2

DLL:	tsextctmapi This function is available from KB2309534 .
Syntax:	<pre> long ServiceContract.ActivateV2(domain tcorno iServiceContract, domain tsctm.cchn iServiceContractChange, domain tcyesno iContinueWhenNoSalesAmountOnLines, domain tcyesno iContinueWhenInvoicingByChanges, domain tcyesno iContinueActivatingChangeWithExpiredContractLines, domain tsctm.upt.cntr iActionOnOrdersCallsWhenExpired, domain tsctm.link iLinkOrApplyCoverageOrIgnoreWhenOpenOrdersOrCalls, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function makes the passed Service Contract, and/or Service Contract Change, active. Note: - Public Interface ServiceContract.Activate has transaction management. - Public Interface ServiceContract.ActivateV2 does not have transaction management. First general checks are executed whether activation is allowed or not.

	Next, for contract renewal, or an incidental change, possible prepared changes in the Installation Group and Physical Breakdowns of the configuration lines are processed. Then the Contract Change is made active. Finally, the table Totals (tsctm100) is filled based on the aggregation process, and the contract installments are handled. Based on the provided input arguments this process can be controlled. It can be indicated to continue the activation process or to stop. These are situations for which in LN UI questions are asked whether the process should continue or stop. When only iServiceContract is passed, the Service Contract, and the Service Contract Change of type Original Contract, is activated. When iServiceContract, and iServiceContractChange are passed, and the Service Contract Change is of type Original Contract, both Service Contract and Service Contract Change are activated. When the Service Contract Change is not of type Original Contract, then the Service Contract Change is activated. Pre: a db.retry.point() must have been specified. Post: an abort.transaction() or commit.transaction() must be executed. Input: iServiceContract Service Contract: Mandatory iServiceContractChange Service Contract Change. When the Service Contract must be activated, pass zero (0). Otherwise, pass a valid Service Contract Change number. (Not mandatory) iContinueWhenNoSalesAmountOnLines Continue activation when sales amount has not been filled on all contract configuration lines that are used for contract coverage. (mandatory Yes/No) iContinueWhenInvoicingByChanges Continue when there are Calls, Service Orders and/or Maintenance Sales Orders present for configurations covered by this contract with Invoicing By Project. Invoicing By will be changed to Service. (mandatory Yes/No) iContinueActivatingChangeWithExpiredContractLines Continue when there are Calls, Service Orders and/or Maintenance Sales Orders present for configurations that are expired by this contract change. (mandatory Yes/No) iActionOnOrdersCallsWhenExpired When there are open Calls, Service Orders and/or Maintenance Sales Orders and the effective or expiry date of the contract line will result in an invalid contract on the order/call, then there are the following options: - Unlink Contract Unlinking is done by updating the orders/calls by setting the coverage time to 0. The contract will be unlinked and coverage is recalculated

	for the cost lines. - Relink Contract Relinking is done by setting the coverage time on the orders/calls to the contract line expiry date, which will redetermine the applicable service contract. (mandatory unlink/relink) iLinkOrApplyCoverageOrIgnoreWhenOpenOrdersOrCalls When there are Calls, Service Orders and/or Maintenance Sales Orders present for configurations covered by this contract, then there are the following options: - Establish Link For each contract configuration line which is present on a Call, Service Order (Activities), or Maintenance Sales Order Line, the Service Contract is filled. - Apply Coverage Contract coverage is applied to orders and calls present for contract configurations. First the Service Contract is updated on each of these Calls and Orders. Next the contract coverage calculation is performed. - Ignore Nothing is updated on open Calls, Service Order (Activities), or Maintenance Sales Order Lines. (mandatory Link/Cover/Ignore) Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Activation successful <> 0 - Error during Activation occurred

ServiceContract.Create

DLL:	tsctmapi
	This function is available from KB3566270 .
Syntax:	<pre>long ServiceContract.Create(long iProcessingOptionSet, ref domain tcorno oServiceContract fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl.: This function creates a Service Contract. It offers the same functionality as session Service Contract (tsctm3100m000). Normal defaulting will be applied for attributes if not specified as input arguments. The Help of this session can be used as reference.

Pre: A db.retry.point() must have been specified.

Call ProcessingOptionSet.Create() to obtain
iProcessingOptionSet.

Post: An abort.transaction() or commit.transaction() must be
executed.

Delete the option set by calling ProcessingOptionSet.Delete().

Input: iProcessingOptionSet

Processing Option Set: a processing option set number
referring to a processing option set containing at
least one valid option.
Mandatory.

NAME	TYPE		
=====			
Series	domain	tcorno	
Description	domain	tsmdm.dsca	mandatory
ContractType	domain	tsctm.ctpc	
ContractEffectiveDate	domain	tsmdm.date	
ServiceOffice	domain	tccwoc	
InternalSalesRepresentative	domain	tcemno	
SalesPriceList	domain	tccplt	
SoldToBusinessPartner	domain	tccom.bpid	mandatory
SoldToAddress	domain	tccom.cadr	
SoldToContact	domain	tccom.ccnt	
FirstInvoiceReference	domain	tcrefa	
SecondInvoiceReference	domain	tcrefb	
LineOfBusiness	domain	tccbrn	
ContractDuration	domain	tsmdm.numb	
ContractDurationPeriodUnit	domain	tsmdm.peru	
PricingMethod	domain	tsctm.prmt	
PercentageOfSalesValue	domain	tsmdm.perc	
OvertimeAllowed	domain	tcyesno	
ContractRenewal	domain	tcyesno	
RenewalPeriod	domain	tsmdm.numb	
RenewalPeriodPeriodUnit	domain	tsmdm.peru	
BpPricesAndDiscounts			

	SalesArea	domain	tccom.bpid
	Indexation	domain	tccreg
	IndexationStartDate	domain	tcyesno
	IndexationInterval	domain	tsmdm.date
	IndexationIntervalPeriodUnit	domain	tsmdm.numb
	IndexationTemplate	domain	tsmdm.peru
	IncidentalChanges	domain	tsctm.cind
	PenaltyAmount	domain	tcyesno
	InstallmentTemplatePerConfiguration	domain	tcamnt
	InstallmentTemplate	domain	tcyesno
	LatePaymentSurcharge	domain	tsctm.ctin
	PaymentTerms	domain	tcccrs
	Currency	domain	tccpay
	TaxCountry	domain	tcccur
	TaxClassification	domain	tcctty
	TaxCode	domain	tctax.bpcl
	BusinessPartnerTaxCountry	domain	tccvat
	TaxExempt	domain	tcctty
	TaxExemptionCertificate	domain	tcyesno
	TaxExemptionReason	domain	tcfovn
	InvoiceToBusinessPartner	domain	tccdis
	InvoiceToAddress	domain	tccom.bpid
	InvoiceToContact	domain	tccom.cadr
	PayByBusinessPartner	domain	tccom.ccnt
	PayByAddress	domain	tccom.bpid
	PayByContact	domain	tccom.cadr
	SalesType	domain	tccom.ccnt
	RevenueRecognitionBasedOn	domain	tcpsty
	RecognizeRevenuePerContractRenewal	domain	tsctm.rvmd
	RecognizeRevenuePerConfiguration	domain	tcyesno
	Provision	domain	tcyesno

	<pre> Project domain tsmdm.perc domain tccprj Eelement domain tccspa Activity domain tccact ServiceContractText domain tsmdm.text HeaderText domain tsmdm.text FooterText domain tsmdm.text InstallmentText domain tsmdm.text domain tsmdm.text </pre>
	Output: oServiceContract The ID of the created service contract. oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - No error; however, error messages can have been set. <> 0 - An error occurred

ServiceContract.GenerateConfigurationLines

DLL:	tsctmapi
	This function is available from KB2289318 .
Syntax:	<pre> long ServiceContract.GenerateConfigurationLines(domain tsctm.termappl iGenerateFor, domain tcorno iServiceContractOrContractQuote, domain tsctm.cchn iServiceContractChange, domain tsbsc.clst iInstallationGroup, domain tcitem iFromItem, domain tcitem iToItem, domain tcibd.sern iFromSerialNumber, domain tcibd.sern iToSerialNumber, domain tsctm.cgrou iRunIdentification, domain tsctm.ctin iInstallmentTemplate, domain tsctm.ccte iContractTemplate, domain tcyesno iApplyItemPriceListForContractTemplate, domain tcyesno iPickBestFitTemplateFromItemPriceList, domain tcyesno iGenerateBaseInstallation, domain tcyesno iGenerateOnlyForInstallations, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function generates Configuration Lines for the passed Service Contract Change, or Contract Quote.

	Use this Public Interface to copy an Installation Group and Items or Serialized Items to Contract Configuration Lines. The functionality offered in this Public Interface is the same as available in session Generate Contract Configuration Lines (tsctm2210m000). See the session's documentation for more information. Pre: a db.retry.point() must have been specified. Post: an abort.transaction() or commit.transaction() must be executed. Input: iGenerateFor Generate Configuration Lines for Service Contract, or Contract Quote Mandatory iServiceContractOrContractQuote Service Contract, or Contract Quote Mandatory iServiceContractChange Service Contract Change; when 0 is passed, the Change Number of the original Service Contract is used. Not mandatory iInstallationGroup Installation Group Mandatory iFromItem From Item Not mandatory iToItem To Item; when left empty it is defaulted to its maximum value. Not mandatory iFromSerialNumber From Serial Number Not mandatory iToSerialNumber To Serial Number; when left empty it is defaulted to its maximum value. Not mandatory iRunIdentification Run Identification. A user-definable identification. Not mandatory iInstallmentTemplate Installment Template: Not mandatory iContractTemplate Contract Template: Not mandatory iApplyItemPriceListForContractTemplate Apply Item Price List For Contract Template Mandatory yes/no iPickBestFitTemplateFromItemPriceList Pick Best-Fit Template from Item-Price List Mandatory yes/no iGenerateBaseInstallation Generate Base-Installation Mandatory yes/no iGenerateOnlyForInstallations Generate Only for Installations Mandatory yes/no Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID

	An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Generation successful <> 0 - Error occurred during Generation.

ServiceContract.GetTotalSalesAmount

DLL:	tsextctmapi This function is available from KB2303875.
Syntax:	<pre>long ServiceContract.GetTotalSalesAmount(domain tcorno iServiceContract, ref domain tcamnt oContractTotalSalesAmount, ref domain tccur oContractCurrency, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl.: This function calculates the net sales amount in contract currency for the passed contract. Pre: - Post: - Input: iServiceContract Service Contract Mandatory Output: oContractTotalSalesAmount Contract Total Sales Amount oContractCurrency Service Contract Currency oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Amount could be determined. <> 0 - Error occurred.

ServiceContract.StartMultiMain

DLL:	tsextctmapi This function is available from KB3613881.
Syntax:	<pre>long ServiceContract.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcorno iServiceContract, ref domain tcmcs.s999m oExceptionMessage mb,</pre>

	ref	long	oExceptionID)
Usage:	Expl:	This function starts the Multi Main Table session 'Service Contract' (tsctm3600m000).	
	Input:	iStartMode	
		Mandatory -	Specifies the start mode for the session.
		Possible values are:	
		MODAL -	The parent session is blocked until the child session exits. The session will be started as a zoom session.
		MODELESS -	Parent and child are parallel sessions that can be manipulated simultaneously.
		iStartFilter	Not Used.
		iQueryExtend	A specific query to be used when zooming to this session.
		iServiceContract	Service Contract
	Output:	oExceptionMessage	The last message if any message is found. If more than one message is given, these are present in the oExceptionID.
		oExceptionID	An ID that refers to the exception information. Use the functions in Exception to get all relevant information.
	Return:	0	Session started
		<> 0	An error occurred

ServiceContract.UnpackObjectReference

DLL:	tsextctmapi		
	This function is available from KB3644790 .		
Syntax:	<pre>long ServiceContract.UnpackObjectReference(domain tcborf iObjectReference, ref domain tsctm.cchn oContractChange, ref domain tsmdm.seqn oConfigurationLine, ref domain tsctm.inst oInstallmentLine, ref domain tcpono oRevenueLine, ref domain tceмм.seqn oCostLine, ref domain tcccp.yrno oYear, ref domain tcccp.peri oPeriod, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>		
Usage:	Expl:	This function unpacks the object reference string to Contract Installment Line number, Contract Revenue Line number or Contract Cost Line number.	
	Pre:	-	
	Post:	-	
	Input:	iObjectReference	- Object Reference; Mandatory
	Output:	oContractChange	- Contract Change number
		oConfigurationLine	- Line number of the Contract Configuration Line
		oInstallmentLine	- Line number or Contract Installment Line

	oRevenueLine - Line number or Revenue Line oCostLine - Line number or Contract Cost Line oYear - Year number oPeriod - Period number oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Note that if the return value of this function is unequal zero, then we are dealing with an error situation. Return: 0 - No Error, and the object reference could be unpacked correctly. <> 0 - Error, and the object reference could not be unpacked correctly.

Public Interfaces for ServiceContractChange

The following functions are available:

[ServiceContractChange.Create](#)

ServiceContractChange.Create

DLL:	tsxtctmapi
	This function is available from KB3565748 .
Syntax:	<pre>long ServiceContractChange.Create(domain tcorno iServiceContract, long iProcessingOptionSet, ref domain tsctm.cchn oContractChange, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl.: This function creates a Service Contract Change. It offers the same functionality as session Service Contract (tsctm3120m000). Normal defaulting will be applied for attributes if not specified as input arguments. The Help of this session can be used as reference. Pre: A db.retry.point() must have been specified.

Call `ProcessingOptionSet.Create()` to obtain `iProcessingOptionSet`.

Post: An `abort.transaction()` or `commit.transaction()` must be executed.

Delete the option set by calling `ProcessingOptionSet.Delete()`.

Input: `iServiceContract`
Service Contract.
Mandatory.

`iProcessingOptionSet`
Processing Option Set: a processing option set number referring to a processing option set containing at least one valid option.
Mandatory.

NAME	TYPE	DEFAULT
=====		
<code>ContractChangeType</code>	domain <code>tsctm.chng</code>	<code>tsctm.chng.renewed</code>
The type of change to be created for the service contract.		
<code>Description</code>	domain <code>tsmdm.dsca</code>	""
The description of the contract change. When not specified then contract header description is set on the new contract change.		
<code>OriginContractChange</code>	domain <code>tsctm.cchn</code>	0
The contract change number on which the change of type Incidental Change is applied.		
<code>ChangeDate</code>	domain <code>tsmdm.date</code>	0
The change date on which the change of type Incidental Change or Indexation is effective. When not specified for a change of type Incidental Change the current date will be defaulted and for a change of type Indexation, the indexation start date defined on the service contract header will be applied to the new contract change.		
<code>Penalty</code>	domain <code>tcamnt</code>	0.0
The to be applied penalty for a contract change of type Incidental Change. When not specified the penalty defined on the service contract header will be applied to the new contract change.		
<code>IndexationTemplate</code>	domain <code>tsctm.cind</code>	""
The to be applied indexation template for a contract change of type Indexation or Renewal with Indexation. When not specified the indexation template defined on the service contract header will be applied to the new contract change.		

	Output: oContractChange The contract change number of the created contract change. oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - No error; however, error messages can have been set. <> 0 - An error occurred

Public Interfaces for ServiceContractConfigurationLine

The following functions are available:

[ServiceContractConfigurationLine.Create](#)

[ServiceContractConfigurationLine.GetContractCoverageDetails](#)

ServiceContractConfigurationLine.Create

DLL:	tsxtctmapi
	This function is available from KB3566270 .
Syntax:	<pre> long ServiceContractConfigurationLine.Create(domain tcorno iServiceContract fixed, domain tsctm.cchn iContractChange, long iProcessingOptionSet, ref domain tsctm.cchn oContractChange, ref domain tsmdm.seqn oConfigurationLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function creates a Service Contract Configuration Line. It offers the same functionality as session Contract Configuration Lines (tsctm1110m300). Normal defaulting will be applied for attributes if not specified as input arguments. The Help of this session can be used as reference. Pre: A db.retry.point() must have been specified.

	Call <code>ProcessingOptionSet.Create()</code> to obtain <code>iProcessingOptionSet</code>. Post: An <code>abort.transaction()</code> or <code>commit.transaction()</code> must be executed. Delete the option set by calling <code>ProcessingOptionSet.Delete()</code>. Input: <code>iServiceContract</code> The service contract for which a new configuration line is to be created. Mandatory. <code>iContractChange</code> The contract change number for which a new configuration line is to be created. When not set the configuration line will be added to the first Free contract change of the given service contract. Optional. <code>iProcessingOptionSet</code> Processing Option Set: a processing option set number referring to a processing option set containing at least one valid option. Mandatory.																																								
	<table> <tr> <th>NAME</th><th>TYPE</th></tr> <tr> <td colspan="2">=====</td></tr> <tr> <td>RunIdentification</td><td></td></tr> <tr> <td>InstallationGroup</td><td>domain tsctm.cgru</td></tr> <tr> <td>Item</td><td>domain tsbsc.clst</td></tr> <tr> <td>SerialNumber</td><td>domain tcitem</td></tr> <tr> <td>NumberOfItems</td><td>domain tcibd.sern</td></tr> <tr> <td>TypeOfConfigurationLine</td><td>domain tsctm.nitm</td></tr> <tr> <td>ReferenceActivity</td><td>domain tsctm.cltp</td></tr> <tr> <td>NumberOfActivities</td><td>domain tsacm.cact</td></tr> <tr> <td>CoverageType</td><td>domain tsctm.nrac</td></tr> <tr> <td>Duration</td><td>domain tsmdm.cctp</td></tr> <tr> <td>ContractTemplate</td><td>domain tsmdm.tmdu</td></tr> <tr> <td>PricingMethod</td><td>domain tsctm.ccte</td></tr> <tr> <td>PercentageOfSalesValue</td><td>domain tsctm.prmt</td></tr> <tr> <td>InstallmentTemplate</td><td>domain tsmdm.perc</td></tr> <tr> <td>EffectiveDate</td><td>domain tsctm.ctin</td></tr> <tr> <td>ContractDiscountDate</td><td>domain tsmdm.date</td></tr> <tr> <td>ExpiryDate</td><td>domain tsmdm.date</td></tr> <tr> <td>SalesPrice</td><td>domain tsmdm.date</td></tr> </table>	NAME	TYPE	=====		RunIdentification		InstallationGroup	domain tsctm.cgru	Item	domain tsbsc.clst	SerialNumber	domain tcitem	NumberOfItems	domain tcibd.sern	TypeOfConfigurationLine	domain tsctm.nitm	ReferenceActivity	domain tsctm.cltp	NumberOfActivities	domain tsacm.cact	CoverageType	domain tsctm.nrac	Duration	domain tsmdm.cctp	ContractTemplate	domain tsmdm.tmdu	PricingMethod	domain tsctm.ccte	PercentageOfSalesValue	domain tsctm.prmt	InstallmentTemplate	domain tsmdm.perc	EffectiveDate	domain tsctm.ctin	ContractDiscountDate	domain tsmdm.date	ExpiryDate	domain tsmdm.date	SalesPrice	domain tsmdm.date
NAME	TYPE																																								
=====																																									
RunIdentification																																									
InstallationGroup	domain tsctm.cgru																																								
Item	domain tsbsc.clst																																								
SerialNumber	domain tcitem																																								
NumberOfItems	domain tcibd.sern																																								
TypeOfConfigurationLine	domain tsctm.nitm																																								
ReferenceActivity	domain tsctm.cltp																																								
NumberOfActivities	domain tsacm.cact																																								
CoverageType	domain tsctm.nrac																																								
Duration	domain tsmdm.cctp																																								
ContractTemplate	domain tsmdm.tmdu																																								
PricingMethod	domain tsctm.ccte																																								
PercentageOfSalesValue	domain tsctm.prmt																																								
InstallmentTemplate	domain tsmdm.perc																																								
EffectiveDate	domain tsctm.ctin																																								
ContractDiscountDate	domain tsmdm.date																																								
ExpiryDate	domain tsmdm.date																																								
SalesPrice	domain tsmdm.date																																								

NumberOfPeriods	domain	tcpric
PeriodUnit	domain	tsmdm.numb
SalesAmount	domain	tsmdm.peru
SurchargePercentage	domain	tcamnt
SurchargeAmount	domain	tcdisc
ContractDiscountScheme	domain	tcamnt
CostAmount	domain	tsctm.ccds
MarkedForExpiry	domain	tcamnt
Project	domain	tcyesno
Element	domain	tccprj
Activity	domain	tccspa
ServiceArea	domain	tccact
CoverageTerms	domain	tsmdm.csar
TimeAndMaterialSalesPrices	domain	tcyesno
FixedSalesPrices	domain	tcyesno
ConformanceReporting	domain	tcyesno
ProblemSolvingMethod	domain	tcmcs.cncd
Text	domain	tcmcs.pbsm
	domain	tsmdm.text

Output: oContractChange
The contract change number the created contract configuration line is added.

oConfigurationLine
The created contract configuration line number.

oExceptionMessage
The last message if any message is found. If more than one message is given, these are present in the oExceptionID.

oExceptionID
An ID that refers to the exception information. Use the functions in Exception to get all relevant information.

Return: 0 - No error; however, error messages can have been set.
<> 0 - An error occurred

ServiceContractConfigurationLine.GetContractCoverageDetails

DLL:	tsxtctmapi This function is available from KB3627884 .
Syntax:	<pre> long ServiceContractConfigurationLine.GetContractCoverageDetails (domain tsbsc.clst iInstallationGroup, domain tcitem iItem, domain tcibd.sern iSerialNumber, domain tccom.bpid iSoldToBusinessPartner, domain tcdate iCoverageTime, long iProcessingOptionSet, ref boolean oConfigurationCoveredByContract, ref domain tcorno oContract, ref domain tsctm.term oTermId, ref domain tsmdm.seqn oConfigurationLine, ref domain tsmdm.date oEffectiveDate, ref domain tsmdm.date oExpiryDate, ref domain tsctm.ccte oContractTemplate, ref domain tsmdm.dsca oContractDescription mb, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function will check if the specified Configuration is covered by any Service Contract. The details of the most specific contract that is found are then returned. Searching for a contract will be done in the following order: - If a Serial Number is provided: 1. Search for a Contract for the Serialized Item. 2. Search for a Contract for the Parent Serialized Items in the Physical Breakdown Tree, if any. 3. Search for a Contract for the given Installation Group or, if not given, the Installation Group of the Serialized Item and the Item code, if any. 4. Search for a Contract for just the given Installation Group or if not given, the Installation Group of the Serialized Item, if any. 5. Search for a Contract for just the Item code. 6. Search for a Contract for just the Item codes of the Parent Serialized Items in the Physical Breakdown Tree, if any. - If no Serial Number is provided, but an Installation Group is: 1. Search for a Contract for the given Installation Group and the Item code, if any. 2. Search for a Contract for the given Installation Group. 3. Search for a Contract for just the Item code. - If no Serial Number or Installation Group are provided: 1. Search for a Contract for just the Item code. Note: In this case, a Business Partner is necessary. - If not enough information is provided: 1. No search will be done. Pre: Installation Group, Item or Serial must be provided. If only Item is provided, a Business Partner must be provided as well. If Serial number is provided then contract coverage must be enabled for that Serialized Item.

	Call <code>ProcessingOptionSet.Create()</code> to obtain <code>iProcessingOptionSet</code>. Post: Delete the option set by calling <code>ProcessingOptionSet.Delete()</code>. Input: <code>iInstallationGroup</code> - Installation group to check coverage for. <code>iItem</code> - Item to check coverage for. <code>iSerialNumber</code> - Configuration to check coverage for. <code>iSoldToBusinessPartner</code> - Search for contracts for this business partner only when the given configuration only consists of an Item code, or when the <code>Contracts</code> parameter "Allow Multiple Contracts" is enabled for Serialized Items or Installation Groups in Service Parameters. Note: <code>iInstallationGroup</code>, <code>iItem</code>, or <code>iSerialNumber</code> is Mandatory. If only an item is provided, then <code>iSoldToBusinessPartner</code> is Mandatory as well. <code>iCoverageTime</code> - The date at which the Contract (change) must be (or have been) active. Mandatory. Note that this domain is a UTC date. <code>iProcessingOptionSet</code> - Processing Option Set: a processing option set number referring to a processing option set containing at least one valid option. Mandatory. <table border="1"> <thead> <tr> <th>NAME</th><th>TYPE</th></tr> </thead> <tbody> <tr> <td>CheckPricing</td><td>Boolean; default is false. If true, then the Pricing Terms field on the contract configuration line (tsctml10.ptm) must be YES. If false, then the Coverage Terms field on the contract configuration line (tsctml10.ctm) must be YES.</td></tr> </tbody> </table> Output: <code>oConfigurationCoveredByContract</code> True if a contract has been found for this configuration. False if not. Then all other output variables will be empty. <code>oContract</code> Contract Number. <code>oTermId</code> Term ID of the found Contract Change. <code>oConfigurationLine</code> The found contract configuration line number. <code>oEffectiveDate</code> Effective Date of the Configuration Line. Note that this date domain is in days from year zero. <code>oExpiryDate</code> Expiry Date of the Configuration Line. Note that this date domain is in days from year zero. <code>oContractTemplate</code> Contract Template of the Configuration Line. <code>oContractDescription</code> Description of the found Contract. <code>oExceptionMessage</code> The last message if any message is found. If more than one message is given, these are present in the	NAME	TYPE	CheckPricing	Boolean; default is false. If true, then the Pricing Terms field on the contract configuration line (tsctml10.ptm) must be YES. If false, then the Coverage Terms field on the contract configuration line (tsctml10.ctm) must be YES.
NAME	TYPE				
CheckPricing	Boolean; default is false. If true, then the Pricing Terms field on the contract configuration line (tsctml10.ptm) must be YES. If false, then the Coverage Terms field on the contract configuration line (tsctml10.ctm) must be YES.				

	<pre> oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - No error; however, error messages can have been set. <> 0 - An error occurred </pre>

Public Interfaces for ServiceContractInstallment

The following functions are available:

[ServiceContractInstallment.Cancel](#)

[ServiceContractInstallment.SwitchStatus](#)

[ServiceContractInstallment.Transfer](#)

ServiceContractInstallment.Cancel

DLL:	tsxtctmapi
	This function is available from KB3606613 .
Syntax:	<pre> long ServiceContractInstallment.Cancel(domain tcorno iServiceContract, domain tsctm.inst iInstallmentNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this Public Interface to cancel a service contract installment line. The functionality offered in this Public Interface is the same as available in session Cancel Contract Installments (tsctm4202m000). See the session's documentation for more information. Pre: a db.retry.point() must have been specified. Post: an abort.transaction() or commit.transaction() must be executed. Input: i.service.contract The service contract based on which the status of the installment must be cancelled. Mandatory Input. i.installment.number The Installment number which must be cancelled. Mandatory Input. Output: oExceptionMessage

	The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 : The status of the contract installment line is changed to 'Cancelled'. <> 0 : Error occurred during cancelling the contract installment line.

ServiceContractInstallment.SwitchStatus

DLL:	tsxtctmapi
	This function is available from KB3597586 .
Syntax:	<pre>long ServiceContractInstallment.SwitchStatus(domain tcorno iServiceContract, domain tsctm.inst iInstallmentNumber, domain tsctm.stin iInstallmentStatus, ref tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this Public Interface to switch the status of a service contract installment line from 'Free' to 'Accepted' or change the status from 'Accepted' to 'Free'. The functionality offered in this Public Interface is the same as available in session Switch Status of Contract Installments (tsctm4201m000). See the session's documentation for more information. Pre: a db.retry.point() must have been specified. Post: an abort.transaction() or commit.transaction() must be executed. Input: iServiceContract The service contract based on which the status of the installment must be switched. Mandatory Input. iInstallmentNumber The Installment number which status must be switched from free to accepted or form accepted to free. Mandatory Input. iInstallmentStatus This is the status the service contract installment will be switched to. Only the status 'Free' or 'Accepted' are applicable for this public interface. If the installment already has the status of this argument, the status will not be changed. Mandatory Input. Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID

	An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 : The status of the contract installment line is changed to 'Accepted'. <> 0 : Error occurred during the change of the status of the contract installment line.

ServiceContractInstallment.Transfer

DLL:	tsxtctmapi This function is available from KB3597592.
Syntax:	<pre>long ServiceContractInstallment.Transfer(domain tcorno iServiceContract, domain tsctm.inst iInstallmentNumber, domain tcsli.stat iInvoiceStatus, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this public interface to transfer installments of a Contract with the status "Accepted" to the central invoicing module. The transferred installment can have the status "On Hold" or "Confirmed" in the central invoicing module. The functionality offered in this Public Interface is the same as available in session Transfer Installments to Invoicing (tsctm4203m000). See the session's documentation for more information. Pre: a db.retry.point() must have been specified. Post: an abort.transaction() or commit.transaction() must be executed. Input: iServiceContract The service contract for which the installment will be transferred. Mandatory Input. iInstallmentNumber The installment number which will be transferred to central invoicing. Mandatory Input. iInvoiceStatus This is the status of the invoice after the transfer. The invoice can have the status "On Hold" or "Confirmed". Mandatory Input. Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 : The status of the contract installment line is changed to 'Accepted'. <> 0 : Error occurred during the change of the status of the

	contract installment line.

Public Interfaces for ServiceOrder

The following functions are available:

[ServiceOrder.Cancel](#)

[ServiceOrder.CheckAllMaterialLinesHaveBeenReceivedAndIssued](#)

[ServiceOrder.Close](#)

[ServiceOrder.Complete](#)

[ServiceOrder.Cost](#)

[ServiceOrder.GenerateActivitiesFromMasterRouting](#)

[ServiceOrder.GetTotalSalesAndCostAmounts](#)

[ServiceOrder.IssueInventory](#)

[ServiceOrder.Plan](#)

[ServiceOrder.ReadShippingAddressForCostLine](#)

[ServiceOrder.Release](#)

[ServiceOrder.StartMultiMain](#)

ServiceOrder.Cancel

DLL:	ttextsocapi			
Syntax:	<pre>long ServiceOrder.Cancel(const domain tcorno iServiceOrder fixed, const domain tccdis iCancelReason fixed, const domain tsmdm.text iCancelReasonText, const domain tsmdm.date iCancelLocalDate, const domain tcyesno iAllowCancelIfDispositionForNCRExists, const domain tcyesno iAllowCancelIfDispositionForFRACASExists, const domain tcyesno iAllowCancelIfOpenAssignmentExists, const domain tcyesno iSetRelatedCallToSolved, const domain tsspc.stat iNewPlannedActivityStatus, const domain tcyesno iCancelPurchaseOrders, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>			

Usage:	Expl: Use this function to cancel one specific service order (tssoc200 record). See functionality when a service order is cancelled with session 'Cancel Service Orders' (tssoc2204m000). If the cancel action does not succeed for whatever reason, this function will return a value unequal zero and the reason why the cancel action was not successful is present in the oExceptionMessage and oExceptionID. If cancelling succeeds then the value zero is returned. Note that because of the possibility that for return warehouse orders, some warehouse activities might have been set to automatic, the system will commit the database transactions in this function. This means that it is not necessary to set a db.retry.point() before calling this function and abort/commit after this function, because that is already handled within this function. Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post: If warehouse procedures have been set to automatic, then these are executed. Input: <table border="0"> <tr> <td>iServiceOrder</td><td>-</td></tr> <tr> <td colspan="2">The service order.</td></tr> <tr> <td colspan="2">Mandatory input.</td></tr> <tr> <td>iCancelReason</td><td>-</td></tr> <tr> <td colspan="2">The cancel reason. Note that only a reason code of type 'Cancellation' which is valid on the iCancelLocalDate is allowed. If the cancel action is successful then this is stored as cancel reason on related cancelled activities and on the service order header.</td></tr> <tr> <td colspan="2">Mandatory input.</td></tr> <tr> <td>iCancelReasonText</td><td>-</td></tr> <tr> <td colspan="2">A valid text number. If the cancel action is successful then it is stored as the cancel text on related cancelled activities and on the service order header.</td></tr> <tr> <td colspan="2">Not mandatory input.</td></tr> <tr> <td>iCancelLocalDate</td><td>-</td></tr> <tr> <td colspan="2">The date which is used as date for the cancel action. Note that this is a local date, so it represents the days since year 0. If the value 0 is given as input argument, the system will use the current date (= date.num()). It is stored as the date of cancellation on related cancelled activities and on the service order header.</td></tr> <tr> <td colspan="2">Not mandatory input.</td></tr> <tr> <td>iAllowCancelIfDispositionForNCRExists</td><td>-</td></tr> <tr> <td colspan="2">Indicator whether it is allowed for the cancel process to continue if the current service order is the disposition of a Non-Conformance report in Quality Management.</td></tr> <tr> <td colspan="2">Mandatory input.</td></tr> <tr> <td>iAllowCancelIfDispositionForFRACASExists</td><td>-</td></tr> <tr> <td colspan="2">Indicator whether it is allowed for the cancel process to continue if the current service order is the disposition of a FRACAS report in Quality Management.</td></tr> <tr> <td colspan="2">Mandatory input.</td></tr> <tr> <td>iAllowCancelIfOpenAssignmentExists</td><td>-</td></tr> <tr> <td colspan="2">Indicator whether it is allowed for the cancel process to continue if related to the current service order, open assignments exist (tssoc205).</td></tr> </table>	iServiceOrder	-	The service order.		Mandatory input.		iCancelReason	-	The cancel reason. Note that only a reason code of type 'Cancellation' which is valid on the iCancelLocalDate is allowed. If the cancel action is successful then this is stored as cancel reason on related cancelled activities and on the service order header.		Mandatory input.		iCancelReasonText	-	A valid text number. If the cancel action is successful then it is stored as the cancel text on related cancelled activities and on the service order header.		Not mandatory input.		iCancelLocalDate	-	The date which is used as date for the cancel action. Note that this is a local date, so it represents the days since year 0. If the value 0 is given as input argument, the system will use the current date (= date.num()). It is stored as the date of cancellation on related cancelled activities and on the service order header.		Not mandatory input.		iAllowCancelIfDispositionForNCRExists	-	Indicator whether it is allowed for the cancel process to continue if the current service order is the disposition of a Non-Conformance report in Quality Management.		Mandatory input.		iAllowCancelIfDispositionForFRACASExists	-	Indicator whether it is allowed for the cancel process to continue if the current service order is the disposition of a FRACAS report in Quality Management.		Mandatory input.		iAllowCancelIfOpenAssignmentExists	-	Indicator whether it is allowed for the cancel process to continue if related to the current service order, open assignments exist (tssoc205).	
iServiceOrder	-																																								
The service order.																																									
Mandatory input.																																									
iCancelReason	-																																								
The cancel reason. Note that only a reason code of type 'Cancellation' which is valid on the iCancelLocalDate is allowed. If the cancel action is successful then this is stored as cancel reason on related cancelled activities and on the service order header.																																									
Mandatory input.																																									
iCancelReasonText	-																																								
A valid text number. If the cancel action is successful then it is stored as the cancel text on related cancelled activities and on the service order header.																																									
Not mandatory input.																																									
iCancelLocalDate	-																																								
The date which is used as date for the cancel action. Note that this is a local date, so it represents the days since year 0. If the value 0 is given as input argument, the system will use the current date (= date.num()). It is stored as the date of cancellation on related cancelled activities and on the service order header.																																									
Not mandatory input.																																									
iAllowCancelIfDispositionForNCRExists	-																																								
Indicator whether it is allowed for the cancel process to continue if the current service order is the disposition of a Non-Conformance report in Quality Management.																																									
Mandatory input.																																									
iAllowCancelIfDispositionForFRACASExists	-																																								
Indicator whether it is allowed for the cancel process to continue if the current service order is the disposition of a FRACAS report in Quality Management.																																									
Mandatory input.																																									
iAllowCancelIfOpenAssignmentExists	-																																								
Indicator whether it is allowed for the cancel process to continue if related to the current service order, open assignments exist (tssoc205).																																									

	Mandatory input. iSetRelatedCallToSolved - If this indicator is set to 'Yes' and the current service order originates from a Call and the cancellation succeeds, the related Call will be set to the status 'Solved'. Mandatory input. iNewPlannedActivityStatus - If the current service order originates from a Planned Activity (tsspc200), this will be the status to which the related planned activity will be set back. Allowed values are tsspc.stat.free, tsspc.stat.released and tsspc.stat.canceled. Mandatory input. iCancelPurchaseOrders - If this indicator is set to 'Yes', then Procurement will try to Delete/Cancel (first delete and if that is no longer allowed cancel) related purchase orders when that is still allowed. Mandatory input. Output: oExceptionMessage The last message if the return value unequals 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error or process information. Use the functions in Exception to get all relevant information. Note that if the return value of this function is unequal zero, then we are dealing with an error situation and the status of the service order was not changed to 'Cancelled'. Return: 0 - No Error and the status of the given service order changed to 'Cancelled'. <> 0 - The status of the service order could not be changed to 'Cancelled'.

ServiceOrder.CheckAllMaterialLinesHaveBeenReceivedAndIssued

DLL:	tsxtsocapi
Syntax:	<pre>long ServiceOrder.CheckAllMaterialLinesHaveBeenReceivedAndIssued(const domain tcorno iServiceOrder fixed, const domain tsmdm.acln iActivityLine, const boolean iCheckSubsequentDeliveryQuantity, const boolean iCheckNonConsumed, ref boolean oAllMaterialLinesHaveBeenReceivedAndIssued, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this function to see if for a certain service order or service order activity all related material lines (tssoc220) have been issued (delivery types with an issue) or received (delivery types with a receipt).

	If all material lines related to the service order have to be checked, then give for the iActivityLine the value 0. If only for a specific activity line number this check has to be executed then specify the iActivityLine with an existing activity line number. If the subsequent delivery quantity also has to be taken into account, then set the iCheckSubsequentDeliveryQuantity to 'true' else to 'false'. The subsequent delivery quantity is used for all the 'from warehouse' kind of flows, so it concerns issues. Normally you also would want to check the subsequent delivery quantity, so fill this input boolean with 'true'. Not checking the subsequent delivery quantity can be an option, when during the costing process the user indicates that the subsequent quantity will be removed automatically. If that is the normal way of working, then you can indicate that the subsequent quantity should not be taken into account when checking if everything has been issued. If also the non-consumed lines have to be taken into account, then set the iCheckNonConsumed to 'true' else to 'false'. If a to-warehouse line is linked (related line number is filled), with a from-warehouse line, then, if the iCheckNonConsumed is set to 'true', this function will also check whether the receipt for this non-consumed line has been executed. Pre: N.A. Post: N.A. Input: iServiceOrder - The service order. This is mandatory input and the service order should exist. iActivityLine - The activity line number. If a value of 0 is specified, then this function will check all material lines (tssoc220) related to the iServiceOrder. If a value <> 0 is specified, then this service order activity has to exist and only the material lines linked to this specific activity line are checked. iCheckSubsequentDeliveryQuantity - If set to 'true', then for material lines with a delivery type which are dealing with an issue the system will also check if the subsequent delivery quantity (tssoc220.qtsd) has been issued. iCheckNonConsumed - If a 'to warehouse' material line is linked with the related line number field to a 'from warehouse' line, and this input boolean is set to 'true', then the system will also check that these 'non-consumed' material lines have been received. Output: oAllMaterialLinesHaveBeenReceivedAndIssued - If 'true' is returned, then all material lines linked to the service order or service order activity have been received and issued. If 'false' is returned, then not all material lines linked to the service order or service order activity have been received and issued. oExceptionMessage The last message if the return value unequals 0. If the return value equals 0, but the oAllMaterialLinesHaveBeenReceivedAndIssued is 'false', then it contains the last message indicating for which material line (and the reason) not everything has been

	issued or received. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Note that if the return value of this function is unequal zero, then we are dealing with an error situation. If the return value = 0 (so not an error), but the oAllMaterialLinesHaveBeenReceivedAndIssued is 'false', then this contains all the usefull information for the first material line for which not everything has been received/issued. Return: 0 - No Error. <> 0 - Error.

ServiceOrder.Close

DLL:	tsxtsocapi
	This function is available from KB2297823 .
Syntax:	<pre>long ServiceOrder.Close(domain tcorno iServiceOrder, domain tcyesno iDeleteServiceOrder, domain tcyesno iCopyServiceOrderToHistory, domain tcyesno iClosePhysicalBreakdownChanges, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this function to Close a Service Order. The same functionality is offered as available in session Close Service Orders (tsoc2201m000). Pre: db.retry.point() set Post: commit/abort transaction Input: iServiceOrder Service Order Mandatory iDeleteServiceOrder Delete Service Order Mandatory iCopyServiceOrderToHistory Copy Service Order to History. This argument is mandatory when Service Order Parameter Service Order History is checked. When iDeleteServiceOrder is Yes, iCopyServiceOrderToHistory is defaulted to Yes. iClosePhysicalBreakdownChanges Close Physical Breakdown Changes. This argument is mandatory when Physical Breakdown Changes is checked in Configuration Management Parameters. Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the

	oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - No Error <> 0 - Error

ServiceOrder.Complete

DLL:	tsextsocapi
Syntax:	<pre>long ServiceOrder.Complete(const domain tcorno iServiceOrder fixed, const domain tcyesno iAllowCompletingWithEmptyProblemCode, const domain tcyesno iAllowCompletingWithEmptySolutionCode, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this function to complete one specific service order (tssoc200 record). See functionality when a service order is completed with form command 'Complete' in session 'Service Orders' (tssoc2100m000) or 'Service Order' (tssoc2100m100). If the complete action does not succeed for whatever reason, this function will return a value unequal zero and the reason why the complete action was not successful is present in the oExceptionMessage and oExceptionID. If completing succeeds then the value zero is returned. This function does its own database handling, so there is no need to specify a db.retry.point() before calling this function and to abort/commit the transactions afterwards. This is necessary because the system will execute the blocking checks and if the order gets blocked, the blocking flag on header level is set and committed and the complete action will fail. If return warehouse orders are created, then the system will at the end also process the warehouse activities which are set to automatic. Note: - When an Electronic Signature is required for completing a service order, this public interface can only be used when called in an LN UI component. Only in that case the signature request dialog can be started. - When interactive counter reading reset rules are defined for any of the service order activities, the session for resetting the counters is not started when this public interface is used. Resetting these counters can be done using LN UI. Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post: If the order gets blocked, the blocking flag is set on header level. If return warehouse orders are created and warehouse activities have been set to 'automatic', these are executed.

	Input: iServiceOrder - The service order. Mandatory input. iAllowCompletingWithEmptyProblemCode - With this input argument the user can indicate that the complete action should continue if an activity is set to 'Completed' which has an empty problem code and either on the user template or service type, it is indicated that this check should result in a warning. Note: this situation would result in LN UI in a question for the user. Mandatory input. iAllowCompletingWithEmptySolutionCode - With this input argument the user can indicate that the complete action should continue if an activity is set to 'Completed' which has an empty solution code and either on the user template or service type, it is indicated that this check should result in a warning. Note: this situation would result in LN UI in a question for the user. Mandatory input. Output: oExceptionMessage The last message if the return value unequal 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error or process information. Use the functions in Exception to get all relevant information. Note that if the return value of this function is unequal zero, then we are dealing with an error situation and the status of the service order was not changed to 'Completed'. Return: 0 - No Error and the status of the given service order changed to 'Completed'. <> 0 - The status of the service order could not be changed to 'Completed'.

ServiceOrder.Cost

DLL:	tsextsocapi
Syntax:	<pre>long ServiceOrder.Cost(const domain tcorno iServiceOrder fixed, const domain tcyesno iIncludeReleasedOrdersAndActivities, const domain tcyesno iCheckUnconsumedItems, const domain tcyesno iRemoveSubsequentDeliveryQuantity, const domain tcyesno iInvoiceFreightLinesinFreightManagement, const domain tcyesno iIgnoreUnapprovedInvoices, const domain tcyesno iCostZeroQuantityForOtherCosts, const domain tcyesno iDoNotCostIfOpenInspectionsExist, const domain tcyesno iSetOpenInspectionsToNotMeasured, const domain tcyesno iDeleteOpenInspections, const domain tcccur iCurrency fixed,</pre>

	const domain tcamnt iMaximumLimitForInvoiceAmount, const domain tcamnt iMaximumLimitForOtherAmount, const domain tcprcg iLowerMargin, const domain tcprcg iUpperMargin, const domain tcsli.stat iInvoiceLineStatus, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: Use this function to cost one specific service order (tssoc200). See functionality in the costing session (tssoc2290m000) when the Selection Criteria Level is set to 'Order'. All options available in the costing session tssoc2290m000 are available as input arguments. The labels of the options from the session tssoc2290m000 are taken over to the input variables, so that also the help from this session can be used to get more information. If setting the order to status 'Costed' does not succeed for whatever reason, this function will return a value unequal zero and the reason why the costing was not successful is present in the oExceptionMessage and oExceptionID. If costing succeeds then the value zero is returned. This function does its own database handling, so there is no need to specify a db.retry.point() before calling this function and to abort/commit the transactions afterwards. Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post: Input: iServiceOrder - The service order. Mandatory input. iIncludeReleasedOrdersAndActivities - If this option is set to 'yes', then also if the status of the order is 'Released', it is considered for costing. Mandatory input. iCheckUnconsumedItems - If a material line is being costed, which is related to the given order, and this parameter is set to 'yes', then if not all unconsumed items have been returned, the costing will not proceed. If set to 'no', then the system will continue the costing even if there is still unconsumed quantity available. Mandatory input. iRemoveSubsequentDeliveryQuantity - If a material line is being costed, which is related to the order, and the subsequent delivery quantity is unequal zero, then with this option set to 'yes', the system will automatically set the subsequent delivery quantity on the material line to zero. Mandatory input. InvoiceFreightLinesinFreightManagement - If this option is set to 'yes', are there are freight invoice lines in Freight Management which have not been sent to Central Invoicing, then these are automatically released to Central Invoicing. Mandatory input. iIgnoreUnapprovedInvoices -

	If this option is set to 'yes' and if a subcontracting other cost line is being costed and not all purchase invoices related to the related purchase order line have been approved, then costing will still proceed. Mandatory input. iCostZeroQuantityForOtherCosts - If this option is set to 'no' and if another cost line is being costed (tssoc240) and the actual quantity is still zero, then costing will not proceed. Note that a Total Travel Line is excluded from this check, because the actual quantity for a Total Travel Line is always zero. Mandatory input. iDoNotCostIfOpenInspectionsExist - If this option is set to 'yes' and if open inspections exist (tscfg300) related to the order, then costing will not succeed. Mandatory input. iSetOpenInspectionsToNotMeasured - If the input argument iDoNotCostIfOpenExpectationsExist is set to 'no', this input can be set to 'yes'. If set to 'yes', then any open inspection will get the status 'Not Measured'. Mandatory input. iDeleteOpenInspections - If the input argument iDoNotCostIfOpenExpectationsExist is set to 'no' and the input argument iSetOpenInspectionsToNotMeasured is set to 'no', this input argument can be set to 'yes'. If set to 'yes', then any open inspection will be deleted. Mandatory input. iCurrency - The currency in which the input arguments iMaximumLimitForInvoiceAmount and iMaximumLimitForOtherAmount (Goodwill Amount) are expressed. Mandatory input. iMaximumLimitForInvoiceAmount - The maximum net invoice amount of the service order which is allowed to be costed. iMaximumLimitForOtherAmount - The maximum other amount (is actually the goodwill amount) which is allowed to be costed. iLowerMargin - If margin control is applicable in the service order parameters (or in the settings by service office if the concept of sites has been activated), then this is the lowest margin for which it is allowed to cost the given service order. iUpperMargin - If margin control is applicable in the service order parameters (or in the settings by service office if the concept of sites has been activated), then this is the highest margin for which it is allowed to cost the given service order. iInvoiceLineStatus - The line status with which the billable line in Central Invoicing is created. Allowed values are: tcsli.stat.on.hold tcsli.stat.confirmed tcsli.stat.not.appl

	If the status is set to 'tcsli.stat.not.appl' then this will mean that the system will default it again from either the service order parameters or from the settings per office, if the concept of sites has been activated. Mandatory input. Output: oExceptionMessage The last message if the return value unequals 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error or process information. Use the functions in Exception to get all relevant information. Note that if the return value of this function is unequal zero, then the status of the service order is not changed to 'Costed'. Return: 0 - No Error and the status of the given service order changed to 'Costed'. <> 0 - The status of the service order could not be changed to 'Costed'.

ServiceOrder.GenerateActivitiesFromMasterRouting

DLL:	tsextsocapi
Syntax:	<pre>long ServiceOrder.GenerateActivitiesFromMasterRouting(const domain tcorno iServiceOrder fixed, const boolean iUseAttributesFromOrder, const domain tsbsc.clst iInstallationGroup fixed, const domain tcitem iItem fixed, const domain tcibd.sern iSerialNumber fixed, const domain tsacm.cact iMasterRouting fixed, const long iNumberOfRoutingOptions, const domain tsacm.cact iRoutingOptions() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl. : With this function service order activities (tssoc210) for a certain service order (iServiceOrder) can be generated based on a master routing (iMasterRouting). With the set of input variables iUseAttributesFromOrder/iInstallationGroup/iItem /iSerialNumber there is control what the installation group, item and serial number on the created service order activities is going to be. See the description of these input arguments for more info. With the set input variables iNumberOfRoutingOptions and the array iRoutingOptions, the user can control the list of routing options which has to be executed for the given master routing. See the description of these input arguments for more info. Pre : a db.retry.point() must have been specified. Post : an abort.transaction() or commit.transaction() must be

	executed. Input : iServiceOrder The service order for which the master routing has to be executed (mandatory input and should exist). iUseAttributesFromOrder If this boolean is set to 'true', then the activities (tssoc210) which are going to be created from the given master routing are using the installation group/item and serial number from the service order header (tssoc200). If this boolean is set to 'false', then the installation group/item/serial number which are going to be used on the created service order activities are taken from the input arguments iInstallationGroup/iItem and iSerialNumber. iInstallationGroup If iUseAttributesFromOrder is set to 'false', then the iInstallationGroup is the installation group which is used on the created service order activities. If iUseAttributesFromOrder is set to 'true', then this input argument is ignored. iItem If iUseAttributesFromOrder is set to 'false', then the iItem is the item which is used on the created service order activities. If iUseAttributesFromOrder is set to 'true', then this input argument is ignored. iSerialNumber If iUseAttributesFromOrder is set to 'false', then the iSerialNumber is the serial number which is used on the created service order activities. If iUseAttributesFromOrder is set to 'true', then this input argument is ignored. iMasterRouting The master routing which is going to be used to generate the service order activities. (mandatory input and should exist). iNumberOfRoutingOptions If the iNumberOfRoutingOptions is specified as zero, then, if multiple routing options are present for the specified master routing, the default routing option is executed. If the iNumberOfRoutingOptions is not zero, then the routing options from the array iRoutingOptions are executed. The maximum number of routing options which can be used is 10. iRoutingOptions The array with routing options which need to be executed for the iMasterRouting. They are only considered if the iNumberOfRoutingOptions is unequal to zero. If iNumberOfRoutingOptions is not zero, then the first iNumberOfRoutingOptions array elements should exist (allocated) and should not be empty. Note that if an array is given with routing options which are not related to the iMasterRouting then no service order activities will be created, so the system does not check the validity of these routing options. oExceptionMessage The last message if the return value unequals 0. If more than one message is given, these are present in

	the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Output : N.A. Return : 0 <> 0 - No error - An error occurred

ServiceOrder.GetTotalSalesAndCostAmounts

DLL:	tsextsocapi
Syntax:	<pre> long ServiceOrder.GetTotalSalesAndCostAmounts(domain tcorno iServiceOrder, domain tsmdm.acln iActivityLine, boolean iOnlyCostLinesDirectlyLinkedToHeader, domain tsmdm.cotp iCostType, boolean iSkipSalesAmountsCoveredByQuote, boolean iCalculateEstimatedAmounts, ref long oNumberOfHomeCurrencies, ref domain tcccur oArrayWithHomeCurrencies() fixed, ref domain tcamnt oTotalEstimatedSalesAmount, ref domain tcamnt oTotalEstimatedCostAmountArray(), ref domain tcamnt oTotalEstimatedInvoiceableSalesAmount, ref domain tcamnt oTotalEstimatedInvoiceableCostAmountArray(), ref domain tcamnt oTotalEstimatedNettSalesAmount, ref domain tcamnt oTotalSalesAmount, ref domain tcamnt oTotalCostAmountArray(), ref domain tcamnt oTotalInvoiceableSalesAmount, ref domain tcamnt oTotalInvoiceableCostAmountArray(), ref domain tcamnt oTotalNettSalesAmount, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function will retrieve the total of all the sales and cost amounts of all material, labor or other lines belonging to the Service Order or Activity Line. If 'iActivityLine' is zero, all lines are aggregated. If 'iActivityLine' is not zero then the totals are calculated for the given activity line. If 'iActivityLine' is zero, but 'iOnlyCostLinesDirectlyLinkedToHeader' is true, the amount of the cost lines directly linked to the header is calculated. If 'iSkipSalesAmountsCoveredByQuote' is true, sales amounts of cost lines that are covered by a quote are not added to the sales totals. If 'iCalculateEstimatedAmounts' is true, the estimated total amounts of estimated lines are calculated as well. Otherwise these will remain zero. Pre: iServiceOrder should be filled and the output arrays should have been allocated three elements deep. Post: N.A. Input: iServiceOrder - The Service Order Mandatory iActivityLine - The Activity Line Not Mandatory If not zero, cost lines

	linked to the specific activity are aggregated. iOnlyCostLinesDirectlyLinkedToHeader - Used if Activity Line is zero only. If true, cost lines directly linked to the header are aggregated. If false, all cost lines belonging to the order are aggregated. iCostType - The Cost Type to indicate the type of costs to be calculated. Possible values: - Material Material Cost lines are aggregated. - Labor Labor Cost lines are aggregated. - Other Other Cost lines are aggregated (all cost types except material and labor). - Order All Cost lines of the entire Order are aggregated. - Activity All Cost lines of the given Activity are aggregated. The Activity is mandatory in this case. Mandatory iSkipSalesAmountsCoveredByQuote - If true, sales amounts of cost lines covered by a quote are not added to the total. iCalculateEstimatedAmounts - If true estimated cost line amounts are calculated as well. Output: oNumberOfHomeCurrencies - The Number of Home Currencies which relate to the financial company of the work order department. oArrayWithHomeCurrencies - The Array with Home Currencies which relate to the financial company of the work order department. oTotalEstimatedSalesAmount - The Total Estimated Sales Amount. oTotalEstimatedCostAmountArray - The array with the Total Estimated Cost Amount. oTotalEstimatedInvoiceableSalesAmount - The Total Estimated Invoiceable Sales Amount. oTotalEstimatedInvoiceableCostAmountArray - The array with the Total

	Estimated Invoiceable Cost Amount. oTotalEstimatedNettSalesAmount oTotalSalesAmount oTotalCostAmountArray oTotalInvoiceableSalesAmount oTotalInvoiceableCostAmountArray oTotalNettSalesAmount oExceptionMessage oExceptionID Return: 0 <> 0	- The Total Net Sales Amount. - The Total Sales Amount. - The array with the Total Cost Amount. - The Total Invoiceable Sales Amount. - The array with the Total Invoiceable Cost Amount. - The Total Net Sales Amount. - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. - No Error - Error

ServiceOrder.IssueInventory

DLL:	tsxtsocapi
	This function is available from KB2307524 .
Syntax:	<pre>long ServiceOrder.IssueInventory(domain tcorno iServiceOrder, domain tsmdm.acln iActivityLine, domain tcpono iMaterialLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this function to start the process of issuing materials for a Service Order, Service Order Activity, or Service Order Material Line. If all material lines related to the Service Order have to be done, then pass iActivityLine and iMaterialLine as 0. If all material lines related to a Service Order Activity have to be done, then pass iMaterialLine as 0. If only for a specific material line this process has to be executed, then pass iServiceOrder and iMaterialLine. iActivityLine is ignored in this case. Note that because of the possibility that for warehouse orders some warehouse activities might have been set to automatic, the system will commit the database transactions in

	this function. This means that it is not necessary to set a <code>db.retry.point()</code> before calling this function and abort/commit after this function, because that is already handled within this function. Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post: If warehouse procedures have been set to automatic, then these are executed. Input: <code>iServiceOrder</code> Service Order Mandatory <i>iActivityLine</i> Service Order Activity Not mandatory <i>iMaterialLine</i> Service Order Material Line Not mandatory Output: <code>oExceptionMessage</code> The last message if any message is found. If more than one message is given, these are present in the <code>oExceptionID</code>. <code>oExceptionID</code> An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - Error

ServiceOrder.Plan

DLL:	tsextsocapi
	This function is available from KB2290749 .
Syntax:	<pre> long ServiceOrder.Plan(domain tcorno iServiceOrder fixed, domain tcyesno iReplan, domain tcyesno iPerformATPCheck, domain tcyesno iPerformPlannedAvailableCheck, domain tcyesno iPerformOnHandAvailableCheck, domain tcyesno iSkipBlockedInventory, domain tcyesno iBlockPlanningOfOrdersWithShortages, domain tsmdm.scin iInventoryScope, domain tcyesno iUpdatePlannedDeliveryTimeOfMaterialLines, domain tcyesno iUpdateActivitiesAndOrdersWithLatestPlannedMaterialLine, domain tcyesno iSynchronizeMaterialsWithLatestPlannedMaterialLine, domain tccotp iPurchaseOrderTypeNormalDelivery, domain tcseri iPurchaseOrderSeriesNormalDelivery, domain tccotp iPurchaseOrderTypeDirectDelivery, domain tcseri iPurchaseOrderSeriesDirectDelivery, domain tccotp iServicesProcurementOrderType, domain tcseri iServicesProcurementOrderSeries, domain tcyesno iOverWriteManualPrice, domain tcyesno iOverWriteManualDiscount, </pre>

	<pre> domain tcyesno iRecalculatePrice, domain tcyesno iApplyCumulativePrice, domain tcyesno iRecalculateDiscount, domain tcyesno iApplyTotalDiscount, domain tcyesno iApplyCumulativeDiscounts, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this function to plan one specific Service Order (tssoc200 record). If planning does not succeed for whatever reason, this function will return a value unequal zero and the reason why the plan action was not successful is present in the oExceptionMessage and oExceptionID. If planning succeeds then the value zero is returned. Even if a value zero is returned, which means that the given Order changed its status to Planned, the oExceptionMessage can be filled (and even more information can be present in the oExceptionID), because the system generates information messages about the flow of the process. This interface has the same options as when the user would select one specific Service Order in session Service Order (tssoc2100m000/tssoc2100m100) and use the specific option Plan, which starts session Service Order Resource Planning (tssoc2260m000). All the options which are available in the planning session are also available as input arguments for this interface function. It can be useful to look at the Help for session tssoc2260m000. Note that this interface behaves exactly the same as the planning session, which means that the same checks are executed for the input process options as is being done in the planning session. If a certain combination of input arguments is not allowed, the user will be notified. Pre: db.retry.point must have been set. Post: Blocking status of the service order could have been changed. Status of Service Order and Service Order Activities could have been changed from Free to Planned. Commit/abort the transaction. Input: iServiceOrder Service Order Mandatory. iReplan If Replan is Yes, Service Order with status Free, Planned, or Released is planned or replanned. Then Service Order Activities with status Free, or Planned, are planned. If Replan is No, only Service Order (Activities) with status Free will be planned. Mandatory. iPerformATPCheck Indicator whether the ATP check is performed. Note: value Yes will be ignored and converted to No if material availability is not present in the Service Order Parameters (or site specific record, if the concept of Sites has been implemented). Mandatory. iPerformPlannedAvailableCheck Indicator whether the Planned Inventory check is performed. Note: value Yes will be ignored and converted to No if material availability is not present in the

	Service Order Parameters (or site specific record, if the concept of Sites has been implemented). Mandatory. iPerformOnHandAvailableCheck Indicator whether the On Hand Inventory check is performed. Note: value Yes will be ignored and converted to No if material availability is not present in the Service Order Parameters (or site specific record, if the concept of Sites has been implemented). Mandatory. iSkipBlockedInventory Indicator whether Blocked Inventory has to be considered during the various material availability checks. Note: value Yes only allowed if material availability is present in the Service Order Parameters (or site specific record, if the concept of Sites has been implemented). Furthermore, at least one of the input arguments iPerformPlannedAvailableCheck or iPerformOnHandAvailableCheck should have the value Yes. Mandatory. iBlockPlanningOfOrdersWithShortages If set to Yes, then the planning of the activity will not be successful if one of the material availability checks reports a shortage. Note: value Yes only allowed if at least one of the material availability checks is being executed, which means that either iPerformATPCheck, iPerformPlannedAvailableCheck or iPerformOnHandAvailableCheck should have the value Yes. Mandatory. iInventoryScope Indicator if during the various material availability checks only the current warehouse should be considered or the whole warehouse cluster. Possible values: Current Warehouse Only Checks if the material is available in the warehouse defined on the Material Line. All Warehouses in Planning Cluster Checks if the material is available in one of the warehouses in the same cluster as the warehouse defined on the Material Line. Mandatory. iUpdatePlannedDeliveryTimeOfMaterialLines Indicator whether the material availability checks should update the planned delivery time of related material lines. Note: value Yes only allowed if at least one of the input arguments iPerformATPCheck or iPerformPlannedAvailableCheck has the value Yes. Mandatory. iUpdateActivitiesAndOrdersWithLatestPlannedMaterialLine Indicator whether the material availability checks should update the activity and order with the time of the latest planned material line. Note: value Yes only allowed if at least one of the input arguments iPerformATPCheck or iPerformPlannedAvailableCheck has the value Yes. Value Yes only allowed if the input argument iUpdatePlannedDeliveryTimeOfMaterialLines is Yes.

	Mandatory input. iSynchronizeMaterialsWithLatestPlannedMaterialLine Indicator whether the material availability checks should synchronize material lines with the latest planned material line. Note: value Yes only allowed if at least one of the input arguments iPerformATPCheck or iPerformPlannedAvailableCheck has the value Yes. Value Yes only allowed if the input argument iUpdatePlannedDeliveryTimeOfMaterialLines is Yes. Mandatory. iPurchaseOrderTypeNormalDelivery If the delivery type of the required material for a service order activity is By Purchase Order, this Purchase Order Type is used to create its purchase order. Not mandatory. iPurchaseOrderSeriesNormalDelivery The number group used to assign series numbers to the generated purchase orders. Not mandatory. iPurchaseOrderTypeDirectDelivery If the delivery type of the required material for a service order activity is Supplier Direct Delivery, this Purchase Order Type is used to create its purchase order. Not mandatory. iPurchaseOrderSeriesDirectDelivery The number group used to assign series numbers to the generated purchase orders. Not mandatory. iServicesProcurementOrderType Services Procurement Order Type Not mandatory. iServicesProcurementOrderSeries Services Procurement Order Series Not mandatory. iOverWriteManualPrice Prices are recalculated only when the value is Yes, and it is allowed to recalculate the prices and discounts. When it is not allowed to recalculate the prices, iOverWriteManualPrice is set to No. Mandatory. iOverWriteManualDiscount Discounts are recalculated only when the value is Yes, and it is allowed to recalculate the discounts. When it is not allowed to recalculate the discounts, iOverWriteManualDiscount is set to No. Mandatory. iRecalculatePrice If in the pricing module, the parameter has been set in such a way that the repricing should be automatic or interactive, then this input parameter is considered. This means that if 'Yes' is given but for this order the automatic repricing is not applicable, it will be ignored and handled as if it is set to 'No'. If repricing is applicable and this input parameter is set to 'Yes', the system will reprice the order. Mandatory. iApplyCumulativePrice

	If in the pricing module, the parameter has been set in such a way that the repricing should be automatic or interactive, then this input parameter is considered. This means that if Yes is given but for this order the automatic repricing is not applicable, it will be ignored and handled as if it is set to 'No'. If repricing is applicable and this input parameter is set to 'Yes', the system will apply the concept of cumulative prices. Mandatory. iRecalculateDiscount If in the pricing module, the parameter has been set in such a way that the repricing should be automatic or interactive, then this input parameter is considered. This means that if 'Yes' is given but for this order the automatic repricing is not applicable, it will be ignored and handled as if it is set to 'No'. If repricing is applicable and this input parameter is set to 'Yes', the system will recalculate the discounts. Mandatory. iApplyTotalDiscount If in the pricing module, the parameter has been set in such a way that the repricing should be automatic or interactive, then this input parameter is considered. This means that if 'Yes' is given but for this order the automatic repricing is not applicable, it will be ignored and handled as if it is set to 'No'. If repricing is applicable and this input parameter is set to 'Yes', the system will apply the concept of total discounts. Mandatory. iApplyCumulativeDiscounts If in the pricing module, the parameter has been set in such a way that the repricing should be automatic or interactive, then this input parameter is considered. This means that if 'Yes' is given but for this order the automatic repricing is not applicable, it will be ignored and handled as if it is set to 'No'. If repricing is applicable and this input parameter is set to 'Yes', the system will apply the concept of cumulative discounts. Mandatory. Output: oExceptionMessage The last message if the return value is not equal to 0. If the return value equals 0, it can contain the last information message about the process. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error or process information. Use the functions in Exception to get all relevant information. Note that if the return value of this function is unequal zero, then we are dealing with an error situation and the status of the service order activity was not changed to Planned. If the return value = 0 (so not an error), the oExceptionID can still contain information about the process. Return: 0 - No Error and the status of the given Service Order Activity changed to Planned.

	<> 0 - The status of the Service Order Activity could not be changed to Planned.

ServiceOrder.ReadShippingAddressForCostLine

DLL:	tsextsocapi
Syntax:	<pre> long ServiceOrder.ReadShippingAddressForCostLine(domain tcorno iServiceOrder, domain tcpono iCostLine, domain tsmdm.cotp iCostType, ref domain tccom.cadr oShippingAddress, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function reads the shipping address of the given service order cost line. For cost type material the ship-to address of the material cost line is returned. For other cost types or if the material cost line ship-to address is empty, first the related Ship Material To attribute is read. Based on the Ship Material To the shipping address is retrieved from one of the following addresses: - Ship-to BP Address Address taken from order header or activity. - Location Address Address taken from order header or activity. - In-use by BP Address Address taken from the installation of order header or activity. - Dealer Address Address taken from the installation of order header or activity. - Subcontractor Address Address taken from the Buy-from BP (Subcontractor). Whether a line is linked to an activity or directly to the service order header determines whether attributes from the header or activity are used. This applies for the Ship Material To and other attributes used to determine the shipping address. The installation is the serialized item or (if empty) the installation group. Pre: N.A. Post: N.A. Input: iServiceOrder The service order: Mandatory iCostLine The cost line: Mandatory iCostType The Cost Type: Mandatory The cost line should exist. Output: oShippingAddress The shipping address. oExceptionMessage The last message if the return value is not equal to 0.

	If more than one message is given, these are present in the oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Session started An error occurred
oExceptionID	
Return: 0	
<> 0	

ServiceOrder.Release

DLL:	tsxtsocapi
Syntax:	<pre> long ServiceOrder.Release(const domain tcorno iServiceOrder fixed, const domain tcyesno iPerformATPCheck, const domain tcyesno iPerformPlannedAvailableCheck, const domain tcyesno iPerformOnHandAvailableCheck, const domain tcyesno iSkipBlockedInventory, const domain tsmdm.scin iInventoryScope, const domain tcyesno iBlockRelease, const domain tcyesno iUpdatePlannedDeliveryTimeOfMaterialLines, const domain tcyesno iUpdateActivitiesAndOrdersWithLatestPlannedMaterialLine, const domain tcyesno iSynchronizeMaterialsWithLatestPlannedMaterialLine, const domain tcyesno iUpdateAllActualRentalLocationAddresses, const domain tcyesno iIncludeOrdersWithStatusFree, const domain tcyesno iCheckCapacityAvailability, const domain tcyesno iCheckProjectStatus, const domain tcyesno iCheckServiceKitAllocation, const domain tcyesno iCheckSkills, const domain tcyesno iCheckAssignments, const domain tcyesno iCopyEstimatesToActualsForOtherCosts, const domain tcyesno iOverWriteManualPrice, const domain tcyesno iOverWriteManualDiscount, const domain tcyesno iRecalculatePrice, const domain tcyesno iApplyCumulativePrice, const domain tcyesno iRecalculateDiscount, const domain tcyesno iApplyTotalDiscount, const domain tcyesno iApplyCumulativeDiscounts, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this function to release one specific service order (tssoc200 record). If releasing does not succeed for whatever reason, this function will return a value unequal zero and the reason why the release action was not successful is present in the oExceptionMessage and oExceptionID. If releasing succeeds then the value zero is returned. Even if a value zero is returned, which means that the given order changed its status to 'Released', the oExceptionMessage can be filled (and even more information can be present in the oExceptionID), because the system generates information messages about the flow of the process (if the activity is first planned, this

	is logged as an information message etc.). This interface has exactly the same options as when the user would select one specific service order in session 'Service Orders' (tssoc2100m000) and would use the specific option 'Release', which starts the session tsspc2200m000. All the options which are available in the release session are also available as input arguments for this interface function. It can be useful to look at the help for session tssoc2200m000. Note that this interface behaves exactly the same as the release session, which means that the same checks are executed for the input process options as is being done in the release session. If a certain combination of input arguments is not allowed, the user will be notified. If the option 'iIncludeOrdersWithStatusFree' is set to 'Yes' and the user has selected an order with status 'Free', then the system will first try to Plan this order. If that succeeds the status of the order is changed to 'Planned' and committed in the database. If the change to 'Planned' succeeds or the service order already had the status 'Planned', the system will try to release this order. If that succeeds and warehouse orders have been created, and some of the warehouse orders activities have been set to automatic, the system will try to execute those warehouse activities. Note that because of the blocking check on the service order and the automatic processing of warehouse orders, the system will commit the database transactions in this function. This means that it is not necessary to set a db.retry.point() before calling this function and abort/commit after this function, because that is already handled within this function. Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post: Blocking status of the service order could have been changed. Status of service order could have been changed from 'Free' to 'Planned', if releasing itself did not succeed, but the order had initial status 'Free' and planning did succeed (this can only happen if also the input argument 'iIncludeOrdersWithStatusFree' has been set to 'Yes'). The status of the service order could have been changed from 'Free' or 'Planned' to 'Released'. If warehouse procedures have been set to automatic, then these are also executed. Input: iServiceOrder - The service order. Mandatory input. iPerformATPCheck - Indicator whether the ATP check has to be executed. Note: value 'Yes' will be ignored and converted to 'No' if material availability is not present in the service order parameters (or site specific record, if the concept of 'sites' is implemented). Furthermore the iIncludeOrdersWithStatusFree should also have the value 'Yes'. Mandatory input. iPerformPlannedAvailableCheck - Indicator whether the ATP check has to be executed. Note: value 'Yes' will be ignored and converted to

	'No' if material availability is not present in the service order parameters (or site specific record, if the concept of 'sites' is implemented). Mandatory input. iPerformOnHandAvailableCheck - Indicator whether the ATP check has to be executed. Note: value 'Yes' will be ignored and converted to 'No' if material availability is not present in the service order parameters (or site specific record, if the concept of 'sites' is implemented). Mandatory input. iSkipBlockedInventory - Indicator whether Blocked Inventory has to be considered during the various material availability checks. Note: value 'Yes' only allowed if material availability is present in the service order parameters (or site specific record, if the concept of 'sites' is implemented). Furthermore at least one of the input arguments iPerformPlannedAvailableCheck or iPerformOnHandAvailableCheck should have the value 'Yes'. Mandatory input. iInventoryScope - Indicator if during the various material availability checks only the current warehouse should be considered or the whole warehouse cluster. Mandatory input. iBlockRelease - If set to 'Yes', then the releasing of the order will not be successful if one of the material availability checks reports a shortage. Note: value 'Yes' only allowed if at least one of the material availability checks is being executed, which means that either iPerformATPCheck, iPerformPlannedAvailableCheck or iPerformOnHandAvailableCheck should have the value 'Yes'. Mandatory input. iUpdatePlannedDeliveryTimeOfMaterialLines - Indicator whether the material availability checks should update the planned delivery time of related material lines. Note: value 'Yes' only allowed if at least one of the input arguments iPerformATPCheck or iPerformPlannedAvailableCheck has the value 'Yes'. Mandatory input. iUpdateActivitiesAndOrdersWithLatestPlannedMaterialLine - Indicator whether the material availability checks should update the activities and order with the time of the latest planned material line. Note: value 'Yes' only allowed if at least one of the input arguments iPerformATPCheck or iPerformPlannedAvailableCheck has the value 'Yes'. Value 'Yes' only allowed if the input argument iUpdatePlannedDeliveryTimeOfMaterialLines is 'Yes'. Mandatory input. iSynchronizeMaterialsWithLatestPlannedMaterialLine - Indicator whether the material availability checks should synchronize material lines with the latest planned material line. Note: value 'Yes' only allowed if at least one of the input arguments iPerformATPCheck or iPerformPlannedAvailableCheck has the value 'Yes'.

	Value 'Yes' only allowed if the input argument <code>iUpdatePlannedDeliveryTimeOfMaterialLines</code> is 'Yes'. Mandatory input. <code>iUpdateAllActualRentalLocationAddresses</code> - Indicator whether the addresses of rental equipment has to be updated. If a related activity is rentable and this indicator is set to 'Yes', then the rental location of the rental equipment is updated with the location address of the activity. Mandatory input. <code>iIncludeOrdersWithStatusFree</code> - Indicator whether also an order with status 'Free' is allowed to be released. Mandatory input. <code>iCheckCapacityAvailability</code> - Indicator whether the system should check during releasing whether there is enough capacity on the specified service order. If set to 'Yes' and there is not enough capacity, the release of the order will not be executed. Mandatory input. <code>iCheckProjectStatus</code> - Indicator whether the system should check during releasing whether the status of a related project allows releasing the order. If set to 'Yes' and the status of the project is not ok, the release of the order will not be executed. Mandatory input. <code>iCheckServiceKitAllocation</code> - Indicator whether the system should check during releasing whether the service kit (if present) can be allocated. If set to 'Yes' and the allocation for the service kit cannot be done, the release of the order will not be executed. Mandatory input. <code>iCheckSkills</code> - Indicator whether the system should check during releasing whether the correct skills are available to execute each related service order activity. If set to 'Yes' and not the correct skills are available, the release of the order will not be executed. Mandatory input. <code>iCheckAssignments</code> - Indicator whether the system should check during releasing that each activity has at least one assignment. If set to 'Yes' and no assignment is related to a related activity, the release of the order will not be executed. Note: the value 'No' will be converted to 'Yes' if assignments are mandatory in the service order parameters. Mandatory input. <code>iCopyEstimatesToActualsForOtherCosts.yn</code> - Indicator whether the system should take over the estimates to the actual values for Other Costs (<code>tssoc240</code>) during the release process. Mandatory input. <code>iOverWriteManualPrice</code> - If in the pricing module, the parameter has been set in such a way that the repricing should be automatic or interactive, then this input parameter is considered. This means that if 'Yes' is given but for this order the

	automatic repricing is not applicable, it will be ignored and handled as if it is set to 'No'. If repricing is applicable for this order, this means that it is allowed by the pricing process to overwrite manually changed prices. Mandatory input. iOverWriteManualDiscount - Mandatory Input. If in the pricing module, the parameter has been set in such a way that the repricing should be automatic or interactive, then this input parameter is considered. This means that if 'Yes' is given but for this order the automatic repricing is not applicable, it will be ignored and handled as if it is set to 'No'. If repricing is applicable for this order, this means that it is allowed by the pricing process to overwrite manually changed discounts. iRecalculatePrice - If in the pricing module, the parameter has been set in such a way that the repricing should be automatic or interactive, then this input parameter is considered. This means that if 'Yes' is given but for this order the automatic repricing is not applicable, it will be ignored and handled as if it is set to 'No'. If repricing is applicable and this input parameter is set to 'Yes', the system will reprice the order. Mandatory Input. iApplyCumulativePrice - If in the pricing module, the parameter has been set in such a way that the repricing should be automatic or interactive, then this input parameter is considered. This means that if 'Yes' is given but for this order the automatic repricing is not applicable, it will be ignored and handled as if it is set to 'No'. If repricing is applicable and this input parameter is set to 'Yes', the system will apply the concept of cumulative prices. Mandatory Input. iRecalculateDiscount - If in the pricing module, the parameter has been set in such a way that the repricing should be automatic or interactive, then this input parameter is considered. This means that if 'Yes' is given but for this order the automatic repricing is not applicable, it will be ignored and handled as if it is set to 'No'. If repricing is applicable and this input parameter is set to 'Yes', the system will recalculate the discounts. Mandatory Input. iApplyTotalDiscount - If in the pricing module, the parameter has been set in such a way that the repricing should be automatic or interactive, then this input parameter is considered. This means that if 'Yes' is given but for this order the automatic repricing is not applicable, it will be ignored and handled as if it is set to 'No'. If repricing is applicable and this input parameter is set to 'Yes', the system will apply the concept of total discounts. Mandatory Input. iApplyCumulativeDiscounts - If in the pricing module, the parameter has been set in such a way that the repricing should be automatic or

	interactive, then this input parameter is considered. This means that if 'Yes' is given but for this order the automatic repricing is not applicable, it will be ignored and handled as if it is set to 'No'. If repricing is applicable and this input parameter is set to 'Yes', the system will apply the concept of cumulative discounts. Mandatory Input. Output: oExceptionMessage The last message if the return value unequal 0. If the return value equals 0, it can contain the last information message about the process. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error or process information. Use the functions in Exception to get all relevant information. Note that if the return value of this function is unequal zero, then we are dealing with an error situation and the status of the service order was not changed to 'Released'. If the return value = 0 (so not an error), the oExceptionID can still contain information about the process. Return: 0 - No Error and the status of the given service order changed to 'Released'. <> 0 - The status of the service order could not be changed to 'Released'.

ServiceOrder.StartMultiMain

DLL:	tsextsocapi
Syntax:	<pre>long ServiceOrder.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcorno iServiceOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the Multi-Main session Service Order (tssoc2100m100). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iQueryExtend A specific query to be used when zooming to this session.

	iServiceOrder Service Order
Output:	
oExceptionMessage	The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
oExceptionID	An ID that refers to all error information. Use the functions in Exception to get all relevant information.
Return: 0	Session started
<> 0	An error occurred

Public Interfaces for ServiceOrderActivity

The following functions are available:

[ServiceOrderActivity.Cancel](#)

[ServiceOrderActivity.Complete](#)

[ServiceOrderActivity.Cost](#)

[ServiceOrderActivity.Plan](#)

[ServiceOrderActivity.ProcessReturnDeliveries](#)

[ServiceOrderActivity.Release](#)

[ServiceOrderActivity.StartMultiMain](#)

ServiceOrderActivity.Cancel

DLL:	tsextsocapi
Syntax:	<pre> long ServiceOrderActivity.Cancel(const domain tcorno iServiceOrder fixed, const domain tsmdm.acln iActivityLine, const domain tccdis iCancelReason fixed, const domain tsmdm.text iCancelReasonText, const domain tsmdm.date iCancelLocalDate, const domain tcyesno iAllowCancelIfDispositionForNCRExists, const domain tcyesno iAllowCancelIfDispositionForFRACASExists, const domain tcyesno iAllowCancelIfOpenAssignmentExists, const domain tcyesno iSetRelatedCallToSolved, const domain tsspc.stat iNewPlannedActivityStatus, </pre>

	const domain tcyesno iCancelPurchaseOrders, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: Use this function to cancel one specific service order activity (tssoc210 record). See functionality when a service order activity is cancelled with session 'Cancel Service Orders' (tssoc2204m000). If the cancel action does not succeed for whatever reason, this function will return a value unequal zero and the reason why the cancel action was not successful is present in the oExceptionMessage and oExceptionID. If cancelling succeeds then the value zero is returned. If in the same process the system also sets the header to 'Cancelled', then this is logged as information message and is present in the oExceptionMessage and oExceptionID. Note that because of the possibility that for return warehouse orders, some warehouse activities might have been set to automatic, the system will commit the database transactions in this function. This means that it is not necessary to set a db.retry.point() before calling this function and abort/commit after this function, because that is already handled within this function. Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post: If the service order parameter for automatic cancellation is set, then if the current activity is the last activity which is set to 'Cancelled', then also the service order header is set to 'Cancelled'. If warehouse procedures have been set to automatic, then these are executed. Input: iServiceOrder - The service order. Mandatory input. iActivityLine - The activity line number. Mandatory input. iCancelReason - The cancel reason. Note that only a reason code of type 'Cancellation' which is valid on the iCancelLocalDate is allowed. If the cancel action succeeds then this reason is stored on the activity. If this activity is the last cancelled activity and the service order parameter for cancellation is set to 'automatic', and the header is set to 'Cancelled', then this is also the reason which is stored on service order header level. Mandatory input. iCancelReasonText - A valid text number. If the cancel action succeeds then this text is stored on the activity. If this activity is the last cancelled activity and the service order parameter for cancellation is set to 'automatic', and the header is set to 'Cancelled', then this is also the text which is stored on service order header level. Not mandatory input. iCancelLocalDate - The date which is used as date for the cancel action. Note that this is a local date, so it represents the days since year 0. If the value 0 is given as input

	argument, the system will use the current date (= date.num()). If this activity is the last cancelled activity and the service order parameter for cancellation is set to 'automatic', and the header is set to 'Cancelled', then this is also the cancellation date which is stored on service order header level. Not mandatory input. iAllowCancelIfDispositionForNCRExists - Indicator whether it is allowed for the cancel process to continue if the current service order activity is the disposition of a Non-Conformance report in Quality Management. Mandatory input. iAllowCancelIfDispositionForFRACASExists - Indicator whether it is allowed for the cancel process to continue if the current service order activity is the disposition of a FRACAS report in Quality Management. Mandatory input. iAllowCancelIfOpenAssignmentExists - Indicator whether it is allowed for the cancel process to continue if related to the current service order activity, open assignments exist (tssoc205). Mandatory input. iSetRelatedCallToSolved - If this indicator is set to 'Yes' and the current service order originates from a Call and the cancellation succeeds, the related Call will be set to the status 'Solved'. Mandatory input. iNewPlannedActivityStatus - If the current service order activity originates from a Planned Activity (tsspc200), this will be the status to which the related planned activity will be set back. Allowed values are tsspc.stat.free, tsspc.stat.released and tsspc.stat.canceled. Mandatory input. iCancelPurchaseOrders - If this indicator is set to 'Yes', then Procurement will try to Delete/Cancel (first delete and if that is no longer allowed cancel) related purchase orders when that is still allowed. Mandatory input. Output: oExceptionMessage The last message if the return value unequal 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error or process information. Use the functions in Exception to get all relevant information. Note that if the return value of this function is unequal zero, then we are dealing with an error situation and the status of the service order activity was not changed to 'Cancelled'. Return: 0 - No Error and the status of the given service order activity changed to 'Cancelled'. <> 0 - The status of the service order activity could not be changed to 'Cancelled'.

ServiceOrderActivity.Complete

DLL:	tsextsocapi
Syntax:	<pre> long ServiceOrderActivity.Complete(const domain tcorno iServiceOrder fixed, const domain tsmdm.acln iActivityLine, const domain tcyesno iAllowCompletingWithEmptyProblemCode, const domain tcyesno iAllowCompletingWithEmptySolutionCode, const domain tssoc2207.upd iUpdateMethodForRentalPeriod, const domain tccom.cadr iActualRentalLocationAddress fixed, const domain tcyesno iSetServiceOrderToCompleteWhenLastActivityIsCompleted, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this function to complete one specific service order activity (tssoc210 record). See functionality when a service order activity is completed with form command 'Set Activity to Completed' in session 'Service Order Activities' (tssoc2110m000) or 'Service Order Activity' (tssoc2110m100). If the complete action does not succeed for whatever reason, this function will return a value unequal zero and the reason why the complete action was not successful is present in the oExceptionMessage and oExceptionID. If completing succeeds then the value zero is returned. This function does its own database handling, so there is no need to specify a db.retry.point() before calling this function and to abort/commit the transactions afterwards. This is necessary because the system will execute the blocking checks and if the order gets blocked, the blocking flag on header level is set and committed and the complete action will fail. If return warehouse orders are created, then the system will at the end also process the warehouse activities which are set to automatic. Note: - When an Electronic Signature is required for completing a service order and this is the last activity which is being set to 'Completed' and the input argument iSetServiceOrderToCompleteWhenLastActivityIsCompleted is set to 'yes', then the header will not be set to 'Completed', because the electronic signature functionality is only available when completing from header level. - When interactive counter reading reset rules are defined for any of the service order activities, the session for resetting the counters is not started when this public interface is used. Resetting these counters can be done using LN UI. Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post: If the order gets blocked, the blocking flag is set on header level. If return warehouse orders are created and warehouse activities

	have been set to 'automatic', these are executed. If the current activity is the last activity which is set to 'Completed' and iSetServiceOrderToCompleteWhenLastActivityIsCompleted has the value 'yes', and the service order header could also be set to 'Completed', the status of the header is changed to 'Completed'. Input: iServiceOrder - The service order. Mandatory input. iActivityLine The activity line number. Mandatory input. iAllowCompletingWithEmptyProblemCode - With this input argument the user can indicate that the complete action should continue if the problem code on the activity is empty and either on the user template or service type, it is indicated that this check should result in a warning. Note: this situation would result in LN UI in a question for the user. Mandatory input. iAllowCompletingWithEmptySolutionCode - With this input argument the user can indicate that the complete action should continue if the solution code on the activity is empty and either on the user template or service type, it is indicated that this check should result in a warning. Note: this situation would result in LN UI in a question for the user. Mandatory input. iActualRentalLocationAddress - If the activity is related to Rental and the default return address is set to 'Other Address', then this will be the actual location address of the rental equipment. If this address is not provided and the activity is related to Rental and the default return address is set to 'Other Address', an error will be returned. Optional input. iUpdateMethodForRentalPeriod - If a related activity is set to 'Completed' which is related to Rental, then with this indicator the user can indicate how the rental period should be updated for the related Rental Cost line (tssoc240). Allowed values are: tssoc2207.upd.copy.planned - Use Agreed Time tssoc2207.upd.use.actual - Use Actual Time tssoc2207.upd.none - No update Mandatory input. iSetServiceOrderToCompleteWhenLastActivityIsCompleted - Set the service order header status to 'Complete' when the given service order activity line is completed successfully and all other service order activity lines are already set to 'Completed'. Mandatory input. Output: oExceptionMessage The last message if the return value unequals 0. If more than one message is given, these are present in the oExceptionID.

	<code>oExceptionID</code> An ID that refers to all error or process information. Use the functions in <code>Exception</code> to get all relevant information. Note that if the return value of this function is unequal zero, then we are dealing with an error situation and the status of the service order activity was not changed to 'Completed'. Return: 0 - No Error and the status of the given service order activity changed to 'Completed'. <> 0 - The status of the service order activity could not be changed to 'Completed'.

ServiceOrderActivity.Cost

DLL:	tsxtsocapi
Syntax:	<pre>long ServiceOrderActivity.Cost(const domain tcorno iServiceOrder fixed, const domain tsmdm.acln iActivityLine, const domain tcyesno iIncludeReleasedOrdersAndActivities, const domain tcyesno iSetServiceOrderToCostedWhenAllActivitiesAreCosted, const domain tcyesno iCheckUnconsumedItems, const domain tcyesno iRemoveSubsequentDeliveryQuantity, const domain tcyesno iInvoiceFreightLinesinFreightManagement, const domain tcyesno iIgnoreUnapprovedInvoices, const domain tcyesno iCostZeroQuantityForOtherCosts, const domain tcyesno iDoNotCostIfOpenInspectionsExist, const domain tcyesno iSetOpenInspectionsToNotMeasured, const domain tcyesno iDeleteOpenInspections, const domain tccur iCurrency fixed, const domain tcamnt iMaximumLimitForInvoiceAmount, const domain tcamnt iMaximumLimitForOtherAmount, const domain tcprcg iLowerMargin, const domain tcprcg iUpperMargin, const domain tcsl.stat iInvoiceLineStatus, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this function to cost one specific service order activity (tssoc210). See functionality in the costing session (tssoc2290m000) when the Selection Criteria Level is set to 'Activity'. All options available in the costing session tssoc2290m000 are available as input arguments. The labels of the options from the session tssoc2290m000 are taken over to the input variables, so that also the help from this session can be used to get more information. If setting the activity to status 'Costed' does not succeed for whatever reason, this function will return a value unequal zero and the reason why the costing was not successful is present in the <code>oExceptionMessage</code> and <code>oExceptionID</code>. If costing succeeds then the value zero is returned.

	This function does its own database handling, so there is no need to specify a <code>db.retry.point()</code> before calling this function and to abort/commit the transactions afterwards. Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post: Input: <code>iServiceOrder</code> - The service order. Mandatory input. <code>iActivityLine</code> - The activity line number. Mandatory input. <code>iIncludeReleasedOrdersAndActivities</code> - If this option is set to 'yes', then also if the status of the activity is 'Released', it is considered for costing. Mandatory input. <code>iSetServiceOrderToCostedWhenAllActivitiesAreCosted</code> - If this option is set to 'yes', the system will also set the service order header to 'Costed' if all related activities are 'Costed' and all related cost lines are 'Costed'. Mandatory input. <code>iCheckUnconsumedItems</code> - If a material line is being costed, which is related to the given activity line, and this parameter is set to 'yes', then if not all unconsumed items have been returned, the costing will not proceed. If set to 'no', then the system will continue the costing even if there is still unconsumed quantity available. Mandatory input. <code>iRemoveSubsequentDeliveryQuantity</code> - If a material line is being costed, which is related to the given activity line, and the subsequent delivery quantity is unequal zero, then with this option set to 'yes', the system will automatically set the subsequent delivery quantity on the material line to zero. Mandatory input. <code>InvoiceFreightLinesinFreightManagement</code> - If this option is set to 'yes', are there are freight invoice lines in Freight Management which have not been sent to Central Invoicing, then these are automatically released to Central Invoicing. Mandatory input. <code>iIgnoreUnapprovedInvoices</code> - If this option is set to 'yes' and if a subcontracting other cost line is being costed and not all purchase invoices related to the related purchase order line have been approved, then costing will still proceed. Mandatory input. <code>iCostZeroQuantityForOtherCosts</code> - If this option is set to 'no' and if another cost line is being costed (tssoc240) and the actual quantity is still zero, then costing will not proceed. Note that a Total Travel Line is excluded from this check, because the actual quantity for a Total Travel Line is always zero. Mandatory input.

	iDoNotCostIfOpenInspectionsExist - If this option is set to 'yes' and if the related activity can also be set to 'Costed' and open inspections exist (tscfg300) related to that activity, then costing will not succeed. If also the the related order header can be set to 'Costed' and open inspections exist related to the order, then costing will not succeed. Mandatory input. iSetOpenInspectionsToNotMeasured - If the input argument iDoNotCostIfOpenExpectationsExist is set to 'no', this input can be set to 'yes'. If set to 'yes', then any open inspection will get the status 'Not Measured'. Mandatory input. iDeleteOpenInspections - If the input argument iDoNotCostIfOpenExpectationsExist is set to 'no' and the input argument iSetOpenInspectionsToNotMeasured is set to 'no', this input argument can be set to 'yes'. If set to 'yes', then any open inspection will be deleted. Mandatory input. iCurrency - The currency in which the input arguments iMaximumLimitForInvoiceAmount and iMaximumLimitForOtherAmount are expressed. Mandatory input. iMaximumLimitForInvoiceAmount - The maximum net invoice amount of the activity line which is allowed to be costed. iMaximumLimitForOtherAmount - The maximum other amount (is actually the goodwill amount) which is allowed to be costed. iLowerMargin - If margin control is applicable in the service order parameters (or in the settings by service office if the concept of sites has been activated), then this is the lowest margin for which it is allowed to cost the given activity line. iUpperMargin - If margin control is applicable in the service order parameters (or in the settings by service office if the concept of sites has been activated), then this is the highest margin for which it is allowed to cost the given activity line. iInvoiceLineStatus - The line status with which the billable line in Central Invoicing is created. Allowed values are: tcsli.stat.on.hold tcsli.stat.confirmed tcsli.stat.not.appl If the status is set to 'tcsli.stat.not.appl' then this will mean that the system will default it again from either the service order parameters or from the settings per office, if the concept of sites has been activated. Mandatory input. Output: oExceptionMessage The last message if the return value unequals 0. If more than one message is given, these are present in the oExceptionID. oExceptionID

	An ID that refers to all error or process information. Use the functions in Exception to get all relevant information. Note that if the return value of this function is unequal zero, then the status of the activity line is not changed to 'Costed'. Return: 0 - No Error and the status of the given service order activity changed to 'Costed'. <> 0 - The status of the service order activity could not be changed to 'Costed'.

ServiceOrderActivity.Plan

DLL:	tsxtsocapi
	This function is available from KB2290749 .
Syntax:	<pre> long ServiceOrderActivity.Plan(domain tcorno iServiceOrder fixed, domain tsmdm.acln iActivityLine, domain tcyesno iReplan, domain tcyesno iPerformATPCheck, domain tcyesno iPerformPlannedAvailableCheck, domain tcyesno iPerformOnHandAvailableCheck, domain tcyesno iSkipBlockedInventory, domain tcyesno iBlockPlanningOfOrdersWithShortages, domain tsmdm.scin iInventoryScope, domain tcyesno iUpdatePlannedDeliveryTimeOfMaterialLines, domain tcyesno iUpdateActivitiesAndOrdersWithLatestPlannedMaterialLine, domain tcyesno iSynchronizeMaterialsWithLatestPlannedMaterialLine, domain tccotp iPurchaseOrderTypeNormalDelivery, domain tcseri iPurchaseOrderSeriesNormalDelivery, domain tccotp iPurchaseOrderTypeDirectDelivery, domain tcseri iPurchaseOrderSeriesDirectDelivery, domain tccotp iServicesProcurementOrderType, domain tcseri iServicesProcurementOrderSeries, ref domain oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this function to plan one specific Service Order Activity (tsoc210 record). If planning does not succeed for whatever reason, this function will return a value unequal zero and the reason why the plan action was not successful is present in the oExceptionMessage and oExceptionID. If planning succeeds then the value zero is returned. Even if a value zero is returned, which means that the given activity changed its status to Planned, the oExceptionMessage can be filled (and even more information can be present in the oExceptionID), because the system generates information messages about the flow of the process. This interface has the same options as when the user would select one specific service order activity in session

	'Service Order Activities' (tssoc2110m000) and use the specific option Plan Activity, which starts session Service Order Resource Planning (tssoc2260m000). All the options which are available in the planning session are also available as input arguments for this interface function. It can be useful to look at the Help for session tssoc2260m000. Note that this interface behaves exactly the same as the planning session, which means that the same checks are executed for the input process options as is being done in the planning session. If a certain combination of input arguments is not allowed, the user will be notified. Pre: db.retry.point must have been set. Post: Blocking status of the service order could have been changed. Status of Service Order Activity could have been changed from Free to Planned. Commit/abort the transaction. Input: iServiceOrder Service Order Mandatory. iActivityLine Service Order Activity Mandatory. iReplan If Replan is Yes, Service Order Activity with status Free, or Planned, are planned or replanned. If Replan is No, only Service Order Activity with status Free will be planned. Mandatory. iPerformATPCheck Indicator whether the ATP check is performed. Note: value Yes will be ignored and converted to No if material availability is not present in the Service Order Parameters (or site specific record, if the concept of Sites has been implemented). Mandatory. iPerformPlannedAvailableCheck Indicator whether the Planned Inventory check is performed. Note: value Yes will be ignored and converted to No if material availability is not present in the Service Order Parameters (or site specific record, if the concept of Sites has been implemented). Mandatory. iPerformOnHandAvailableCheck Indicator whether the On Hand Inventory check is performed. Note: value Yes will be ignored and converted to No if material availability is not present in the Service Order Parameters (or site specific record, if the concept of Sites has been implemented). Mandatory. iSkipBlockedInventory Indicator whether Blocked Inventory has to be considered during the various material availability checks. Note: value Yes only allowed if material availability is present in the Service Order Parameters (or site specific record, if the concept of Sites has been implemented). Furthermore, at least one of the input arguments iPerformPlannedAvailableCheck or iPerformOnHandAvailableCheck should have the value Yes.

	Mandatory. iBlockPlanningOfOrdersWithShortages If set to Yes, then the planning of the activity will not be successful if one of the material availability checks reports a shortage. Note: value Yes only allowed if at least one of the material availability checks is being executed, which means that either iPerformATPCheck, iPerformPlannedAvailableCheck or iPerformOnHandAvailableCheck should have the value Yes. Mandatory. iInventoryScope Indicator if during the various material availability checks only the current warehouse should be considered or the whole warehouse cluster. Possible values: Current Warehouse Only Checks if the material is available in the warehouse defined on the Material Line. All Warehouses in Planning Cluster Checks if the material is available in one of the warehouses in the same cluster as the warehouse defined on the Material Line. Mandatory. iUpdatePlannedDeliveryTimeOfMaterialLines Indicator whether the material availability checks should update the planned delivery time of related material lines. Note: value Yes only allowed if at least one of the input arguments iPerformATPCheck or iPerformPlannedAvailableCheck has the value Yes. Mandatory. iUpdateActivitiesAndOrdersWithLatestPlannedMaterialLine Indicator whether the material availability checks should update the activity and order with the time of the latest planned material line. Note: value Yes only allowed if at least one of the input arguments iPerformATPCheck or iPerformPlannedAvailableCheck has the value Yes. Value Yes only allowed if the input argument iUpdatePlannedDeliveryTimeOfMaterialLines is Yes. Mandatory. iSynchronizeMaterialsWithLatestPlannedMaterialLine Indicator whether the material availability checks should synchronize material lines with the latest planned material line. Note: value Yes only allowed if at least one of the input arguments iPerformATPCheck or iPerformPlannedAvailableCheck has the value Yes. Value Yes only allowed if the input argument iUpdatePlannedDeliveryTimeOfMaterialLines is Yes. Mandatory. iPurchaseOrderTypeNormalDelivery If the delivery type of the required material for a service order activity is By Purchase Order, this Purchase Order Type is used to create its purchase order. Not mandatory. iPurchaseOrderSeriesNormalDelivery The number group used to assign series numbers to the generated purchase orders. Not mandatory.

	iPurchaseOrderTypeDirectDelivery If the delivery type of the required material for a service order activity is Supplier Direct Delivery, this Purchase Order Type is used to create its purchase order. Not mandatory. iPurchaseOrderSeriesDirectDelivery The number group used to assign series numbers to the generated purchase orders. Not mandatory. iServicesProcurementOrderType Services Procurement Order Type Not mandatory. iServicesProcurementOrderSeries Services Procurement Order Series Not mandatory. Output: oExceptionMessage The last message if the return value is not equal to 0. If the return value equals 0, it can contain the last information message about the process. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error or process information. Use the functions in Exception to get all relevant information. Note that if the return value of this function is unequal zero, then we are dealing with an error situation and the status of the service order activity was not changed to Planned. If the return value = 0 (so not an error), the oExceptionID can still contain information about the process. Return: 0 - No Error and the status of the given Service Order Activity changed to Planned. <> 0 - The status of the Service Order Activity could not be changed to Planned.

ServiceOrderActivity.ProcessReturnDeliveries

DLL:	tsxtsocapi
Syntax:	<pre>long ServiceOrderActivity.ProcessReturnDeliveries(domain tcorno iServiceOrder, domain tsmdm.acln iActivityLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this function to start the process of creating Return Deliveries for service order material lines linked to the service order activity. If all material lines related to the service order have to be done, then iActivityLine must be 0. If only for a specific activity line number this process has to be executed then specify the iActivityLine with an existing activity line number.

	Note that because of the possibility that for return warehouse orders, some warehouse activities might have been set to automatic, the system will commit the database transactions in this function. This means that it is not necessary to set a <code>db.retry.point()</code> before calling this function and abort/commit after this function, because that is already handled within this function. Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post: If warehouse procedures have been set to automatic, then these are executed. Input: <code>iServiceOrder</code> - Service Order: Mandatory <code>iActivityLine</code> - Service Order Activity: Not mandatory Output: <code>oExceptionMessage</code> - The last message if the return value is not equal to 0. If more than one message is given, these are present in the <code>oExceptionID</code> <code>oExceptionID</code> - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - No Error <code><> 0</code> - Error

ServiceOrderActivity.Release

DLL:	tsxtsocapi
Syntax:	<pre> long ServiceOrderActivity.Release(const domain tcorno iServiceOrder fixed, const domain tcpono iActivityLine, const domain tcyesno iPerformATPCheck, const domain tcyesno iPerformPlannedAvailableCheck, const domain tcyesno iPerformOnHandAvailableCheck, const domain tcyesno iSkipBlockedInventory, const domain tsmdm.scin iInventoryScope, const domain tcyesno iBlockRelease, const domain tcyesno iUpdatePlannedDeliveryTimeOfMaterialLines, const domain tcyesno iUpdateActivitiesAndOrdersWithLatestPlannedMaterialLine, const domain tcyesno iSynchronizeMaterialsWithLatestPlannedMaterialLine, const domain tcyesno iUpdateActualRentalLocationAddress, const boolean iUseActivityLocationAddressForRentalAddress, const domain tccom.cadr iActualRentalLocationAddress fixed, const domain tcyesno iIncludeOrdersWithStatusFree, const domain tcyesno iCheckCapacityAvailability, const domain tcyesno iCheckProjectStatus, const domain tcyesno iCheckServiceKitAllocation, const domain tcyesno iCheckSkills, const domain tcyesno iCheckAssignments, const domain tcyesno iCopyEstimatesToActualsForOtherCosts, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>

Usage:	Expl: Use this function to release one specific service order activity (tssoc210 record). If releasing does not succeed for whatever reason, this function will return a value unequal zero and the reason why the release action was not successful is present in the oExceptionMessage and oExceptionID. If releasing succeeds then the value zero is returned. Even if a value zero is returned, which means that the given activity changed its status to 'Released', the oExceptionMessage can be filled (and even more information can be present in the oExceptionID), because the system generates information messages about the flow of the process (if the activity is first planned, this is logged as an information message etc.). This interface has exactly the same options as when the user would select one specific service order activity in session 'Service Order Activities' (tssoc2110m000) and would use the specific option 'Release Activity', which starts the session tsspc2200m000. All the options which are available in the release session are also available as input arguments for this interface function. It can be useful to look at the help for session tssoc2200m000. Note that this interface behaves exactly the same as the release session, which means that the same checks are executed for the input process options as is being done in the release session. If a certain combination of input arguments is not allowed, the user will be notified. If the option 'iIncludeOrdersWithStatusFree' is set to 'yes' and the user has selected an activity with status 'Free', then the system will first try to Plan this activity. If that succeeds the status of the activity (and order header if necessary) is changed to 'Planned' and committed in the database. If the change to 'Planned' succeeds or the service order activity already had the status 'Planned', the system will try to release this activity. If that succeeds and warehouse orders have been created, and some of the warehouse orders activities have been set to automatic, the system will try to execute those warehouse activities. Note that because of the blocking check on the service order and the automatic processing of warehouse orders, the system will commit the database transactions in this function. This means that it is not necessary to set a db.retry.point() before calling this function and abort/commit after this function, because that is already handled within this function. Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post: Blocking status of the service order could have been changed. Status of service order activity could have been changed from 'free' to 'Planned', if releasing itself did not succeed, but the activity had initial status 'Planned' and planning did succeed (this can only happen if also the input argument 'iIncludeOrdersWithStatusFree' has been set to 'Yes'). The status of the service order could have been changed from 'Free' or 'Planned' to 'Released'. If warehouse procedures have been set to automatic, then these are also executed. Input: iServiceOrder - The service order.

	Mandatory input. iActivityLine - The activity line. Mandatory input. iPerformATPCheck - Indicator whether the ATP check has to be executed. Note: value 'Yes' will be ignored and converted to 'No' if material availability is not present in the service order parameters (or site specific record, if the concept of 'sites' is implemented). Furthermore the iIncludeOrdersWithStatus Free should also have the value 'Yes'. Mandatory input. iPerformPlannedAvailableCheck - Indicator whether the ATP check has to be executed. Note: value 'Yes' will be ignored and converted to 'No' if material availability is not present in the service order parameters (or site specific record, if the concept of 'sites' is implemented). Mandatory input. iPerformOnHandAvailableCheck - Indicator whether the ATP check has to be executed. Note: value 'Yes' will be ignored and converted to 'No' if material availability is not present in the service order parameters (or site specific record, if the concept of 'sites' is implemented). Mandatory input. iSkipBlockedInventory - Indicator whether Blocked Inventory has to be considered during the various material availability checks. At least one of the input arguments iPerformPlannedAvailableCheck or iPerformOnHandAvailableCheck should have the value 'Yes'. Mandatory input. iInventoryScope - Indicator if during the various material availability checks only the current warehouse should be considered or the whole warehouse cluster. Mandatory input. iBlockRelease - If set to 'Yes', then the releasing of the activity will not be successful if one of the material availability checks reports a shortage. Note: value 'Yes' only allowed if at least one of the material availability checks is being executed, which means that either iPerformATPCheck, iPerformPlannedAvailableCheck or iPerformOnHandAvailableCheck should have the value 'Yes'. Mandatory input. iUpdatePlannedDeliveryTimeOfMaterialLines - Indicator whether the material availability checks should update the planned delivery time of related material lines. Note: value 'Yes' only allowed if at least one of the input arguments iPerformATPCheck or iPerformPlannedAvailableCheck has the value 'Yes'. Mandatory input. iUpdateActivitiesAndOrdersWithLatestPlannedMaterialLine - Indicator whether the material availability checks should update the activities and order with the time of the latest planned material line.

	Note: value 'Yes' only allowed if at least one of the input arguments iPerformATPCheck or iPerformPlannedAvailableCheck has the value 'Yes'. Value 'Yes' only allowed if the input argument iUpdatePlannedDeliveryTimeOfMaterialLines is 'Yes'. Mandatory input. iSynchronizeMaterialsWithLatestPlannedMaterialLine - Indicator whether the material availability checks should synchronize material lines with the latest planned material line. Note: value 'Yes' only allowed if at least one of the input arguments iPerformATPCheck or iPerformPlannedAvailableCheck has the value 'Yes'. Value 'Yes' only allowed if the input argument iUpdatePlannedDeliveryTimeOfMaterialLines is 'Yes'. Mandatory input. iUpdateActualRentalLocationAddress - Indicator whether the address of the rental equipment has to be updated. Note: value 'Yes' only allowed if the current service order activity is 'Rentable'. Mandatory input. iUseActivityLocationAddressForRentalAddress - Boolean indicator whether the address of the rental equipment should be updated using the location address of the activity or the iActualRentalLocationAddress. Note: value 'True' only allowed if the input argument iUpdateActualRentalLocationAddress has the value 'Yes'. iActualRentalLocationAddress - The address code which will be used to update the actual location of the rental equipment. Note: only allowed to be filled if iUpdateActualRentalLocationAddress is 'Yes' and iUseActivityLocationAddressforRentalAddress is 'False'. It should be filled if iUpdateActualRentalLocationAddress is 'Yes' and iUseActivityLocationAddressforRentalAddress is 'True'. iIncludeOrdersWithStatusFree - Indicator whether also an activity with status 'Free' is allowed to be released. Mandatory input. iCheckCapacityAvailability - Indicator whether the system should check during releasing whether there is enough capacity on the specified service order activity. If set to 'Yes' and there is not enough capacity, the release of the activity will not be executed. Mandatory input. iCheckProjectStatus - Indicator whether the system should check during releasing whether the status of a related project allows releasing the order. If set to 'Yes' and the status of the project is no ok, the release of the activity will not be executed. Mandatory input. iCheckServiceKitAllocation - Indicator whether the system should check during releasing whether the service kit (if present) can be allocated. If set to 'Yes' and the allocation for the service kit cannot be done, the release of the activity will not be executed.

	Mandatory input. iCheckSkills - Indicator whether the system should check during releasing whether the correct skills are available to execute the service order activity. If set to 'Yes' and not the correct skills are available, the release of the activity will not be executed. Mandatory input. iCheckAssignments - Indicator whether the system should check during releasing that the activity has at least one assignment. If set to 'Yes' and no assignment is related to the given activity, the release of the activity will not be executed. Note: the value 'No' will be converted to 'Yes' if assignments are mandatory in the service order parameters. Mandatory input. iCopyEstimatesToActualsForOtherCosts.yn - Indicator whether the system should take over the estimates to the actual values for Other Costs (tssoc240) during the release process. Mandatory input. Output: oExceptionMessage The last message if the return value unequal 0. If the return value equals 0, it can contain the last information message about the process. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error or process information. Use the functions in Exception to get all relevant information. Note that if the return value of this function is unequal zero, then we are dealing with an error situation and the status of the service order activity was not changed to 'Released'. If the return value = 0 (so not an error), the oExceptionID can still contain information about the process. Return: 0 - No Error and the status of the given service order activity changed to 'Released'. <> 0 - The status of the service order activity could not be changed to 'Released'.

ServiceOrderActivity.StartMultiMain

DLL:	tsextsocapi
Syntax:	<pre>long ServiceOrderActivity.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcorno iServiceOrder, domain tsmdm.acln iActivityLine,</pre>

	ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function starts the Multi Main Table session Service Order Activity (ts soc2110m100). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iQueryExtend A specific query to be used when zooming to this session. iServiceOrder Service Order iActivityLine Service Order Activity line number Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

Public Interfaces for ServiceOrderMaterialCosts

The following functions are available:

[ServiceOrderMaterialCosts.ProcessReturnDeliveries](#)

ServiceOrderMaterialCosts.ProcessReturnDeliveries

DLL:	tsxtsocapi
	This function is available from KB2262498 .
Syntax:	<pre>long ServiceOrderMaterialCosts.ProcessReturnDeliveries (domain tcorno iServiceOrder, domain tcpono iMaterialLine, ref domain tcmcs.s999m oExceptionMessage mb,</pre>

	ref	long	oExceptionID)
Usage:	Expl:	Use this function to start the process of creating a Return Delivery for a Service Order Material line.	
		Note that because of the possibility that for return warehouse orders, some warehouse activities might have been set to automatic, the system will commit the database transactions in this function. This means that it is not necessary to set a db.retry.point() before calling this function and abort/commit after this function, because that is already handled within this function.	
	Pre:	No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed).	
	Post:	If warehouse procedures have been set to automatic, then these are executed.	
	Input:	iServiceOrder	- Service Order: Mandatory
		iMaterialLine	- Service Order Material Line: Mandatory
	Output:	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
		oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
	Return:	0	- No Error
		<> 0	- Error

Public Interfaces for ServiceOrderCostLine

The following functions are available:

[ServiceOrderCostLine.Cost](#)

ServiceOrderCostLine.Cost

DLL:	tsextsocapi		
Syntax:	<pre>long ServiceOrderCostLine.Cost(const domain tcorno iServiceOrder fixed, const domain tsmdm.cotp iCostType, const domain tcpono iCostLine, const domain tcyesno iIncludeReleasedOrdersAndActivities, const domain tcyesno iSetServiceOrderActivityToCostedWhenAllCostLinesAreCosted,</pre>		

	<pre> const domain tcyesno iSetServiceOrderToCostedWhenAllActivitiesAreCosted, const domain tcyesno iCheckUnconsumedItems, const domain tcyesno iRemoveSubsequentDeliveryQuantity, const domain tcyesno iInvoiceFreightLinesinFreightManagement, const domain tcyesno iIgnoreUnapprovedInvoices, const domain tcyesno iCostZeroQuantityForOtherCosts, const domain tcyesno iDoNotCostIfOpenInspectionsExist, const domain tcyesno iSetOpenInspectionsToNotMeasured, const domain tcyesno iDeleteOpenInspections, const domain tccur iCurrency fixed, const domain tcamnt iMaximumLimitForInvoiceAmount, const domain tcamnt iMaximumLimitForOtherAmount, const domain tcprcg iLowerMargin, const domain tcprcg iUpperMargin, const domain tcsli.stat iInvoiceLineStatus, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this function to cost one specific service order cost line (either a material line from tssoc220, a labor line from tssoc230 or another cost line from tssoc240). See functionality in the costing session (tssoc2290m000) when the Selection Criteria Level is set to 'Cost Line'. All options available in the costing session tssoc2290m000 are available as input arguments. The labels of the options from the session tssoc2290m000 are taken over to the input variables, so that also the help from this session can be used to get more information. If setting the cost line to status 'Costed' does not succeed for whatever reason, this function will return a value unequal zero and the reason why the costing was not successful is present in the oExceptionMessage and oExceptionID. If costing succeeds then the value zero is returned. This function does its own database handling, so there is no need to specify a db.retry.point() before calling this function and to abort/commit the transactions afterwards. Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post: Input: iServiceOrder - The service order. Mandatory input. iCostType - The cost Type. Allowed values are: tsmdm.cotp.material - To cost a material line from tssoc220. tsmdm.cotp.labor - To cost a labor line from tssoc230. And the following cost types are from the other costs tssoc240. tsmdm.cotp.tool - To cost a line with a tool. tsmdm.cotp.travel- To cost a travel line. tsmdm.cotp.subcon- To cost a subcontracting line. tsmdm.cotp.helpdesk - To cost a helpdesk line. tsmdm.cotp.other - To cost another cost line.

	tsmdm.cotp.freight - To cost a freight line. tsmdm.cotp.quotinv - To cost the quote invoice line. tsmdm.cotp.rental - To cost a rental other cost line. Mandatory input. iCostLine - The cost line number - This is the line number from either the material lines (tssoc220), the labor lines (tssoc230) or the other cost lines (tssoc240). Mandatory input. iIncludeReleasedOrdersAndActivities - If this option is set to 'yes', then if the cost line is either related to an activity with status 'Released' or directly to the order header and the order header has status 'Released', then the cost line is selected for costing. If this option is set to 'no', then only if the cost line is related to a 'Completed' activity or order, the costing is executed. Mandatory input. iSetServiceOrderActivityToCostedWhenAllCostLinesAreCosted - If this option is set to 'yes' and by costing the current cost line, all cost lines related to the activity to which the cost line is linked, are set to costed, also the activity is set to costed. Mandatory input. iSetServiceOrderToCostedWhenAllActivitiesAreCosted - If this option is set to 'yes', the system will also set the service order header to 'Costed' if all related activities are 'Costed' and all related cost lines are 'Costed'. Mandatory input. iCheckUnconsumedItems - If a material line is being costed, and this parameter is set to 'yes', then if not all unconsumed items have been returned, the costing will not proceed. If set to 'no', then the system will continue the costing even if there is still unconsumed quantity available. Mandatory input. iRemoveSubsequentDeliveryQuantity - If a material line is being costed and the subsequent delivery quantity is unequal zero, then with this option set to 'yes', the system will automatically set the subsequent delivery quantity on the material line to zero. Mandatory input. InvoiceFreightLinesinFreightManagement - If this option is set to 'yes', are there are freight invoice lines in Freight Management which have not been sent to Central Invoicing, then these are automatically released to Central Invoicing. Mandatory input. iIgnoreUnapprovedInvoices - If this option is set to 'yes' and if a subcontracting other cost line is being costed and not all purchase invoices related to the related purchase order line have been approved, then costing will still proceed. Mandatory input.

	iCostZeroQuantityForOtherCosts - If this option is set to 'no' and if another cost line is being costed (tssoc240) and the actual quantity is still zero, then costing will not proceed. Note that a Total Travel Line is excluded from this check, because the actual quantity for a Total Travel Line is always zero. Mandatory input. iDoNotCostIfOpenInspectionsExist - If this option is set to 'yes' and if the related activity can also be set to 'Costed' and open inspections exist (tscfg300) related to that activity, then costing will not succeed. If also the related order header can be set to 'Costed' and open inspections exist related to the order, then costing will not succeed. Mandatory input. iSetOpenInspectionsToNotMeasured - If the input argument iDoNotCostIfOpenExpectationsExist is set to 'no', this input can be set to 'yes'. If set to 'yes', then any open inspection will get the status 'Not Measured'. Mandatory input. iDeleteOpenInspections - If the input argument iDoNotCostIfOpenExpectationsExist is set to 'no' and the input argument iSetOpenInspectionsToNotMeasured is set to 'no', this input argument can be set to 'yes'. If set to 'yes', then any open inspection will be deleted. Mandatory input. iCurrency - The currency in which the input arguments iMaximumLimitForInvoiceAmount and iMaximumLimitForOtherAmount are expressed. Mandatory input. iMaximumLimitForInvoiceAmount - The maximum net invoice amount of the cost line which is allowed to be costed. iMaximumLimitForOtherAmount - The maximum other amount (is actually the goodwill amount) which is allowed to be costed. iLowerMargin - If margin control is applicable in the service order parameters (or in the settings by service office if the concept of sites has been activated), then this is the lowest margin for which it is allowed to cost the given cost line. iUpperMargin - If margin control is applicable in the service order parameters (or in the settings by service office if the concept of sites has been activated), then this is the highest margin for which it is allowed to cost the given cost line. iInvoiceLineStatus - The line status with which the billable line in Central Invoicing is created. Allowed values are: tcsli.stat.on.hold tcsli.stat.confirmed tcsli.stat.not.appl If the status is set to 'tcsli.stat.not.appl' then this will mean that the system will default it again from either the service order parameters or from the

	settings per office, if the concept of sites has been activated. Mandatory input. Output: oExceptionMessage The last message if the return value unequal 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error or process information. Use the functions in Exception to get all relevant information. Note that if the return value of this function is unequal zero, then the status of cost line is not changed to 'Costed'. Return: 0 - No Error and the status of the given service order cost line changed to 'Costed'. <> 0 - The status of the service order cost line could not be changed to 'Costed'.

Public Interfaces for ServiceQuote

The following functions are available:

[ServiceQuote.Process](#)

ServiceQuote.Process

DLL:	tsxteppapi
Syntax:	<pre> long ServiceQuote.Process(domain tcorno iQuote, domain tcpono iQuoteRevision, domain tcseri iServiceOrderSeries, domain tcseri iMaintenanceSalesOrderSeries, domain tcyesno iATPCheck, domain tcyesno iCheckPlannedInventory, domain tcyesno iCheckInventoryOnHand, domain tcyesno iSkipBlockedInventory, domain tcyesno iBlockInCaseOfInventoryShortage, domain tsmdm.scin iScopeOfInventoryCheck, domain tcyesno iUpdatePlannedDeliveryTimeMaterialLine, domain tcyesno iUpdateQuoteLinesWithLastPlannedMaterialLine, domain tcyesno iSynchronizeQuoteLinesWithLastPlannedMaterialLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>

Usage:	Expl: This function handles the processing of a service quote to a service order or maintenance sales order. Based on the given options/settings for processing the quote, additional checks related to inventory availability are done. Note: - If a part maintenance line is created, no work order will be created, regardless the value of the related maintenance sales control parameter. Pre : a db.retry.point() must have been specified. Post : an abort.transaction() or commit.transaction() must be executed. Method: 'WorkOrder.CreateMSO' of BDE 'WorkOrder' can be used to create the related work order. Input: - iQuote - Service Quote (mandatory) - iQuoteRevision - Service Quote Revision (optional) - iServiceOrderSeries - Service Order Series (not mandatory) - iMaintenanceSalesOrderSeries - Maintenance Sales Order Series (not mandatory) - iATPCheck - Perform ATP check (mandatory Yes/No) - iCheckPlannedInventory - Check Planned Available Inventory (mandatory Yes/No) - iCheckInventoryOnHand - Check On Hand Inventory (mandatory Yes/No) - iSkipBlockedInventory - Skip Blocked Inventory (mandatory Yes/No) - iBlockInCaseOfInventoryShortage - Block Planning or Releasing of Orders with Shortages (mandatory Yes/No) - iScopeOfInventoryCheck - Scope of Inventory Check (mandatory) Possible values: - current warehouse only - all warehouses in planning cluster - iUpdatePlannedDeliveryTimeMaterialLine - Update Planned Delivery Time of Material Lines (mandatory Yes/No) - iUpdateQuoteLinesWithLastPlannedMaterialLine - Update Quote Lines with Latest Planned Material Line (mandatory Yes/No) - iSynchronizeQuoteLinesWithLastPlannedMaterialLine - Synchronize Quote lines with Latest Planned Material Line (mandatory Yes/No) Output: - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information.

	Return: 0 - process succesful <> 0 - Error during process occurred

Public Interfaces for MaintenanceSalesOrder

The following functions are available:

[MaintenanceSalesOrder.Cost](#)

[MaintenanceSalesOrder.GenerateSerializedItem](#)

MaintenanceSalesOrder.Cost

DLL:	tsxtmncapi									
	This function is available from KB3569267 .									
Syntax:	<pre>long MaintenanceSalesOrder.Cost(domain tcorno iMaintenanceSalesOrder, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>									
Usage:	Expl.: This function sets the Maintenance Sales Order, and all related Part Lines and Coverage Lines, to Costed. Pre: db.retry.point must have been set. Call ProcessingOptionSet.Create() to obtain iProcessingOptionSet. Post: Commit/abort the transaction. Delete the option set by calling ProcessingOptionSet.Delete(). Input: iMaintenanceSalesOrder - Maintenance Sales Order; mandatory iProcessingOptionSet Processing Option Set: a processing option set number referring to a processing option set containing at least one valid option. Mandatory. <table><tr><td>NAME</td><td>TYPE</td><td>DEFAULT</td></tr><tr><td colspan="3">=====</td></tr><tr><td>InvoiceFreightLines</td><td>domain tcyesno</td><td>yes</td></tr></table> When there are Part Lines with related Freight Lines,	NAME	TYPE	DEFAULT	=====			InvoiceFreightLines	domain tcyesno	yes
NAME	TYPE	DEFAULT								
=====										
InvoiceFreightLines	domain tcyesno	yes								

	these Freight Lines are sent to Invoicing. IgnoreUnapprovedPurchaseInvoices domain tcyesno no If this argument is set to Yes, the maintenance sales order is costed, even when there are unapproved purchase invoices present. Output : ExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return : 0 - No error; however, error messages can have been set. <> 0 - An error occurred

MaintenanceSalesOrder.GenerateSerializedItem

DLL:	tsxtmscapi This function is available from KB2261704.
Syntax:	<pre>long MaintenanceSalesOrder.GenerateSerializedItem(domain tcorno iMaintenanceSalesOrder, domain tcpono iLine, domain tcibd.sern iSerialNumber, domain tcyesno iLinkToInstallationGroup, domain tcyesno iUpdateSerialOnMaintenanceSalesOrder, ref domain tcibd.sern oGeneratedSerialNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function generates a serialized item based on the data of the given Maintenance Sales Order (Line). A new serial will be generated for the item defined on the Maintenance Sales Order or, when iLine is filled, on the Maintenance Sales Order Item Line. The sold-to business partner defined on the Maintenance Sales Order will be set as owner of the serialized item. Optionally the generated serialized item can be linked to the installation group defined on the Maintenance Sales Order (Line). Optionally the Maintenance Sales Order (Line) can be updated with the generated serial number. Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post: This function does its own database handling, so there is no need to specify a db.retry.point() before calling this function and to abort/commit the transactions afterwards.

	Input: iMaintenanceSalesOrder The order number based on which a new serialized item must be generated. (mandatory) iLine The line number based on which a new serialized item must be generated. (optional) iSerialNumber The Serial Number to be used for generating a serialized item. When not set, a new serial number will be generated based on a predefined mask. (optional) iLinkToInstallationGroup When an Installation Group is defined on the Maintenance Sales Order (Line) the generated serialized item can be added as installation in this group. Note that this is controlled by Configuration Management (CFG) parameter 'Link Generated Serialized Item to Installation Group'. When this parameter is not set or no installation group is defined on the Maintenance Sales Order (Line), the generated serialized item will not be linked to the installation group and the value of iLinkToInstallationGroup will be ignored. (mandatory) iUpdateSerialOnMaintenanceSalesOrder Controls if the serial number must be set on the Maintenance Sales Order (Line) based on which the serialized item is generated. - yes: The serial number on the order/line is updated with the generated serial number. - no: Only a serialized item is generated. (mandatory) Output: oGeneratedSerialNumber The serial number of the generated serialized item. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Serialized Item generated succesfull and (optionally) updated on the Maintenance Sales Order (Line). <> 0 - Error during generating serialized item occurred When oGeneratedSerialNumber is filled, the serialized item is generated successfully but the update of the serial number on the Maintenance Sales Order (Line) failed.

Public Interfaces for MaintenanceSalesOrderLine

The following functions are available:

[MaintenanceSalesOrderLine.Cost](#)

[MaintenanceSalesOrderLine.ItemReceipt](#)

[MaintenanceSalesOrderLine.Release](#)

MaintenanceSalesOrderLine.Cost

DLL:	tsextmscapi														
	This function is available from KB3569267 .														
Syntax:	<pre>long MaintenanceSalesOrderLine.Cost (domain tcorno iMaintenanceSalesOrder, domain tcpono iMaintenanceSalesOrderLine, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>														
Usage:	Expl.: This function sets a Maintenance Sales Order Line, and related Coverage Lines, to Costed. When it is the last line, the Maintenance Sales Order can optionally be set to Costed. Pre: db.retry.point must have been set. Call ProcessingOptionSet.Create() to obtain iProcessingOptionSet. Post: Commit/abort the transaction. Delete the option set by calling ProcessingOptionSet.Delete(). Input: iMaintenanceSalesOrder - Maintenance Sales Order; mandatory iMaintenanceSalesOrderLine - Maintenance Sales Order Line; mandatory iProcessingOptionSet Processing Option Set: a processing option set number referring to a processing option set containing at least one valid option. Mandatory. <table><thead><tr><th>NAME</th><th>TYPE</th><th>DEFAULT</th></tr></thead><tbody><tr><td colspan="3">=====</td></tr><tr><td>SetOrderToCosted</td><td>domain tcyesno</td><td>yes</td></tr></tbody></table> When SetOrderToCosted is Yes, the Maintenance Sales Order is set to Costed when all lines have been costed. Otherwise, the order status remains unchanged. InvoiceFreightLines <table><tbody><tr><td></td><td>domain tcyesno</td><td>yes</td></tr></tbody></table> When there are Part Lines with related Freight Lines,			NAME	TYPE	DEFAULT	=====			SetOrderToCosted	domain tcyesno	yes		domain tcyesno	yes
NAME	TYPE	DEFAULT													
=====															
SetOrderToCosted	domain tcyesno	yes													
	domain tcyesno	yes													

	these Freight Lines are sent to Invoicing. IgnoreUnapprovedPurchaseInvoices domain tcyesno no If this argument is set to Yes, the Maintenance Sales Order Line is costed, even when there are unapproved Purchase Invoices present. Output : ExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return : 0 - No error; however, error messages can have been set. <> 0 - An error occurred

MaintenanceSalesOrderLine.ItemReceipt

DLL:	tsxtmscapi
	This function is available from KB3636915 .
Syntax:	<pre>long MaintenanceSalesOrderLine.ItemReceipt (domain tcorno iMaintenanceSalesOrder, domain tcpono iMaintenanceSalesOrderLine, domain tsmdm.qmat iReceivedQuantity, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this function to register the Received Quantity in the Receipt Location of a released Part Maintenance Line. On registering, the Received Quantity is updated, including the Actual Receipt Time. The line status is set to In Process. Pre: db.retry.point must have been set. Post: Commit/abort the transaction. Input: iMaintenanceSalesOrder The maintenance sales order number Mandatory input. iMaintenanceSalesOrderLine The maintenance sales order part line for which the actual received quantity must be registered. Mandatory input. iReceivedQuantity The Received Quantity. Mandatory input when the Part Maintenance Line is for an anonymous item. When the Part Maintenance Line is for a serialized item, the received quantity is defaulted with 1.0. Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID.

	<code>oExceptionID</code> An ID that refers to the exception information. Use the functions in <code>Exception</code> to get all relevant information. Return: 0 - No Error <> 0 - Error situation

MaintenanceSalesOrderLine.Release

DLL:	<code>tsxtmscapi</code> This function is available from KB3567472.
Syntax:	<pre>long MaintenanceSalesOrderLine.Release(domain tcorno iMaintenanceSalesOrder, domain tcpono iMaintenanceSalesOrderLine, ref tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this function to release one specific maintenance sales order line (tsmscl110 record). See functionality when a maintenance sales order line is released with the form command 'Release' in sessions: - Maintenance Sales Order - Part Maintenance Lines (tsmscl110m100); - Maintenance Sales Order - Part Loan Lines (tsmscl110m200); - Maintenance Sales Order - Part Delivery Lines (tsmscl110m300); - Maintenance Sales Order - Part Receipt Lines (tsmscl110m400). If the complete action does not succeed, this function will return a value unequal zero. If completing succeeds then the value zero is returned. This function does its own database handling, so before calling this function the existing database transactions should either have been aborted or committed. This is necessary because the system will execute the blocking checks and if the order gets blocked, the blocking flag on header level is set and committed and the release action will fail. If return warehouse orders are created, then the system will at the end also process the warehouse activities which are set to automatic. Note: - The part lines of a maintenance sales order with the status 'Free' cannot be released. The release of the maintenance sales order line will be aborted. Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post: If the order gets blocked, the blocking flag is set on header level. If return warehouse orders are created and warehouse activities have been set to Automatic, these are executed. If the maintenance sales order of the line that is released has the status free when trying to release the maintenance sales order line, the maintenance sales order is also released.

	Input: iMaintenanceSalesOrder The maintenance sales order number for the maintenance sales order line that needs to be released. Mandatory input. iMaintenanceSalesOrderLine The maintenance sales order line that needs to be released. Mandatory input. Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - No Error and the status of the given maintenance sales order line changed to released. <> 0 - The status of the maintenance sales order line could not be changed to released.

Public Interfaces for WorkOrder

The following functions are available:

[WorkOrder.Cancel](#)

[WorkOrder.Close](#)

[WorkOrder.Complete](#)

[WorkOrder.DetermineStartOrFinishTime](#)

[WorkOrder.GenerateActivitiesFromMasterRouting](#)

[WorkOrder.GetActualRepairCosts](#)

[WorkOrder.Release](#)

[WorkOrder.StartDetail](#)

[WorkOrder.StartMultiMain](#)

[WorkOrder.StartOverview](#)

WorkOrder.Cancel

DLL:	tsextwcsapi
Syntax:	<pre> long WorkOrder.Cancel (domain tcorno iWorkOrder, domain tsmdm.acln iWorkOrderActivityLine, domain tsspc.stat iNewPlannedActivityStatus, domain tcyesno iSetLinkedCallToSolved, domain tcyesno iCancelPurchaseMaterialLines, domain tccdis iCancelReason, domain tsmdm.utct iCancelTime, domain tsmdm.text iCancelText, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function handles the cancellation of a work order or single activity. If iWorkOrderActivityLine is zero the whole work order will be cancelled. If iWorkOrderActivityLine is not zero the specific work order activity line will be cancelled. If the work order or activity has follow-up work orders, these will be cancelled as well. This function has the same options as session 'Cancel Work Order' (tswcs2260m000) when cancelling a work order or work order activity. These are options available as input field and conditional options asked by questions. - iNewPlannedActivityStatus In case the order or activity is originating from a planned activity this input argument can be used to set the status of the originating planned activity after cancelling. - iSetLinkedCallToSolved In case the order or activity is originating from a call and this argument has value 'Yes' the call status is set to 'Solved'. - iCancelPurchaseMaterialLines If purchase orders are created for work order material lines, this option can be used to cancel or delete these. When the purchase order status is Planned, the purchase order is deleted. If the purchase order is already processed, the link in the service order related orders is removed. Pre : a db.retry.point() must have been specified. Post : an abort.transaction() or commit.transaction() must be executed. Input: iWorkOrder Work Order (mandatory) iWorkOrderActivityLine Work Order Activity Line (optional) iNewPlannedActivityStatus (mandatory, Free/Released/Cancelled) In case the order or activity is originating from a planned activity this input argument can be used to set the status of the originating planned activity after cancelling. Allowed values are: - Free

	- Released - Cancelled iSetLinkedCallToSolved (mandatory, Yes/No) In case the order or activity is originating from a call and this argument has value 'Yes' the call status is set to 'Solved'. iCancelPurchaseMaterialLines (mandatory, Yes/No) If this option has value 'Yes' purchase orders created for work order material lines are cancelled or deleted. When the purchase order status is Planned, the purchase order is deleted. If the purchase order is already processed, the link in the service order related orders is removed. iCancelReason (mandatory) Must be of type 'Cancellation' and date effective when cancelling. iCancelTime (mandatory) The cancel time. iCancelText (optional) Cancellation text. Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Work order or activity set to cancelled succesfull <> 0 - Error during cancelling the work order or activity occurred

WorkOrder.Close

DLL:	tsxtwcsapi
	This function is available from KB3610861 .
Syntax:	<pre>long WorkOrder.Close(domain tcorno iWorkOrder, long long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this function to close a Work Order. This function has the same options as reference form command "Close Order" in the Work Order Session tswcs2100m100. Pre: Call ProcessingOptionSet.Create() to obtain iProcessingOptionSet. A db.retry.point() must have been specified. Post: Delete the option set by calling ProcessingOptionSet.Delete(). An abort.transaction() or commit.transaction() must be

	executed.	
Input: iWorkOrder	Work Order Mandatory.	
iProcessingOptionSet	Processing Option Set: a processing option set number referring to a processing option set. Mandatory.	
NAME	TYPE	DEFAULT
=====		
IgnoreUnapprovedSubcontractedPurchaseOrders	domain tcyesno	tcyesno.no
	If this option is set to 'No', and the Work Order is internal and there are related subcontracted Purchase Orders that are not approved, then the Work Order is not closed.	
CloseAllOpenPhysicalBreakdownChanges	domain tcyesno	tcyesno.no
	If this option is set to 'Yes' and the Work Order is closed, then all related open Physical Breakdown Changes are closed.	
ActionForNonExecutedInspections	domain tscfg.upd.ista	tscfg.upd.ista.cancel
	If this option is set to 'Change Status', the status of every non-executed inspection is set to 'Not Measured'.	
	If this option is set to 'Delete Inspections', every non-executed inspection is deleted.	
	If this option is set to 'Cancel', and non-executed inspections are present, then closing the Work Order is canceled.	
Output: oExceptionMessage	The last message if any message is found. If more than one message is given, these are present in the oExceptionID.	
oExceptionID	An ID that refers to the exception information. Use the functions in Exception to get all relevant information.	
Return: 0	No errors occurred. Depending on the options in the processing option set, closing may still have been canceled.	
<> 0	Error(s) occurred.	

WorkOrder.Complete

DLL:	tsextwcsapi
Syntax:	<pre> long WorkOrder.Complete(domain tcorno iWorkOrder, domain tsmdm.acln iWorkOrderActivityLine, domain tsmdm.qmat iActualQuantity, domain tscfg.cfst iNewSerialStatus, domain tcyesno iCreateFollowupWorkOrderForRemainingQuantity, domain tcyesno iCopyEstimatesToActuals, domain tcyesno iContinueIfNoReplacementForRemovedOutgoingSubassembly, domain tcyesno iUpdateActualQuantityOnWorkOrderActivity, domain tcyesno iSetWorkOrderToCompleteWhenLastActivityIsCompleted, domain tscfg.upd.ista iActionOnPendingInspections, domain tcyesno iContinueCloseWorkOrderIfPurchaseOrderNotApprovedYet, domain tcyesno iClosePhysicalBreakdownChange, ref domain tcyesno oWorkOrderSignedOff, ref domain tcyesno oWorkOrderClosed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function handles setting a work order to completed. When iWorkOrderActivityLine is filled, this specific work order activity line will be set to completed. When all other work order activities are already completed and iSetWorkOrderToCompleteWhenLastActivityIsCompleted is set to yes, the work order header will be set to completed as well. When the work order is set to completed successfully, the warehouse procedure steps that are set to automatic execution, are started. Next, based on WCS parameters/office settings, the work order can be signed-off and closed automatically. Note: - When an Electronic Signature is required for completing a work order, this public interface can only be used when called in an LN UI component. Only in that case the signature request dialog can be started. - When interactive counter reading reset rules are defined for any of the work order activities, the session for resetting the counters is not started when this public interface is used. Resetting these counters can be done using LN UI. Pre: None. Post: This function sets a retry-point and will commit and/or abort the transaction. Input: iWorkOrder Work Order. (mandatory) iWorkOrderActivityLine Work Order Activity Line. (optional) iActualQuantity The actual maintained quantity to be set on the work order. Must be >= 0. An actual maintained quantity of zero implies that the serialized item is not repaired and will not be returned.

	Note that when setting an individual activity line to Completed, this quantity is not set on the work order activity line. When the actual quantity on the work order activity line is zero and iUpdateActualQuantityOnWorkOrderActivity is passed as 'yes', the planned quantity set on the work order will be set on the work order activity line. When passed as 'no' the actual quantity on the activity line is not updated. iNewSerialStatus If the main item on the work order is serialized controlled, this status will be assigned to it. (mandatory) If the work order item is serialized controlled and the actual quantity is set to 1, only the following statuses are valid: - active - working condition - defective - to be recycled An actual maintained quantity of zero, implies that the serialized item is not repaired and will not be returned. Its status is set to Removed. iCreateFollowupWorkOrderForRemainingQuantity Create Follow-up Work Order for remaining quantity. (mandatory Yes/No) Only applicable for non-serialized controlled items. iCopyEstimatesToActuals Copy Estimates to Actuals. (mandatory Yes/No) The required quantity of work order other resource lines and of material resource lines with delivery type 'From Service Inventory', is assigned to the actual quantity. iContinueIfNoReplacementForRemovedOutgoingSubassembly Continue if no replacement material line is defined for a removed Outgoing Subassembly. (mandatory Yes/No) iUpdateActualQuantityOnWorkOrderActivity Set the actual maintained quantity on the work order activity. When the activity is derived from a master routing and a previous activity is present with status Completed, Singed-off or Closed, the actual quantity of the previous activity is taken and set on the current work order activity line. When no master routing is used the planned quantity as set on the work order will be set on the current work order activity line. (mandatory Yes/No) iSetWorkOrderToCompleteWhenLastActivityIsCompleted Set the work order header status to complete when the given work order activity line is completed successfully and all other work order activity lines are already completed. (mandatory Yes/No) iActionOnPendingInspections When the work order is set to completed and there are open/pending inspections present, this variable can be used to indicate what to do with these inspections. Possible values are:

	- change status - inspection status is set to Not Measured - delete inspection - the inspections are deleted. - cancel - the inspections remain as is. iContinueCloseWorkOrderIfPurchaseOrderNotApprovedYet When the work order is set to completed and next automatically processed to status closed, based on WCS parameters/office settings, and a purchase invoice is present for a purchase order linked to the work order, which is not approved yet, this argument determines to continue closing the work order. (mandatory Yes/No) iClosePhysicalBreakdownChange When the work order is set to completed and next automatically processed to status closed, based on WCS parameters/office settings, open physical breakdown changes can be set to closed. (mandatory Yes/No) Output: oWorkOrderSignedOff Indicates if the work order is set to Signed-Off. oWorkOrderClosed Indicates if the work order is set to Closed. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - work order set to completed successful - Error during completing work order occurred

WorkOrder.DetermineStartOrFinishTime

DLL:	tsextwcsapi
Syntax:	<pre>long WorkOrder.DetermineStartOrFinishTime(domain tsmdm.cstp iServiceType, domain tcemno iAssignedEngineer, domain tcemno iPreferredEngineer, domain tccwoc iOperationsDepartment, domain tccwoc iWorkCenter, domain tsmdm.pldt iBaseTime, domain tcplcd iPlanningMethod, domain tcccp.ccal iCalendarCode, domain tcccp.ract iAvailabilityType, domain tsmdm.tmdu iDuration, ref domain tsmdm.pldt oTime, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function determines a finish time for a given start time or a start time for a given finish time: - When plan method is 'plan forward' the start time must be given and the finish time is calculated.

	- When plan method is 'plan backward' the finish time must be given and the start time is calculated. If passed empty, the calendar code used for planning is retrieved in the following way. When the parameter 'Use Engineer Calendar for Planning' is No, the calendar is searched in the following order: 1. Assigned Service Engineer 2. Preferred Service Engineer 3. Work Center 4. Operations Department 5. Enterprise Unit 6. Company This means that the assigned engineer calendar is retrieved first. When no assigned engineer calendar is available the calendar of the preferred engineer is retrieved. And so on. Finally the company calendar is retrieved, this calendar should always be available. When this parameter is no, the calendar hierarchy is: 1. Work Center 2. Service Department 3. Enterprise Unit 4. Company The Time Zone is searched in the same order as done for the Calendar. With the exception that no Time Zone is present on Enterprise Unit or Company level. If passed empty, the Availability Type is retrieved from the Service Type. Input: <table> <tr> <td>iServiceType</td><td>Service Type</td></tr> <tr> <td></td><td>Not Mandatory</td></tr> <tr> <td>iAssignedEngineer</td><td>Assigned Service Engineer</td></tr> <tr> <td></td><td>Not Mandatory</td></tr> <tr> <td>iPreferredEngineer</td><td>Preferred Service Engineer</td></tr> <tr> <td></td><td>Not Mandatory</td></tr> <tr> <td>iOperationsDepartment</td><td>Operations Department</td></tr> <tr> <td>iWorkCenter</td><td>Work Center</td></tr> <tr> <td></td><td>Either Operation Department or Work Center must be filled.</td></tr> <tr> <td>iBaseTime</td><td>Base Time, , dependent on the Planning Method this is the start or finish time.</td></tr> <tr> <td></td><td>Mandatory</td></tr> <tr> <td>iDuration</td><td>Duration, expressed in general service parameter 'Time Duration Unit' (tsmdm000.unttd)</td></tr> <tr> <td></td><td>Not Mandatory</td></tr> <tr> <td>iCalendarCode</td><td>Calendar</td></tr> <tr> <td></td><td>Not Mandatory</td></tr> <tr> <td>iAvailabilityType</td><td>Availability Type</td></tr> <tr> <td></td><td>Not Mandatory</td></tr> <tr> <td>iPlanMethod</td><td>Planning Method (forward/ backward)</td></tr> <tr> <td></td><td>Mandatory</td></tr> </table> Output: <table> <tr> <td>oTime</td><td>Result Time (planned start/ finish date)</td></tr> <tr> <td>oExceptionMessage</td><td>The last message if the return value is not equal to 0.</td></tr> </table>	iServiceType	Service Type		Not Mandatory	iAssignedEngineer	Assigned Service Engineer		Not Mandatory	iPreferredEngineer	Preferred Service Engineer		Not Mandatory	iOperationsDepartment	Operations Department	iWorkCenter	Work Center		Either Operation Department or Work Center must be filled.	iBaseTime	Base Time, , dependent on the Planning Method this is the start or finish time.		Mandatory	iDuration	Duration, expressed in general service parameter 'Time Duration Unit' (tsmdm000.unttd)		Not Mandatory	iCalendarCode	Calendar		Not Mandatory	iAvailabilityType	Availability Type		Not Mandatory	iPlanMethod	Planning Method (forward/ backward)		Mandatory	oTime	Result Time (planned start/ finish date)	oExceptionMessage	The last message if the return value is not equal to 0.
iServiceType	Service Type																																										
	Not Mandatory																																										
iAssignedEngineer	Assigned Service Engineer																																										
	Not Mandatory																																										
iPreferredEngineer	Preferred Service Engineer																																										
	Not Mandatory																																										
iOperationsDepartment	Operations Department																																										
iWorkCenter	Work Center																																										
	Either Operation Department or Work Center must be filled.																																										
iBaseTime	Base Time, , dependent on the Planning Method this is the start or finish time.																																										
	Mandatory																																										
iDuration	Duration, expressed in general service parameter 'Time Duration Unit' (tsmdm000.unttd)																																										
	Not Mandatory																																										
iCalendarCode	Calendar																																										
	Not Mandatory																																										
iAvailabilityType	Availability Type																																										
	Not Mandatory																																										
iPlanMethod	Planning Method (forward/ backward)																																										
	Mandatory																																										
oTime	Result Time (planned start/ finish date)																																										
oExceptionMessage	The last message if the return value is not equal to 0.																																										

	If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - No Error. <> 0 - Error.

WorkOrder.GenerateActivitiesFromMasterRouting

DLL:	tsextwcsapi
Syntax:	<pre>long WorkOrder.GenerateActivitiesFromMasterRouting(const domain tcorno iWorkOrder fixed, const domain tsacm.cact iMasterRouting fixed, const long iNumberOfRoutingOptions, const domain tsacm.cact iRoutingOptions() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl. : With this function work order activities (tswcs210) for a certain work order (iWorkOrder) can be generated based on a master routing (iMasterRouting). With the set input variables iNumberOfRoutingOptions and the array iRoutingOptions, the user can control the list of routing options which has to be executed for the given master routing. See the description of these input arguments for more info. Pre : a db.retry.point() must have been specified. Post : an abort.transaction() or commit.transaction() must be executed. Input : iWorkOrder The work order for which the master routing has to be executed (mandatory input and should exist). iMasterRouting The master routing which is going to be used to generate the service order activities. (mandatory input and should exist). iNumberOfRoutingOptions If the iNumberOfRoutingOptions is specified as zero, then, if multiple routing options are present for the specified master routing, the default routing option is executed. If in this scenario the default routing option has not been specified, then an error is returned. If the iNumberOfRoutingOptions is not zero, then the routing options from the array iRoutingOptions are executed. The maximum number of routing options which can be used is 10. iRoutingOptions The array with routing options which need to be executed for the iMasterRouting. They are only considered if the iNumberOfRoutingOptions is unequal to

	zero. If iNumberOfRoutingOptions is not zero, then the first iNumberOfRoutingOptions array elements should exist (allocated) and should not be empty. Note that if an array is given with routing options which are not related to the iMasterRouting then no work order activities will be created, so the system does not check the validity of these routing options. oExceptionMessage The last message if the return value unequals 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Output : N.A. Return : 0 <> 0 - No error - An error occurred

WorkOrder.GetActualRepairCosts

DLL:	tsxtwcsapi
Syntax:	<pre>long WorkOrder.GetActualRepairCosts(domain tcorno iWorkOrder, ref long oNumberOfHomeCurrencies, ref domain tccur oHomeCurrencies() fixed, ref domain tcamnt oMaterialCosts(), ref domain tcamnt oLaborCosts(), ref domain tcamnt oOtherCosts(), ref domain tcamnt oTotalCosts(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: With this function the actual material, labor and other costs for a work order can be determined. The cost arrays are expressed in the home currencies which relate to the financial company of the service department of the work order (tswcs200.cwoc). Note: The output arrays should already have been allocated at the calling side with a depth of 3! Important: the cost arrays are always filled on the first array element position, which represents the local currency. However the costs are not always filled on the reporting currency amount positions, which are the element positions 2 and 3 in the arrays. Only in an independent currency system, or in a standard currency system with the usage of functional currencies switched on, the costs are present on all the used home currency positions. So suppose we have a dependent currency system which uses 3 home currencies, i.e. [EUR, USD, YEN]. This means that the output argument oNumberOfHomeCurrencies

	is filled with 3 and the array oHomeCurrencies is filled with [EUR, USD, YEN]. However, the arrays with the costs are only filled on the first (local currency) position, so i.e. oMaterialCosts = [100.0, 0.0, 0.0] etc. Suppose we have an independent currency system which uses 2 home currencies, i.e. [BRL, USD], then the oNumberOfHomeCurrencies is filled with 2 and the array oHomeCurrencies is filled with [BRL, USD]. The array with costs is filled on all home currency positions, so i.e. oMaterialCosts = [100, 10]. Pre: The output arrays should have been allocated three elements deep. Post: N.A. Input: iWorkOrder The work order (mandatory input). The system will generate an error if the work order does not exist. Output: oNumberOfHomeCurrencies The number of home currencies of the financial company of the work order department. oHomeCurrencies The array (3 deep) with the home currencies of the financial company of the work order department. oMaterialCosts The array (3 deep) with the material costs. oLaborCosts The array (3 deep) with the labor costs. oOtherCosts The array (3 deep) with the other costs. oTotalCosts The array (3 deep) with the total costs. The sum of material, labor and other costs. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - The work order actual repair costs could be determined. > 0 - Error while determining the work order actual repair costs.

WorkOrder.Release

DLL:	tsxtwcsapi
Syntax:	<pre>long WorkOrder.Release(domain tcorno iWorkOrder, domain tsmdm.acln iWorkOrderActivityLine, domain tccotp iPurchaseOrderType, domain tcseri iPurchaseOrderSeries, domain tcyesno iATPCheck, domain tcyesno iCheckPlannedInventory,</pre>

	<pre> domain tcyesno iCheckInventoryOnHand, domain tcyesno iSkipBlockedInventory, domain tsmdm.scin iScopeOfInventoryCheck, domain tcyesno iBlockInCaseOfInventoryShortage, domain tcyesno iUpdatePlannedDeliveryTimeMaterialLine, domain tcyesno iUpdateOrderActivityWithLastPlannedMaterialLine, domain tcyesno iSynchronizeMaterialLinesWithLastPlannedMaterialLine, domain tcgen.ynds iAutomaticItemReceipt, domain tcyesno iUpdatePlannedQuantityToIssuedQuantity, domain tcyesno iShiftWorkOrderToActualReceiptTime, domain tcyesno iCheckAssignments, domain tcyesno iCheckSkills, ref domain tcyesno oMaintenanceSalesOrderBlocked, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function handles the release of a work order. When iWorkOrderActivityLine is filled, the work order will be released and next only this specific work order activity line will be released. Based on the given options/settings for releasing the work order additional checks related to inventory availability are done, as well as checks on the presence of assignments and skills. When 'automatic item receipt' is set to a value other than no, the work order item is first issued to the repair shop before releasing the work item. Before releasing the work order, order blocking checks are done, if defined in the Maintenance Sales Control parameters or service office settings, based on which the linked maintenance sales order can be blocked. When the linked maintenance sales order is set to blocked, the work order cannot be released. In that case output argument oMaintenanceSalesOrderBlocked will be set to 'yes' as the maintenance sales order is blocked. However releasing the work order is not successful. Pre: None. Post: This function sets a retry-point and will commit and/or abort the transaction. Input: iWorkOrder Work Order. (mandatory) iWorkOrderActivityLine Work Order Activity Line. (optional) iPurchaseOrderType Purchase Order Type. (not mandatory) iPurchaseOrderSeries Purchase Order Series. (not mandatory) iATPCheck Perform ATP check. (mandatory Yes/No) iCheckPlannedInventory Check Planned Available Inventory. (mandatory Yes/No) iCheckInventoryOnHand Check On Hand Inventory. (mandatory Yes/No) iSkipBlockedInventory Skip Blocked Inventory. (mandatory Yes/No) iScopeOfInventoryCheck Scope of Inventory Check. (mandatory) Possible values: - current warehouse only - all warehouses in planning cluster

	iBlockInCaseOfInventoryShortage Block Planning or Releasing of Orders with Shortages. (mandatory Yes/No) iUpdatePlannedDeliveryTimeMaterialLine Update Planned Delivery Time of Material Lines. (mandatory Yes/No) iUpdateOrderActivityWithLastPlannedMaterialLine Update Activities and Orders with Latest Planned. Material Line (mandatory Yes/No) iSynchronizeMaterialLinesWithLastPlannedMaterialLine Synchronize Materials with Latest Planned Material Line. (mandatory Yes/No) iAutomaticItemReceipt Automatic Item Receipt. (mandatory) Possible values: - yes - no - use default settings iUpdatePlannedQuantityToIssuedQuantity (Yes/No) Update Planned Quantity to Issued Quantity. (mandatory Yes/No) iShiftWorkOrderToActualReceiptTime Shift Work Order to Actual Receipt Time. (mandatory Yes/No) iCheckAssignments Check Assignments. (mandatory Yes/No) iCheckSkills Check Skills. (mandatory Yes/No) Output: oMaintenanceSalesOrderBlocked Indicates if the maintenance sales order linked to the work order is set to blocked. oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - release successful <> 0 - Error during release occurred

WorkOrder.StartDetail

DLL:	tsextwcsapi
	This function is available from KB3543185 .
Syntax:	<pre>long WorkOrder.StartDetail(long iStartMode, domain tcorno iWorkOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the detail session Work Orders (tswcs2100m000). Input: iStartMode Specifies the start mode for the session.

	Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iWorkOrder Work Order (Mandatory) Note that the given work order must exist. Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

WorkOrder.StartMultiMain

DLL:	tsextwcsapi
Syntax:	<pre>long WorkOrder.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcorno iWorkOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the Multi-Main session Work Order (tswcs2100m100). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. MODAL+START.WITH.ADD.SET - Session is started in Add-mode, so a new work order can be entered directly. The parent session is blocked until the child session exits. MODELESS+START.WITH.ADD.SET - Session is started in Add-mode, so a new work order can be entered directly. Parent and child are parallel sessions that

	can be manipulated simultaneously. iStartFilter Not Used. iQueryExtend A specific query to be used when zooming to this session. This argument is ignored when started in Add-mode. iWorkOrder Work Order This argument is ignored when started in Add-mode. Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred

WorkOrder.StartOverview

DLL:	tsextwcsapi This function is available from KB3543185 .
Syntax:	<pre>long WorkOrder.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iWorkOrder, long iViewFieldSet, ref domain tcorno oWorkOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function starts the overview session Work Orders (tswcs2100m000). Before calling WorkOrder.StartOverview(), call ProcessingOptionSet.Create() and assign the value to iViewFieldSet. After the call the option set can be deleted by calling ProcessingOptionSet.Delete(). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously.

iStartFilter
 Not Used.

iSessionIndex
 Specifies the table-index that is to be used. (Optional)
 Supported values:
 1: sort by Work Order
 2: sort by (Serialized) Item
 3: sort by Service Type
 4: sort by Department, Latest Finish Time
 5: sort by Department, Planned Finish Time
 6: sort by Status
 7: sort by Top Work Order, Initiating Work Order
 8: sort by Sold-to Business Partner, Project

iQueryExtend
 A specific query to be used when zooming to this session. Note that only extensions on main table tswcs200 are supported. (Optional)

iWorkOrder
 The Work Order to display at the top of the grid if present in the chosen view. (Optional)

iViewFieldSet
 Processing Option Set to set the view fields when the index used is not index 1. (Optional)

INDEX	NAME	TYPE	DEFAULT
2	Item	domain tcitem	empty
2	SerialNumber	domain tcibd.sern	empty
3	ServiceType	domain tsmdm.cstp	empty
4*	OperationsDepartment	domain tccwoc	empty
4	LatestFinishTime	domain tsmdm.pldt	0
5*	OperationsDepartment	domain tccwoc	empty
5	PlannedFinishTime	domain tsmdm.pldt	0
6	Status	domain tswcs.stat	tswcs.stat.free
7	MaintenanceSalesOrder	domain tcorno	empty
7	PartMaintenanceLine	domain tcpono	0
7	TopWorkOrder	domain tcorno	empty
7	ReferenceA	domain tcrefa	empty
7	InitiatingWorkOrder	domain tcorno	empty
8	SoldToBusinessPartner	domain tccom.bpid	empty
8	Project	domain tccprj	empty

* The OperationsDepartment property is used for indices 4 and 5.

Output:

oWorkOrder
 The selected Work Order if iStartMode is MODAL and the session is closed by a single selection.

oExceptionMessage
 The last message if any message is found. If more than one message is given, these are present in the oExceptionID.

oExceptionID
 An ID that refers to the exception information. Use the functions in Exception to get all relevant information.

	Return: 0 Session started <> 0 An error occurred

Public Interfaces for WorkOrderActivity

The following functions are available:

[WorkOrderActivity.ProcessReturnDeliveries](#)

[WorkOrderActivity.SignOff](#)

WorkOrderActivity.ProcessReturnDeliveries

DLL:	tsxtwcsapi
	This function is available from KB2281710 .
Syntax:	<pre>long WorkOrderActivity.ProcessReturnDeliveries(domain tcorno iWorkOrder, domain tsmdm.acln iActivityLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this function to start the process of creating Return Deliveries for Work Order Material lines linked to a Work Order Activity. If all material lines related to the Work Order have to be done, then iActivityLine must be 0. If only for a specific activity line number this process has to be executed then specify the iActivityLine with an existing activity line number. Note that because of the possibility that for return warehouse orders, some warehouse activities might have been set to automatic, the system will commit the database transactions in this function. This means that it is not necessary to set a db.retry.point() before calling this function and abort/commit after this function, because that is already handled within this function. Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post: If warehouse procedures have been set to automatic, then these are executed. Input: iWorkOrder - Work Order: Mandatory iActivityLine - Work Order Activity: Not mandatory

Output: oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
Return: 0	- No Error
<> 0	- Error

WorkOrderActivity.SignOff

DLL:	tsxtwcsapi									
	This function is available from KB3610664 .									
Syntax:	<pre>long WorkOrderActivity.SignOff(domain tcorno iWorkOrder, domain tsmdm.acln iActivityLine, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>									
Usage:	Expl: Use this function to sign-off a Work Order Activity. Note that transaction handling is done within this public interface. Pre: Call ProcessingOptionSet.Create() to obtain iProcessingOptionSet. No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post: Delete the option set by calling ProcessingOptionSet.Delete(). Input: iWorkOrder Work Order Mandatory. iActivityLine Work Order Activity Line Mandatory. iProcessingOptionSet Processing Option Set: a processing option set number referring to a processing option set. Mandatory. <table><tr><td>NAME</td><td>TYPE</td><td>DEFAULT</td></tr><tr><td colspan="3">=====</td></tr><tr><td>SignOffWorkOrderWhenAllActivitiesAreSignedOff</td><td>domain tsyesno</td><td>tsyesno.no</td></tr></table> If this option is set to 'Yes', and all Work Order Activities are signed off, the Work Order is signed off.	NAME	TYPE	DEFAULT	=====			SignOffWorkOrderWhenAllActivitiesAreSignedOff	domain tsyesno	tsyesno.no
NAME	TYPE	DEFAULT								
=====										
SignOffWorkOrderWhenAllActivitiesAreSignedOff	domain tsyesno	tsyesno.no								

	CloseWorkOrderWhenAllActivitiesAreClosed domain tyesno tyesno.no If this option is set to 'Yes', and all Work Order Activities are closed, the Work Order is closed. Note: The Work Order Activity is only closed automatically when the option 'Automatically Close Work Order' is turned on in the parameters or settings by Operations Department. AlwaysIgnoreEmptyActualProblemOrSolution domain tyesno tyesno.no If this option is set to 'Yes', signing off the Work Order Activity is not cancelled when the actual problem or actual solution is not filled. Otherwise when the actual problem or actual solution is empty, the signing off is cancelled if all of the below are true: - The Work Order did not originate from a call; - The Problem Solution handling on the service type is 'Warning'; - Parameter 'Diagnostics' is not implemented; - There is no next department transfer work order present. ActionForNonExecutedInspections domain tscfg.upd.ista tscfg.upd.ista.change.status If this option is set to 'Change Status', the status of every non-executed inspection is set to 'Not Measured'. If this option is set to 'Delete Inspections', every non-executed inspection is deleted. If this option is set to 'Cancel', and non-executed inspections are present, then signing off the Work Order Activity is cancelled. IgnoreUnapprovedSubcontractedPurchaseOrders domain tyesno tyesno.no If this option is set to 'No', and the Work Order is internal and there are related subcontracted Purchase Orders that are not approved, then the Work Order Activity is not closed. CloseAllOpenPhysicalBreakdownChanges domain tyesno tyesno.no If this option is set to 'Yes' and the Work Order Activity is closed, then all related open Physical Breakdown Changes are closed. Output: oExceptionMessage The last message if any message is

	found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 No errors occurred. Depending on the options in the processing option set, signing off or closing may still have been cancelled without any message given. <> 0 Error(s) occurred.

Public Interfaces for WorkOrderMaterialCosts

The following functions are available:

[WorkOrderMaterialCosts.ProcessReturnDeliveries](#)

WorkOrderMaterialCosts.ProcessReturnDeliveries

DLL:	tsxtwcsapi
	This function is available from KB2281710 .
Syntax:	<pre>long WorkOrderMaterialCosts.ProcessReturnDeliveries(domain tcorno iWorkOrder, domain tcpono iMaterialLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this function to start the process of creating a Return Delivery for a Work Order Material line. Note that because of the possibility that for return warehouse orders, some warehouse activities might have been set to automatic, the system will commit the database transactions in this function. This means that it is not necessary to set a db.retry.point() before calling this function and abort/commit after this function, because that is already handled within this function. Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post: If warehouse procedures have been set to automatic, then these are executed. Input: iWorkOrder - Work Order: Mandatory

	<code>iMaterialLine</code> - Work Order Material Line: Mandatory Output: <code>oExceptionMessage</code> - The last message if the return value is not equal to 0. If more than one message is given, these are present in the <code>oExceptionID</code> <code>oExceptionID</code> - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - No Error <> 0 - Error

Public Interfaces for PlannedActivity

The following functions are available:

[PlannedActivity.CreateMaintenancePlan](#)

[PlannedActivity.SwitchStatus](#)

[PlannedActivity.TransferToFieldService](#)

PlannedActivity.CreateMaintenancePlan

DLL:	<code>ttextspcapi</code> This function is available from KB3569155.
Syntax:	<pre>long PlannedActivity.CreateMaintenancePlan(long iProcessingOptionSet, ref boolean oPlannedActivityCreated, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: Use this function to generate the maintenance plan. (almost) All selection ranges and options which are available in session <code>tsspc2200m000</code> (Generate Maintenance Plan) and which are not report specific are available as input arguments. Also see the help for session Generate Maintenance Plan (<code>tsspc2200m000</code>). Depending on the selection ranges, the system will determine the set of serialized items for which planned activities can be deleted/created. The system will commit the database changes per serialized item. If for whatever reason, the system

encounters an error, this function will return a value unequal zero and the reason why the generation of the maintenance plan was not successful is present in the oExceptionMessage and oExceptionID.

Note that this means that even though a value <> 0 is returned, still the oPlannedActivityCreated can be true!

This function does its own database handling, so there is no need to specify a db.retry.point() before calling this function and to abort/commit the transactions afterwards.

Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed).

Call ProcessingOptionSet.Create() to obtain the iProcessingOptionSet (XML node).

Post: Delete the option set by calling ProcessingOptionSet.Delete().

Input: iProcessingOptionSet: a processing option set number referring to a processing option set containing at least one valid option. If a certain processing option is not provided in the option set, then the default value will be used!
Mandatory.

NAME	TYPE	DEFAULT
=====		
SelectionMaintenanceScenarios	domain tsspc.sel.pmsc	tsspc.sel.pmsc.all
With this option the user can indicate whether a filter on maintenance scenario level is applicable. If set to tsspc.sel.pmsc.all (or not set as that is the default), then no filtering will be done on scenario level.		
TypeOfScenarioSelection	domain tsspc.sel.type	tsspc.sel.type.both
Only if the SelectionMaintenanceScenarios is set to tsspc.sel.pmsc.specific, this option is considered. There are two categories of scenario lines (Time Based and Usage Based). If the category Time Based is selected, then only time based and pattern based scenario lines are processed. If the category Usage Based is selected, then only usage based scenario lines are processed.		
MaintenanceScenarioFrom	domain tcorno	empty
If the SelectionMaintenanceScenarios is set to tsspc.sel.pmsc.all, this option is ignored.		
MaintenanceScenarioTo	domain tcorno	Maximum of the domain.
If the SelectionMaintenanceScenarios is set to tsspc.sel.pmsc.all, this option is ignored.		
ScenarioLineFrom	domain tcpono	0
If the SelectionMaintenanceScenarios is set to		

	tsspc.sel.pmsc.all, this option is ignored. ScenarioLineTo domain tcpono Maximum of the domain. If the SelectionMaintenanceScenarios is set to tsspc.sel.pmsc.all, this option is ignored. SelectionSerializedItems domain tsspc.sel.item tsspc.sel.item.all With this option the user can indicate whether a filter on serialized item level is applicable. If set to tsspc.sel.item.all (or not set, because this is the default) then no filtering will be done on serialized item level. OwnerFrom domain tccom.bpid empty If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. OwnerTo domain tccom.bpid Maximum of the domain. If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. InUseByBPFrom domain tccom.bpid empty If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. InUseByBPTo domain tccom.bpid Maximum of the domain. If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. DepartmentFrom domain tccwoc empty If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. DepartmentTo domain tccwoc Maximum of the domain. If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. ServiceItemGroupFrom domain tsmdm.csgr empty If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. ServiceItemGroupTo domain tsmdm.csgr Maximum of the domain. If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. SerializedItemGroupFrom domain tscfg.sigr empty If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. If in the Configuration Management Parameters (tscfg0100m000) the option 'Serialized Item Group Usage' is not switched on, then this option is also ignored.

	SerializedItemGroupTo domain tscfg.sigr Maximum of the domain. If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. If in the Configuration Management Parameters (tscfg0100m000) the option 'Serialized Item Group Usage' is not switched on, then this option is also ignored. UsageClassFrom domain tsspc.cusc empty If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. UsageClassTo domain tsspc.cusc Maximum of the domain. If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. InstallationGroupFrom domain tsbsc.clst empty If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. If in the General Service Parameters (tsmdm0100m000), the implemented parameter 'Installation Groups' is not switched on, then this option is also ignored. InstallationGroupTo domain tsbsc.clst Maximum of the domain. If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. If in the General Service Parameters (tsmdm0100m000), the implemented parameter 'Installation Groups' is not switched on, then this option is also ignored. ItemFrom domain tcitem empty If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. ItemTo domain tcitem Maximum of the domain. If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. SerialNumberFrom domain tcibd.sern empty If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. SerialNumberTo domain tcbid.sern Maximum of the domain. If the SelectionSerializedItems is set to tsspc.sel.item.all, this option is ignored. UseHorizon domain tcyesno tcyesno.yes If set to tcyesno.yes, the system will use the FromDateUTC together with the Horizon to calculate a ToDateUTC domain tsmdm.numb 1

	If the UseHorizon is set to tcyesno.no, this option is ignored. HorizonUnit domain tsmdm.pern tsmdm.pern.year If the UseHorizon is set to tcyesno.no, this option is ignored. FromDateUTC domain tcdater utc.num() ToDateUTC domain tcdater FromDateUTC + 1 year. If the UseHorizon is set to tcyesno.yes, this option is ignored. The FromDateUTC together with the Horizon and HorizonUnit will determine then the ToDateUTC. If the UseHorizon is set to tcyesno.no and the ToDateUTC is not provided, then the default will be the FromDateUTC + 1 year. GenerateOnlyInActiveContract domain tcyesno tcyesno.no If set to tcyesno.yes, then only serialized items are considered with an active service contract and planned activities will only be created within the contract period. IgnorePlannedUntilTime domain tcyesno tcyesno.no If set to tcyesno.yes, the system will not check whether a planned activity which is going to be created has a planned finish time before the Planned Until Time (attribute on serialized item level, i.e. tscfg200.pltm). Note that it is dangerous to deviate from the default value (tcyesno.no), because then it is possible that the system will generate duplicate planned activities. TimeBasedMethod domain tsspc.2200.mth tsspc.2200.mth.start.date If a time based scenario line is handled for a serialized item, then this determines the start date from which the planning for that specific serialized item will start. By default (tsspc.2200.mth.start.date) it will start using the serialized item planning start date which is specified in the service planning parameters (tsspc0100m000). See the priorities in the parameters (Generate Planned Activities - Serialized Item Start Date). GenerationTypeTimeBased domain tsspc.gen.type tsspc.gen.type.keep This option controls the fact whether first the existing planned activities will be deleted in the specified planning horizon. For time based the default will be that existing planned activities remain and are not deleted. GenerationTypeUsageBased domain tsspc.gen.type tsspc.gen.type.regenerate This option controls the fact whether first the existing planned activities will be deleted in the specified

	planning horizon. For usage based the default will be that existing planned activities are first deleted, before any new ones are generated. Output: oPlannedActivityCreated If at least one planned activity was generated, then this output variable is set to 'True'. oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - No Error <> 0 - Error present. Note that this function is going to create the maintenance plan (planned activities) for a set of serialized items. If an error occurs while generating the planned activities for a specific serialized item, then the transaction is aborted and the errors are stored in the oExceptionID. The process however continues with the next serialized item. So in the end it could be that the return value is <> 0 (so error occurred), but there are also planned activities created. The latter can be checked with the output argument oPlannedActivityCreated.

PlannedActivity.SwitchStatus

DLL:	tsxtspcapi
Syntax:	<pre>long PlannedActivity.SwitchStatus(const domain tcorno iPlannedActivity fixed, const domain tcpono iActivityLine, const domain tsspc.stat iSwitchStatusTo, const domain tcyesno iPerformATPCheck, const domain tcyesno iPerformPlannedAvailableCheck, const domain tcyesno iPerformOnHandAvailableCheck, const domain tcyesno iSkipBlockedInventory, const domain tsmdm.scin iInventoryScope, const domain tcyesno iBlockRelease, const domain tcyesno iUpdateTimeWithLatestPlannedMaterialLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl. : With this function it is possible for one single planned activity (tsspc200, with key iPlannedActivity and iActivityLine), to switch its status (tsspc200.stat). If a status change is not allowed, then a value <> 0 is returned and the reason why is present in the oExceptionMessage/oExceptionID. Note that even if the return value is 0, so no error, and the status switch is executed, there can still be information in the oExceptionMessage and oExceptionID. This can happen if

	for example the status switch is done to 'Released' and the material availability checks are executed and there is a shortage, but the input argument iBlockRelease is not set to 'yes'. In this scenario the status switch is executed, but a message of type 'Information' is put on the message stack and is present in the oExceptionMessage and oExceptionID. Note that the input arguments after the input argument iSwitchStatusTo (so iPerformATPCheck upto and including i.UpdateTimeWithLatestPlannedMaterialLine) are only used if: 1) The switch is done to the status 'Released' (so i.SwitchStatusTo = tsspc.stat.released). 2) The Material Availability functionality has been switched on in the SPC planning parameters (or if the concept 'Sites' has been implemented and there is a site specific record in tsspc003 with the site from the planned activity (tsspc200.site)). Pre : A db.retry.point() must have been specified. Post : An abort.transaction() or commit.transaction() must be executed. Input : iPlannedActivity - The planned activity order number. Mandatory input. iActivityLine - The planned activity line number. Mandatory input. iSwitchStatusTo - The status to which the planned activity has to be switched. Mandatory input. iPerformATPCheck If the functionality of Material Availability is switched on in the planning parameters and if the status is switched to 'Released' and this input argument is set to 'yes', then the system will perform the ATP (Available To Promise) check. Mandatory input. iPerformPlannedAvailableCheck - If the functionality of Material Availability is switched on in the planning parameters and if the status is switched to 'Released', then when this input argument is set to 'yes', the system will perform the Planned Available check. Mandatory input. iPerformOnHandAvailableCheck - If the functionality of Material Availability is switched on in the planning parameters and if the status is switched to 'Released', then when this input argument is set to 'yes', the system will perform the On Hand check. Mandatory input. iSkipBlockedInventory - If the functionality of Material Availability is

	switched on in the planning parameters and if the status is switched to 'Released', then when this input argument is set to 'yes', the system will skip blocked inventory during the different Material Availability checks. Mandatory input. iInventoryScope - If the functionality of Material Availability is switched on in the planning parameters and if the status is switched to 'Released', then with this input argument it can be controlled whether the Material Availability checks should be executed for the current Warehouse only or for the whole warehouse cluster. Mandatory input. iBlockRelease - If the functionality of Material Availability is switched on in the planning parameters and if the status is switched to 'Released', then if the material availability checks result in a shortage, or if the supply is too late, a DALHOOKERROR is returned otherwise only an information message is put on the message stack. Mandatory input. iUpdateTimeWithLatestPlannedMaterialLine - If the functionality of Material Availability is switched on in the planning parameters and if the status is switched to 'Released', and this input parameters is set to 'yes', then the planning times on the planned activity line (tsspc200) are updated with the times of the latest material line. Mandatory input. Output : oExceptionMessage The last message if the return value is not 0 (error situation). Note that if the return value equals 0, so no error, and the status switch to 'Released' was executed successfully, then it can contain information that there is a material shortage, but that that is not blocking because of the input argument iBlockRelease. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information if the return value of the function is <> 0. Use the functions in Exception to get all relevant information. Note that if the return value equals 0, so no error, and the status switch to 'Released' was executed successfully, then it can contain information that there is a material shortage, but that that is not blocking because of the setting of the input argument iBlockRelease. Return : 0 - No error <> 0 - An error occurred

PlannedActivity.TransferToFieldService

DLL:	tsxextspcapi This function is available from KB2290765 .
Syntax:	<pre> long PlannedActivity.TransferToFieldService(domain tcorno iPlannedActivity fixed, domain tcpono iActivityLine, domain tcyesno iUseCallData, domain tsmdm.pldt iEarliestStartTime, domain tsmdm.pldt iLatestFinishTime, domain tsmdm.pldt iPlannedStartTime, domain tsmdm.pldt iPlannedFinishTime, domain tcyesno iAppointment, domain tcyesno iGroupActivities, domain tcyesno iGroupWithExistingServiceOrders, domain tcyesno iExcludeEmergencyOrders, domain tcyesno iGroupByInstallationGroup, domain tcyesno iGroupByItem, domain tcyesno iGroupBySerialNumber, domain tcyesno iGroupByActivityGroup, domain tcyesno iGroupByLocationAddress, domain tcyesno iGroupByPlannedStartTimesWithinSpanOf, domain tsmdm.tmdu iDifferenceInStartTime, domain tcyesno iPlanActivitiesInSequence, domain tsspc.2220.tra iTravelTime, domain tcyesno iPerformATPCheck, domain tcyesno iPerformPlannedAvailableCheck, domain tcyesno iPerformOnHandAvailableCheck, domain tcyesno iSkipBlockedInventory, domain tsmdm.scin iInventoryScope, domain tcyesno iBlockRelease, domain tcyesno iUpdateActivitiesWithLatestPlannedMaterialLine, domain tcyesno iRecalculatePlannedTimes, domain tcseri iServiceOrderSeries, ref domain tcorno oServiceOrder, ref domain tsmdm.acln oServiceOrderActivity, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: With this function it is possible to transfer a Planned Activity (tsspc200, with key iPlannedActivity and iActivityLine) to a Service Order. If a transfer is not allowed, then a value <> 0 is returned and the reason why is present in the oExceptionMessage/oExceptionID. Note that even if the return value is 0, so no error, and the transfer is executed, there can still be information in the oExceptionMessage and oExceptionID. This can happen when the material availability checks are executed and there is a shortage, but the input argument iBlockRelease is not set to Yes. In this scenario the transfer is executed, but a message of type 'Information' is put on the message stack and is present in the oExceptionMessage and oExceptionID. Note that the input arguments after the input argument iTravelTime (so iPerformATPCheck upto and including iUpdateActivitiesWithLatestPlannedMaterialLine,) are only used when the Material Availability functionality has been

	switched on in the SPC Planning Parameters. All the options which are available in the Transfer-session are also available as input arguments for this interface function. It can be useful to look at the Help for session tsspc2220m000. Pre: A db.retry.point() must have been set. Post: An abort.transaction() or commit.transaction() must be executed. Input: iPlannedActivity The Planned Activity number. Mandatory input. iActivityLine The Planned Activity line number. Mandatory input. iUseCallData Indicator whether the Call or the Planned Activity is used in case the Planned Activity originates from a Call. Mandatory input. iEarliestStartTime Earliest Start Time. Not mandatory. iLatestFinishTime Latest Finish Time. Not mandatory. iPlannedStartTime Planned Start Time. Not mandatory. iPlannedFinishTime Planned Finish Time. Not mandatory. iAppointment Appointment. Only used when iUseCallData is Yes and the Planned Activity originates from a Call. Mandatory input. iGroupActivities Group activities. Mandatory input. iGroupWithExistingServiceOrders If possible the transferred Planned Activity is added to an existing service order. Mandatory input. iExcludeEmergencyOrders Emergency Orders are excluded when the transferred Planned Activity is added to an existing service order. Mandatory input. iGroupByInstallationGroup Group by Installation Group

	Mandatory input. iGroupByItem Group by Item Mandatory input. iGroupBySerialNumber Group by Serial Number Mandatory input. iGroupByActivityGroup Group by Activity Group Mandatory input. iGroupByLocationAddress Group by Location Address Mandatory input. iGroupByPlannedStartTimesWithinSpanOf Group by Planned Start Times within span of... Mandatory input. iDifferenceInStartTime Difference in Start Time, expressed in Time Duration Unit as defined in the General Service Parameters. Not mandatory. iPlanActivitiesInSequence Use parallel planning Yes/No. Mandatory input. iTravelTime When Planned Activity is added to a Service Order, it must be ensured that the Planned Activity's Travel Time is used effectively. This argument can have the following values: Minimum The travel time is the minimum of the planned activity and that of the service order. Maximum The travel time is the maximum of the planned activity and that of the service order. Cumulative The travel time is the cumulative of the planned activity and that of the service order. Mandatory input. iPerformATPCheck Indicator whether the ATP check is performed. Note: value Yes will be ignored and converted to No if material availability is not present in the Planning Parameters (or site specific record, if the concept of Sites has been implemented). Mandatory input. iPerformPlannedAvailableCheck Indicator whether the Planned Inventory check is performed. Note: value Yes will be ignored and converted to No if material availability is not present in the Planning Parameters (or site specific record, if

	the concept of Sites has been implemented). Mandatory input. iPerformOnHandAvailableCheck Indicator whether the On Hand Inventory check is performed. Note: value Yes will be ignored and converted to No if material availability is not present in the Planning Parameters (or site specific record, if the concept of Sites has been implemented). Mandatory input. iSkipBlockedInventory Indicator whether Blocked Inventory has to be considered during the various material availability checks. Note: value Yes only allowed if material availability is present in the Planning Parameters (or site specific record, if the concept of Sites has been implemented). Furthermore, at least one of the input arguments iPerformPlannedAvailableCheck or iPerformOnHandAvailableCheck should have the value Yes. Mandatory input. iInventoryScope Indicator if during the various material availability checks only the current warehouse should be considered or the whole warehouse cluster. Possible values: Current Warehouse Only Checks if the material is available in the warehouse defined on the Material Line. All Warehouses in Planning Cluster Checks if the material is available in one of the warehouses in the same cluster as the warehouse defined on the Material Line. Mandatory input. iBlockRelease If the functionality of Material Availability is switched on in the Planning Parameters and the Service Order is released, then if the material availability checks result in a shortage, or if the supply is too late, a DALHOOKERROR is returned otherwise only an information message is put on the message stack. Mandatory input. iUpdateActivitiesWithLatestPlannedMaterialLine Indicator whether the material availability checks should update the activity and order with the time of the latest planned material line. Note: value Yes only allowed if at least one of the input arguments iPerformATPCheck or iPerformPlannedAvailableCheck has the value Yes. Mandatory input. iRecalculatePlannedTimes Recalculate the planned time for the Planned Activity. Mandatory input. iServiceOrderSeries The series of the Service Orders to which the

	Planned Activity is transferred. Not mandatory. Output : - oServiceOrder Service Order - oServiceOrderActivity Service Order Activity - oExceptionMessage The last message if the return value is not 0 (error situation). Note that if the return value equals 0, so no error, and the status switch to 'Released' was executed successfully, then it can contain information that there is a material shortage, but that that is not blocking because of the input argument iBlockRelease. If more than one message is given, these are present in the oExceptionID - oExceptionID An ID that refers to all error information if the return value of the function is <> 0. Use the functions in Exception to get all relevant information. Note that if the return value equals 0, so no error, and the status switch to 'Released' was executed successfully, then it can contain information that there is a material shortage, but that that is not blocking because of the setting of the input argument iBlockRelease. Return : 0 - No error <> 0 - An error occurred

Public Interfaces for CustomerClaim

The following functions are available:

[CustomerClaim.Close](#)

[CustomerClaim.StartMultiMain](#)

[CustomerClaim.Submit](#)

CustomerClaim.Close

DLL:	tsxtcmmap
	This function is available from KB2267965 .
Syntax:	long CustomerClaim.Close(

	<pre> const domain tcorno iCustomerClaim fixed, const domain tcyesno iMoveClaimToHistory, const domain tcyesno iDeleteClaim, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this function to Close one specific Customer Claim (tscmm100 record). See functionality when a Customer Claim is Closed with session 'Close Customer Claims' (tscmm1250m000). Note that even if the iCustomerClaim already has the status 'Closed', this function can still be called because of the fact that the claim still needs to be moved to the history or should be deleted. Pre: No open database transaction should be present (so before calling this function the existing database transactions should either have been aborted or committed). Post: This function does its own database handling, so there is no need to specify a db.retry.point() before calling this function and to abort/commit the transactions afterwards. Input: iCustomerClaim The customer claim which needs to be closed, moved to history or needs to be deleted. (mandatory) iMoveClaimToHistory Indicator whether the claim should be moved to history after its status is changed to 'Closed'. Note that if customer claim history is not implemented, then the claim cannot be moved to history, so in that situation when iMoveClaimToHistory is set to 'tcyesno.yes', the claim will not be moved to the history tables. (mandatory) iDeleteClaim Indicator whether the claim should be deleted after its status is changed to 'Closed'. Note that if customer claim history is implemented and if the iMoveClaimToHistory is set to 'tcyesno.no' and the iDeleteClaim is set to 'tcyesno.yes', then the claim will not be deleted. This is in line with how the session tscmm1250m000 is handling this. (mandatory) Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Customer Claim status is set to 'Closed' or the status was already 'Closed', but the claim is moved to history and/or is deleted without errors. <> 0 - Error during closing the claim.

CustomerClaim.StartMultiMain

DLL:	tsextcmmap

	This function is available from KB2309036 .	
Syntax:	<pre> long CustomerClaim.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcorno iCustomerClaim, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>	
Usage:	Expl: This function starts the Multi-Main session Customer Claim (tscmm1600m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iQueryExtend A specific query to be used when zooming to this session. iCustomerClaim Customer Claim number Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 Session started <> 0 An error occurred	

CustomerClaim.Submit

DLL:	ttextcmmap	
	This function is available from KB2295922 .	
Syntax:	<pre> long CustomerClaim.Submit(domain tcorno iCustomerClaim, ref boolean oLinesForRMA, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>	
Usage:	Expl: Use this function to submit a Customer Claim. Note that because of the automatic processing of warehousing orders, the system commits the database transactions in this function. Therefore, it is not necessary to set a db.retry.point() before calling this function, and abort/commit after this function, because that is already handled within this function.	

	<pre> Pre: - Post: This function sets a retry-point and will commit/abort the transaction. When oLinesForRMA is true, Customer Claim Acknowledgements might be printed. Input: iCustomerClaim Customer Claim: Mandatory Output: oLinesForRMA When lines are set to Pending Material Return this flag is set to true. oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - No error <> 0 - An error occurred </pre>

Public Interfaces for CustomerClaimLine

The following functions are available:

[CustomerClaimLine.Approve](#)

[CustomerClaimLine.ApproveForInvoiceLinking](#)

[CustomerClaimLine.CalculateClaimAmount](#)

[CustomerClaimLine.Reject](#)

CustomerClaimLine.Approve

DLL:	ttextcmmap	
	This function is available from KB2288749 .	
Syntax:	<pre> long CustomerClaimLine.Approve(domain tcorno iCustomerClaim, domain tcpono iClaimLine, domain tccdis iApprovalReason, domain tcdate iApprovalDate, domain tsmdm.text iApprovalText, domain tscmm.actn iApprovalAction, domain tclogn iApprovedBy, domain tcyesno iUpdateHeaderStatus, </pre>	

	<pre> domain tcyesno.c iReleaseMaterials, ref domain tcyesno iIgnoreCanceledSupplierClaim, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: Use this function to Approve one specific Customer Claim Line. See functionality when a Customer Claim Line is Approved with session Customer Claim Lines (tscmm1110m000). Note that approval is not allowed on line level for the following scenarios: * If Customer Claim Document Approval is set on the claim; * If Workflow is active for Customer Claims. Pre: - Post: This function sets a retry-point and will commit and/or abort the transaction. Input: iCustomerClaim Customer Claim: Mandatory iClaimLine Claim Line: Mandatory iApprovalReason Approval Reason. Note that only a reason code of type Claim Approval which is valid on the current date is allowed. Mandatory when Customer Claim Document Approval is Yes, or when Workflow is active for Customer Claims. Otherwise, not mandatory. iApprovalDate Approval Date: Mandatory iApprovalText Approval Text; A valid text number: Not mandatory iApprovalAction Approval Action: Not mandatory. iApprovedBy Login code of the user approving the line: Mandatory. iUpdateHeaderStatus If Yes, and the approved Claim Line is the last one, the Claim header status is set to Approved too. Mandatory (Yes/No) iReleaseMaterials When not all materials have been released for approval, they are released when iReleaseMaterials is Yes. When not all materials have been released for approval, and iReleaseMaterials is Cancel, Approval is stopped. When iReleaseMaterials is No, approval continues but unreleased materials are not released. Mandatory (Yes/No/Cancel) iIgnoreCanceledSupplierClaim Continue approving when the Supplier Claim for the Claim Line has been canceled. Otherwise, approval stops. Mandatory (Yes/No) Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - No error <> 0 - An error occurred

CustomerClaimLine.ApproveForInvoiceLinking

DLL:	tsxtcmmapi This function is available from KB2295974 .
Syntax:	<pre> long CustomerClaimLine.ApproveForInvoiceLinking(domain tcorno iCustomerClaim, domain tcpono iClaimLine, domain tclogn iApprovedBy, domain tcdate iApprovalDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl.: This function approves a Customer Claim Line for Invoice Linking. Approve for Invoice Linking is only applicable when the Customer Claim Invoice Procedure is Invoice Based. Approve for Invoice Linking is allowed when: - Customer Claim Status is Pending Approval, or Approved. - Customer Claim Line has been Approved. - Customer Claim Line Claim Method is Reimburse Costs. When one of these conditions is not met, an error is returned. Pre: db.retry.point set Post: commit/abort transaction Input: iCustomerClaim Customer Claim: Mandatory iClaimLine Claim Line: Mandatory iApprovedBy Login code of the user approving the line: Mandatory. iApprovalDate Approval Date: Mandatory Output: oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - No error <> 0 - An error occurred

CustomerClaimLine.CalculateClaimAmount

DLL:	tsxtcmmapi This function is available from KB3543998 .

Syntax:	long CustomerClaimLine.CalculateClaimAmount(longiProcessingOptionSet, ref domain tcamntoClaimAmount, ref domain tcmcs.s999moExceptionMessage mb, ref longoExceptionID)																																	
Usage:	Expl.: Use this function to determine the default Claim Amount for the Claim Line. Pre: Call ProcessingOptionSet.Create() to obtain iProcessingOptionSet. Post: Delete the option set by calling ProcessingOptionSet.Delete(). Input: iProcessingOptionSet Processing Option Set: a processing option set number referring to a processing option set containing at least one valid option. Mandatory. <table><tr><th>NAME</th><th>TYPE</th><th>DEFAULT</th></tr><tr><td colspan="3">=====</td></tr><tr><td>Claim</td><td>domain tcorno</td><td>empty</td></tr></table> The Customer Claim to calculate a Claim Amount for. (mandatory) <table><tr><td>ClaimLine</td><td>domain tcpono</td><td>0</td></tr></table> The Claim Line to calculate a Claim Amount for. Mandatory when UseClaimLineData is Yes. Claim Line <table><tr><td>UseClaimLineData</td><td>domain tcyesno</td><td>tcyesno.yes</td></tr></table> Use Claim Line Data; when Yes, the data of the passed Claim Line is retrieved and used for the calculation. In this case, ClaimLine is mandatory. When No, the Claim Line is ignored and the data passed via iProcessingOptionSet is used. In this way a simulated Claim Amount can be calculated. <table><tr><td>CostType</td><td>domain tsmdm.cotp</td><td>tsmdm.cotp.material</td></tr></table> Cost Type (tscmm110.cotp) <table><tr><td>ClaimMethod</td><td>domain tscmm.clmt</td><td>tscmm.clmt.costs</td></tr></table> Claim Method (tscmm110.clmt) <table><tr><td>ServiceType</td><td>domain tsmdm.cstp</td><td>empty</td></tr></table> Service Type (tscmm110.cstp) <table><tr><td>Item</td><td>domain tcitem</td><td>empty</td></tr></table> Item (tscmm110.item) <table><tr><td>SerialNumber</td><td>domain tcibd.sern</td><td>empty</td></tr></table> Serial Number (tscmm110.sern) <table><tr><td>LaborRateCode</td><td>domain tcppl.clrt</td><td>empty</td></tr></table>	NAME	TYPE	DEFAULT	=====			Claim	domain tcorno	empty	ClaimLine	domain tcpono	0	UseClaimLineData	domain tcyesno	tcyesno.yes	CostType	domain tsmdm.cotp	tsmdm.cotp.material	ClaimMethod	domain tscmm.clmt	tscmm.clmt.costs	ServiceType	domain tsmdm.cstp	empty	Item	domain tcitem	empty	SerialNumber	domain tcibd.sern	empty	LaborRateCode	domain tcppl.clrt	empty
NAME	TYPE	DEFAULT																																
=====																																		
Claim	domain tcorno	empty																																
ClaimLine	domain tcpono	0																																
UseClaimLineData	domain tcyesno	tcyesno.yes																																
CostType	domain tsmdm.cotp	tsmdm.cotp.material																																
ClaimMethod	domain tscmm.clmt	tscmm.clmt.costs																																
ServiceType	domain tsmdm.cstp	empty																																
Item	domain tcitem	empty																																
SerialNumber	domain tcibd.sern	empty																																
LaborRateCode	domain tcppl.clrt	empty																																

	Labor Rate Code (tscmm110.clrt) LaborType domain tckow empty Labor Type (tscmm110.chlt) TravelSpecificationType domain tssoc.tspt tssoc.tspt.not.app Travel Specification Type (tscmm110.tspt) ClaimedQuantity domain tsmdm.qmat 0.0 Claimed Quantity (tscmm110.qccl) QuantityUnit domain tccuni empty Quantity Unit of Claimed Quantity (tscmm110.cucq) ConvFactQtyUnit domain tcconv 1.0 Conversion Factor of Claimed Quantity Unit (tscmm110.cvcq) Output: oClaimAmount The Claim Amount expressed in the Claim Currency. oExceptionMessage The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 No Error and the Claim Amount is returned. <> 0 Error situation.

CustomerClaimLine.Reject

DLL:	tsxtcmmap
	This function is available from KB2288749 .
Syntax:	<pre>long CustomerClaimLine.Reject(domain tcorno iCustomerClaim, domain tcpono iClaimLine, domain tccdis iRejectionReason, domain tcdate iRejectionDate, domain tsmdm.text iRejectionText, domain tscmm.actn iRejectionAction, domain tclogn iRejectedBy, domain tcyesno iUpdateHeaderStatus, domain tcyesno.c iReleaseMaterials, domain tcyesno iSolveRelatedCall, ref domain tcmcs.s999m oExceptionMessage mb,</pre>

	ref	long	oExceptionID)
Usage:	Expl:	Use this function to Reject one specific Customer Claim Line. See functionality when a Customer Claim Line is Rejected with session Reject Customer Claim (tscmm1220m000). On rejecting the last claim line, the header status can be set to Rejected as well. This behavior can be controlled with the use of argument iUpdateHeaderStatus.	
	Pre:	-	
	Post:	This function sets a retry-point and will commit and/or abort the transaction.	
	Input:	iCustomerClaim Customer Claim: Mandatory iClaimLine Claim Line: Mandatory iRejectionReason Rejection Reason. Note that only a reason code of type Claim Rejection which is valid on the current date is allowed: Mandatory. iRejectionDate Rejection Date: Mandatory iRejectionText Rejection Text; A valid text number: Not mandatory. iRejectionAction Rejection Action: Mandatory iRejectedBy Login code of the user rejecting the line: Mandatory. iUpdateHeaderStatus If Yes, and the rejected Claim Line is the last one, the Claim header status is set to Rejected too. Mandatory (Yes/No) iReleaseMaterials When not all materials have been released for approval, they are released when iReleaseMaterials is Yes. When not all materials have been released for approval, and iReleaseMaterials is Cancel, Approval is stopped. When iReleaseMaterials is No, approval continues but unreleased materials are not released. Mandatory (Yes/No/Cancel) iSolveRelatedCall When header status is set to Rejected, and there is a related Call, the Call is set to Solved when iSolveRelatedCall is Yes. Mandatory (Yes/No)	
	Output:	oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information.	
	Return	: 0 - No error <> 0 - An error occurred	

Public Interfaces for SupplierClaim

The following functions are available:

[SupplierClaim.SetRMAReceived](#)

SupplierClaim.SetRMAReceived

DLL:	tsxtcmmapi
	This function is available from KB2294949 .
Syntax:	<pre> long SupplierClaim.SetRMAReceived(domain tcorno iSupplierClaim, domain tcpono iClaimLine, domain tcitem iItem, domain tcibd.sern iSerialNumber, domain tcyesno iReturnMaterialRequired, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this function to set the field "Receive RMA" of the Supplier Claim Delivery Line related to the given Supplier Claim Line to Received. If Return Material is required and a Warehouse order does not yet exist, this function creates a Warehouse order for the Supplier Claim Delivery Line. Pre: a db.retry.point() must have been specified. Post: an abort.transaction() or commit.transaction() must be executed. Input: iSupplierClaim Supplier Claim: Mandatory iClaimLine Supplier Claim Line: Mandatory iItem Item: Mandatory if return material is required iSerialNumber Serial Number: Mandatory if Item is serialized iReturnMaterialRequired Whether or not return material is required: Mandatory Output: oExceptionMessage The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - No error <> 0 - An error occurred

Public Interfaces for SupplierClaimLine

The following functions are available:

[SupplierClaimLine.Approve](#)

[SupplierClaimLine.Reject](#)

[SupplierClaimLine.Settle](#)

SupplierClaimLine.Approve

DLL:	tsextcmmap	
Syntax:	<pre> long SupplierClaimLine.Approve(domain tcorno iSupplierClaim, domain tcpono iClaimLine, boolean iAllowToUpdateHeaderStatus, domain tsmdm.text iApprovalText, domain tcdis iApprovalDecision, domain tscmm.actn iApprovalAction, domain tclogn iApprovedBy, domain tcdate iApprovalDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>	
Usage:	Expl: Use this function to Approve one specific Supplier Claim Line (tscmm210 record). See functionality when a Supplier Claim Line is Approved with session 'Supplier Claim Lines' (tscmm2110m000). Note that approval is not allowed on line level for the scenarios below: * If Supplier Claim Document Approval is set on the claim. * If Workflow is active for supplier claims Pre : a db.retry.point() must have been specified. Post : an abort.transaction() or commit.transaction() must be executed. Input : - iSupplierClaim - The Supplier Claim Mandatory - iClaimLine - The Claim Line Mandatory - iAllowToUpdateHeaderStatus - If true, and the approved Claim Line is the last one, the Claim header status is set to Approved as well. - iApprovalText - The Approval Text, A valid text number. Not mandatory. - iApprovalDecision - The Approval Decision. Note that only a reason code of type 'Claim Approval' which is	

	valid on the current date is allowed. Not mandatory. iApprovalAction - The Approval Action. Not mandatory. iApprovedBy - The login code of the user who is approving the line. Mandatory. iApprovalDate - The Approval Date Mandatory Output : oExceptionMessage - The last message if the return value unequals 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return : 0 - No error <> 0 - An error occurred

SupplierClaimLine.Reject

DLL:	tsxtcmmap
Syntax:	<pre> long SupplierClaimLine.Reject(domain tcorno iSupplierClaim, domain tcpono iClaimLine, boolean iAllowToUpdateHeaderStatus, domain tclogn iRejectedBy, domain tccdis iRejectionReason, domain tcdate iRejectionDate, domain tsmdm.text iRejectionText, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this function to Reject one specific Supplier Claim Line (tscmm210 record). See functionality when a Supplier Claim Line is Rejected with session 'Reject Supplier Claim' (tscmm2220m000). When manually Rejecting the last claim line, a question is asked whether to set the header status to Rejected as well. This behaviour can be controlled with the use of argument 'i.allow.to.update.header.status'. Pre : a db.retry.point() must have been specified. Post : an abort.transaction() or commit.transaction() must be executed. Input : iSupplierClaim - The Supplier Claim Mandatory iClaimLine - The Claim Line Mandatory iAllowToUpdateHeaderStatus - If true, and the rejected Claim Line is the last one, the Claim header status is set to Rejected as well.

	iRejectedBy iRejectionReason iRejectionDate iRejectionText Output : oExceptionMessage oExceptionID Return : 0 <> 0	- The login code of the user who is rejecting. Mandatory. - The Rejection Reason. Note that only a reason code of type 'Claim Rejection' which is valid on the current date is allowed. Mandatory. - The Rejection Date Mandatory - The Rejection Text, A valid text number. Not mandatory. - The last message if the return value unequals 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - No error - An error occurred

SupplierClaimLine.Settle

DLL:	tsextcmmapi	
	This function is available from KB3554806 .	
Syntax:	<pre>long SupplierClaimLine.Settle(domain tcorno iSupplierClaim, domain tcpono iClaimLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: Use this function to Settle one specific Supplier Claim Line (tscmm210 record). See functionality when a Supplier Claim Line is Settled with Form Command 'Settle' via a Supplier Claim Line(s) session (tscmm2110m000 etc.). Pre: a db.retry.point() must have been specified. Post: an abort.transaction() or commit.transaction() must be executed. Input: iSupplierClaim - The Supplier Claim number, Mandatory iClaimLine - The Claim Line number, Mandatory Output: oExceptionMessage oExceptionID Return: 0 <> 0	

Chapter 28 Public Interfaces for Invoicing

Public Interfaces for Invoice

The following functions are available:

[Invoice.ComposePrintPostInvoices](#)

[Invoice.CreditInvoiceLine](#)

[Invoice.GetInterimRevenueLedgerAccountAndDimensions](#)

[Invoice.StartMultiMain](#)

Invoice.ComposePrintPostInvoices

DLL:	ciextslapi This function is available from KB3529255 .
Syntax:	<pre> long Invoice.ComposePrintPostInvoices (domain tcncmp iSalesInvoiceCompany, domain cisli.rbrs iRecurringInvoicingBatch, domain tcdate iCutoffDate, domain tcslitinv iTypeOfInvoice, domain tcorno iInvoicingBatchFrom, domain tcorno iInvoicingBatchTo, boolean iUseNextOpenPeriod, long iProcessingOptionSet, ref long oNumberOfInvoicingBatchesComposed, ref long oNumberOfInvoicingBatchesDecomposed, ref long oNumberOfInvoicingBatchesSubmitted, oNumberOfInvoicingBatchesSubmittedToExternalSystem, ref long oNumberOfInvoicingBatchesPrinted, ref long oNumberOfInvoicingBatchesPosted, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface is used to execute compose/print/post process for the given invoicing batches. Pre: No open database transaction. Post: Not Applicable

Input: iSalesInvoiceCompany - Company of the Sales Invoice This is a Mandatory field. iRecurringInvoicingBatch- Recurring Invoicing Batch iCutoffDate - Cut-off Date iTypeOfInvoice - Type of Invoice This is a Mandatory field. Possible values : Standard tcsli.tinv.standard : Pro Forma tcsli.tinv.pro.forma : Customs tcsli.tinv.customs : Consignment tcsli.tinv.consignment iInvoicingBatchFrom - Invoicing Batch From iInvoicingBatchTo - Invoicing Batch To iUseNextOpenPeriod - Use Next Open Periods. This is only applicable for Invoice Date and Post Transaction Entry Date. Possible Values : True - If the period is closed for the given Invoice Date or Post Transaction Entry Date then the next open period will be assigned. false - If the period is closed for the given Invoice Date or Post Transaction Entry Date then the process will stop and error message will be given. iProcessingOptionSet - Processing Option Set This is a Mandatory field. A Processing Option Set can be created via a call to ProcessingOptionSet.Create(). Processing Options have a direct relationship with the form fields on session Compose/Print/Post Invoices (cisli2200m000) and are not explained in further detail here. Please refer to the session help for additional information. Processing Options which are set while on session cisli2200m000 the option is not applicable are ignored.		
NAME	TYPE	DEFAULT
* Actions		
invoiceCompose	domain tcyesno	tcyesno.no
invoiceDate	domain tcdate	utc.num()
invoiceUndoCompose	domain tcyesno	tcyesno.no
invoiceDelete	domain tcyesno	tcyesno.no
invoiceSubmit	domain tcyesno	tcyesno.no
invoiceSubmitToExternalSystem	domain tcyesno	tcyesno.no
invoicePrint	domain tcyesno	tcyesno.no
invoicePost	domain tcyesno	tcyesno.no
invoicePrintTransactionEntryDate	domain tcdate	0
invoicePostTransactionEntryDate	domain tcdate	0
* Print		
invoicePrintOption	domain cisli.prno	cisli.prno.not.appl
invoicePrintinEuro	domain tcyesno	tcyesno.no
invoicePrintDeliveryMethod	domain tcyesno	tcyesno.no
invoicePrintToSingleFile	domain tcyesno	tcyesno.no
invoicePrintOriginalCreditNote	domain tcyesno	tcyesno.no
invoicePrintSequence	domain cisli.psco	" "
* Output Devices		
invoicePrintDevice	domain cisli.devc	" "
invoicePostingDevice	domain cisli.devc	" "

	<pre> invoiceExceptionsDevice domain cisli.devcc "" Output: oNumberOfInvoicingBatchesComposed - Number of invoicing batch composed. oNumberOfInvoicingBatchesDecomposed - Number of invoicing batch decomposed. oNumberOfInvoicingBatchesSubmitted - Number of invoicing batch submitted. oNumberOfInvoicingBatchesSubmittedToExternalSystem - Number of invoicing batch submitted to external system. oNumberOfInvoicingBatchesPrinted - Number of invoicing batch printed. oNumberOfInvoicingBatchesPosted - Number of invoicing batch posted oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - <> 0 - Error </pre>

Invoice.CreditInvoiceLine

DLL:	ciextsliapi This function is available from KB3513949 .
Syntax:	<pre> long Invoice.CreditInvoiceLine(domain tcncmp iFinancialCompany, domain tctran iTransactionType, domain tcgld.docn iInvoiceNumber, domain tciseq iInvoiceLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface is used to create Credit Note for the given Invoice Line. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: - iFinancialCompany - Financial Company of the Invoice. (This is a Mandatory field) - iTransactionType - Transaction Type of the Invoice. (This is a Mandatory field) - iInvoiceNumber - Invoice Number. (This is a Mandatory field) - iInvoiceLine - Invoice Line. (This is a Mandatory field) Output: - oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID.

	oExceptionID Return: 0 <> 0	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information. - Credit Note created successfully. - Error

Invoice.GetInterimRevenueLedgerAccountAndDimensions

DLL:	ciextslapi																																	
	This function is available from KB2331941 .																																	
Syntax:	<pre>long Invoice.GetInterimRevenueLedgerAccountAndDimensions(domain tcncmp iFinancialCompany, domain tctran iTransactionType, domain tcgld.docn iInvoiceNumber, domain tciseq iInvoiceLine, domain tccom.bpid iInvoiceToBusinessPartner, domain tccom.cadr iInvoiceToAddress, domain tccom.bpid iSoldToBusinessPartner, domain tccom.cadr iSoldToAddress, domain tcccty iTaxCountry, domain tccvat iTaxCode, domain tcccty iBpTaxCountry, domain tcyesno iAssetDisposal, ref domain tfgld.leac oLedgerAccount, ref domain tfgld.dimx oDimensions() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>																																	
Usage:	Expl: This function retrieves ledger account and dimensions from the mapping scheme for a manual sales invoice with a fixed asset. Pre: Not applicable Post: Not applicable Input: <table><tr><td>iFinancialCompany</td><td>- Financial Company: Mandatory</td></tr><tr><td>iTransactionType</td><td>- Invoice Transaction Type: Mandatory</td></tr><tr><td>iInvoiceNumber</td><td>- Invoice Number: Mandatory</td></tr><tr><td>iInvoiceLine</td><td>- Invoice Line: Optional</td></tr><tr><td>iInvoiceToBusinessPartner</td><td>- Invoice-to Business Partner: Optional</td></tr><tr><td>iInvoiceToAddress</td><td>- Address Invoice-to Business Partner: Optional</td></tr><tr><td>iSoldToBusinessPartner</td><td>- Sold-to Business Partner: Optional</td></tr><tr><td>iSoldToAddress</td><td>- Address Sold-to Business Partner: Optional</td></tr><tr><td>iTaxCountry</td><td>- Tax Country: Optional</td></tr><tr><td>iTaxCode</td><td>- Tax Code: Optional</td></tr><tr><td>iBpTaxCountry</td><td>- Business Partner Tax Country: Optional</td></tr><tr><td>iAssetDisposal</td><td>- Asset Disposal: Mandatory</td></tr></table> Output: <table><tr><td>oLedgerAccount</td><td>- Ledger Account</td></tr><tr><td>oDimensions</td><td>- Array with Dimensions</td></tr><tr><td>oExceptionMessage</td><td>- The last message if any message is found. If more than one message is given, these are present in the oExceptionID</td></tr><tr><td>oExceptionID</td><td>- An ID that refers to the exception</td></tr></table>		iFinancialCompany	- Financial Company: Mandatory	iTransactionType	- Invoice Transaction Type: Mandatory	iInvoiceNumber	- Invoice Number: Mandatory	iInvoiceLine	- Invoice Line: Optional	iInvoiceToBusinessPartner	- Invoice-to Business Partner: Optional	iInvoiceToAddress	- Address Invoice-to Business Partner: Optional	iSoldToBusinessPartner	- Sold-to Business Partner: Optional	iSoldToAddress	- Address Sold-to Business Partner: Optional	iTaxCountry	- Tax Country: Optional	iTaxCode	- Tax Code: Optional	iBpTaxCountry	- Business Partner Tax Country: Optional	iAssetDisposal	- Asset Disposal: Mandatory	oLedgerAccount	- Ledger Account	oDimensions	- Array with Dimensions	oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID	oExceptionID	- An ID that refers to the exception
iFinancialCompany	- Financial Company: Mandatory																																	
iTransactionType	- Invoice Transaction Type: Mandatory																																	
iInvoiceNumber	- Invoice Number: Mandatory																																	
iInvoiceLine	- Invoice Line: Optional																																	
iInvoiceToBusinessPartner	- Invoice-to Business Partner: Optional																																	
iInvoiceToAddress	- Address Invoice-to Business Partner: Optional																																	
iSoldToBusinessPartner	- Sold-to Business Partner: Optional																																	
iSoldToAddress	- Address Sold-to Business Partner: Optional																																	
iTaxCountry	- Tax Country: Optional																																	
iTaxCode	- Tax Code: Optional																																	
iBpTaxCountry	- Business Partner Tax Country: Optional																																	
iAssetDisposal	- Asset Disposal: Mandatory																																	
oLedgerAccount	- Ledger Account																																	
oDimensions	- Array with Dimensions																																	
oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID																																	
oExceptionID	- An ID that refers to the exception																																	

	information. Use the functions in Exception to get all relevant information Return: 0/DALHOOKERROR

Invoice.StartMultiMain

DLL:	ciextsliapi
Syntax:	<pre> long Invoice.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcncmp iFinancialCompany, domain tctran iTransactionType, domain tcgld.docn iInvoiceNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface starts session 'Invoice' (cisli3605m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "" (Session has no start filter) iSessionIndex iSessionIndex is not used (yet) because session has only 1 index iQueryExtend Optional A specific query to be used when zooming to this session. Primary Key Fields: iFinancialCompany iTransactionType iInvoiceNumber Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started

	<> 0	- Error

Public Interfaces for ProformaInvoice

The following functions are available:

[ProformaInvoice.CancelInvoice](#)

[ProformaInvoice.StartWorkbench](#)

ProformaInvoice.CancelInvoice

DLL:	ciextslapi
	This function is available from KB3533175 .
Syntax:	<pre> long ProformaInvoice.CancelInvoice(domain tcncmp iFinancialCompany, domain tctran iTransactionType, domain tcgld.docn iInvoiceNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface is used to cancel a printed Proforma Invoice. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: - iFinancialCompany - Financial Company of the Invoice. (This is a Mandatory field) - iTransactionType - Transaction Type of the Invoice. (This is a Mandatory field) - iInvoiceNumber - Invoice Number. (This is a Mandatory field) Output: - oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Printed Pro Forma Invoice is Canceled successfully. <> 0 - Error

ProformaInvoice.StartWorkbench

DLL:	ciextsliapi
Syntax:	<pre> long ProformaInvoice.StartWorkbench(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccom.bpid iInvoiceToBusinessPartner, domain tcncmp iSourceCompany, domain tcsli.srtp iSourceType, domain tcorno iOrder, domain tcsli.oref iSourceDocumentLine, domain tcinv iInvoiceLineType, domain tcsli.tinv iTypeOfInvoice, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface starts session 'Proforma Invoicing Workbench' (cisli3640m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter iStartFilter is not used (yet) iSessionIndex iSessionIndex is not used (yet) because session has only 1 index iQueryExtend iQueryExtend is not used (yet) iInvoiceToBusinessPartner In the satellites the data for this Invoice to Business Partner will be shown. iSourceCompany In the satellites the data for this Source Company will be shown. iSourceType In the satellites the data for this Source Type will be shown. iOrder In the satellites the data for this Ordernumber will be shown. iSourceDocumentLine In the satellites the data for this Source Document Line will be shown. iInvoiceLineType In the satellites the data for this Invoice Line Type will be shown. The value of the 'source type' determines which Invoice Line types are possible. iTypeOfInvoice In the satellites the data for this Invoice Type will be shown. Possible values : Pro Forma tcsli.tinv.pro.forma : Customs tcsli.tinv.customs

The following functions are available:

Installment.CreateCorrections

Installment.ApproveCorrections

Infor LN Public Interfaces & Process Extensions Reference Guide (On-premises) | 1051

	Sales Order - tcsli.srtp.sales.order Service Order - tcsli.srtp.service.order Maintenance Sales Order - tcsli.srtp.maint.sales.ord iOrderNumber - Order Number. (This is a Mandatory field) iOrderReference - Order Reference of the Order. Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Corrections approved. <> 0 - Error

Installment.CreateCorrections

DLL:	ciextslapi
	This function is available from KB3507528 .
Syntax:	<pre>long Installment.CreateCorrections(domain tcncmp iSourceCompany, domain tcsli.srtp iSourceType, domain tcorno iOrderNumber, domain tcsli.oref iOrderReference, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface is used to create corrections for billable installment lines of the given order. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iSourceCompany - Source Company of the Order. (This is a Mandatory field) iSourceType - Source Type of the Order. (This is a Mandatory field) Possible values : Sales Order - tcsli.srtp.sales.order Service Order - tcsli.srtp.service.order Maintenance Sales Order - tcsli.srtp.maint.sales.ord iOrderNumber - Order Number. (This is a Mandatory field) iOrderReference - Order Reference of the Order. Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant

	Return: 0 <> 0	information. - Corrections Created - Error

Installment.TransferCorrections

DLL:	ciextsliapi
	This function is available from KB3507555 .
Syntax:	<pre> long Installment.TransferCorrections(domain tcncmp iSourceCompany, domain tcsli.srtp iSourceType, domain tcorno iOrderNumber, domain tcsli.oref iOrderReference, ref tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface is used to transfer the corrections of billable installment lines for the given order. Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: iSourceCompany - Source Company of the Order. (This is a Mandatory field) iSourceType - Source Type of the Order. (This is a Mandatory field) Possible values : Sales Order - tcsli.srtp.sales.order Service Order - tcsli.srtp.service.order Maintenance Sales Order - tcsli.srtp.maint.sales.ord iOrderNumber - Order Number. (This is a Mandatory field) iOrderReference - Order Reference of the Order. Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Corrections transferred. <> 0 - Error

Public Interfaces for InvoicingBatch

The following functions are available:

[InvoicingBatch.CreateInvoice](#)

InvoicingBatch.CreateInvoice

DLL:	ciextslapi
	This function is available from KB2306144 .
Syntax:	<pre> long InvoicingBatch.CreateInvoice(domain tcncmp iSalesInvoiceCompany, domain tcsli.tinv iTypeOfInvoice, domain tcorno iInvoicingBatchFrom, domain tcorno iInvoicingBatchTo, ref tcmcs.long oNumberOfInvoicingBatchesPosted, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface is used to create an invoice for the given range of Invoicing Batches. The billable lines linked to the invoicing batch will be composed, printed and posted. Number of invoicing batches posted will be returned as an output. During this process, the transaction handling is being done in this function. Retry point, commit transaction and abort transaction will be handled in this function. Pre: No open database transaction. Post: NA Input: iSalesInvoiceCompany - Sales Invoice Company. This is a Mandatory field. iTypeOfInvoice - The Type of Invoice. This is a Mandatory field. Possible values : Standard tcsli.tinv.standard : Pro Forma tcsli.tinv.pro.forma : Customs tcsli.tinv.customs : Consignment tcsli.tinv.consignment iInvoicingBatchFrom - Invoicing Batch from iInvoicingBatchTo - Invoicing Batch to This is a Mandatory field. Output: oNumberOfInvoicingBatchesPosted - Number of Invoicing Batches posted. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Session started

	<> 0	- Error

Public Interfaces for BillableLine

The following functions are available:

[BillableLine.CreateInvoice](#)

[BillableLine.UpdateStatus](#)

BillableLine.CreateInvoice

DLL:	ciextslapi
	This function is available from KB2306144 .
Syntax:	<pre> long BillableLine.CreateInvoice(domain tcncmp iSalesInvoiceCompany, domain tcsli.tinv iTypeOfInvoice, domain tcmcs.long iBillableLineKeyCollection, ref domain tcmcs.long oNumberOfInvoicingBatchesPosted, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface is used to create an invoice for the given key collection of billable lines. If no invoicing batch is linked to a billable line, a new invoicing batch will be created automatically and will be processed to create an invoice. If an invoicing batch is already linked to the billable line, an invoice will be created for the invoicing batch. The billable lines linked to the invoicing batch will be composed, printed and posted. Number of invoicing batches posted will be returned as an output. During this process, the transaction handling is being done in this function. Retry point, commit transaction and abort transaction will be handled in this function. ----- Start of Example of Implementation Requirement - Create Invoice for a Manual Sales Invoice. Solution - Read all the confirmed billable lines created for the manual sales invoice id and create a

	<pre> billable line key collection. Pseudocode - table tcisli810 * Billable Lines long collection.key.fields long keyfields.object long ret collection.key.fields = create.keyfields.collection() select cisli810.* from cisli810 where cisli810._index1 = { :i.financial.company, tcsli.srtp.manual.sales, :i.manual.sales } selectdo keyfields.object = create.keyfields.object("cisli810", collection.key.fields) keyfields.to.object(keyfields.object) endselect ret = BillableLine.CreateInvoice(iSalesInvoiceCompany, iTypeOfInvoice, collection.key.fields, oNumberOfInvoicingBatchPosted) delete.keyfields.collection(collection.key.fields) End of Example of Implementation ----- Pre: No open database transaction. Post: Billable line key collection 'iBillableLineKeyCollection' should be deleted after the public interface has been called. Use 'delete.keyfields.collection(iBillableLineKeyCollection)' Input: iSalesInvoiceCompany - Sales Invoice Company. This is a Mandatory field. iTypeOfInvoice - The Type of Invoice. This is a Mandatory field. Possible values : Standard tcsli.tinv.standard : Pro Forma tcsli.tinv.pro.forma : Customs tcsli.tinv.customs : Consignment tcsli.tinv.consignment iBillableLineKeyCollection - Billable line Key Collection This is a Mandatory field. Output: oNumberOfInvoicingBatchesPosted - Number of Invoicing Batches posted. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant </pre>

	Return: 0 <> 0	information. - Session started - Error

BillableLine.UpdateStatus

DLL:	ciextslapi	
	This function is available from KB3519213 .	
Syntax:	<pre> long BillableLine.UpdateStatus(domain tcncmp iSourceCompany, domain tcsli.srtp iSourceType, domain tcorno iOrderNumber, domain tcsli.orln iOrderLine, domain tcsli.oref iOrderReference, domain tcsli.tref iTechnicalReference, domain tcpono iBillingSequence, domain tcsli.stat iNewBillableLineStatus, ref boolean iSynchronizeBillingSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>	
Usage:	Expl: This public interface is used to update the billable line status to the given new status if possible. Pre: Caller must set retry-point. Post: Caller must commit/abort transaction. Input: - i.calling.api - Calling API - iSourceCompany - Source Company (This is a Mandatory field) - iSourceType - Source Type (This is a Mandatory field) - iOrderNumber - Order Number (This is a Mandatory field) - iOrderLine - Order Line - iOrderReference - Order Reference - iTechnicalReference - Technical Reference - iBillingSequence - Billing Sequence (This is a Mandatory field) - iNewBillableLineStatus - New Billable Line Status (This is a Mandatory field) - iSynchronizeBillingSet - Synchronize the status for all the billable lines having the same billing set. Output: - oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - Billable Line updated successfully - Otherwise	

Public Interfaces for ManualSalesInvoice

The following functions are available:

[ManualSalesInvoice.ConfirmInvoices](#)

ManualSalesInvoice.ConfirmInvoices

DLL:	ciextsliapi This function is available from KB3553572 .
Syntax:	<pre> long ManualSalesInvoice.ConfirmInvoices (domain tcncmp iFinancialCompany, boolean iContinueIfShipToBusinessPartnerIsInactive, domain tcorno iManualSalesInvoiceFrom, domain tcorno iManualSalesInvoiceTo, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This public interface is used to confirm the manual sales invoices for the given range of manual sales invoice Id's. The lines whose status is On-hold will be set to confirmed. Note: - Transaction handling is done within the function. The commit will be done per manual sales id. The behaviour is set similar to cisli2219m000 session. Pre: Not applicable. Post: Not applicable. Input: iContinueIfShipToBusinessPartnerIsInactive - This variable is taken into account only when Ship-to Business Partner of the Manual Sales invoice is Inactive. - Possible values are: - True: - If Ship-to business partner status is not active then manual sales invoice will be confirmed but Ship-to Business Partner will be empty in the billable lines. - False:- If Ship-to business partner status is not active then the manual sales invoice will not be confirmed and the process will skip the record and search for the next record. iFinancialCompany - Financial Company (This is a Mandatory field) iManualSalesInvoiceFrom - Manual Sales Invoice From (This is a Mandatory field) iManualSalesInvoiceTo - Manual Sales Invoice To (This is a Mandatory field) Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception

	information. Use the functions in Exception to get all relevant information. - Manual Sales Invoices confirmed successfully - Otherwise Return: 0 <> 0

Chapter 29 Public Interfaces for Taxation

Public Interfaces for Tax

The following functions are available:

[Tax.Calculate](#)

[Tax.GetBusinessPartnerTaxRegistrationID](#)

[Tax.GetDefaultsGoods](#)

[Tax.GetRate](#)

[Tax.ProcessTestScenario](#)

Tax.Calculate

DLL:	tcexttaxapi		
Syntax:	<pre>long Tax.Calculate(domain tcncmp iFinancialCompany, domain tcccty iTaxCountry, domain tccvat iTaxCode, domain tcccur iCurrency, domain tccom.bpid iInvoiceToBusinessPartner, domain tccom.bpid iSoldToBusinessPartner, domain tccom.bpid iBuyFromBusinessPartner, domain tcfcicu iFinancialBPGroup, domain tccwar iWarehouse, domain tccwoc iDepartment, domain tcmcs.st30m iCustomerOrderReference mb, domain tccdec iDeliveryTerms, domain tcptpa iPointOfTitlePassage, domain tcfovn iExemptCertificate, domain tccdis iExemptReason, domain tctax.toam iTypeOfAmount, domain tcamnt iTaxableOrderLineAmount, domain tcamnt iCustomsValue, domain tcdate iOrderDate, domain tcdate iInvoiceDate, domain tcdate iDeliveryDate,</pre>		

	<pre> domain tcdate iTaxDate, domain tcorno iOrderNumber, domain tcpono iOrderLine, domain tcinvn iInvoiceNumber, domain tcmcs.long iInvoiceLine, domain boolean iLastInvoiceLine, domain tcgld.ttyp iTransactionType, domain tcqdhc iOrderQuantity, domain tctxin iSalesServiceRentalUsage, domain tcprel iRelationType, domain tcitem iItem, domain tccitg iItemGroup, domain tclct.type iLandedCostType, domain tccprj iProject, domain tctano iStandardLabor, domain tctano iProjectLabor, domain tctax.cico iStandardSundryCosts, domain tctax.cico iProjectSundryCosts, domain tccom.sctp iServiceCostType, domain tctax.ctpc iContractType, domain tctax.cstp iServiceType, domain tcgld.leac iAccountNumber, domain tccom.cadr iShipFromAddress, domain tccom.cadr iShipToAddress, domain tccom.cadr iAdminShipFromAddress, domain tccom.cadr iAdminShipToAddress, domain tcyesno iRoundTaxAmount, domain tcyesno iUpdateTaxRegister, domain boolean iUseAmountEnteredInFinancials, domain tcyesno iPerceptionTax, ref domain tcamnt oTotalTaxAmount, ref domain tcpvat oTotalTaxPercentage, ref domain tcmcs.xml oTaxCalcResultsNode, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl. This Public Interface calculates the tax amount of a logistic or financial transaction. Pre: not applicable Post: not applicable Input: In the description below an indicator has been given for every input field to make clear where it is used: I : used in Internal Tax calculation E : used in External Tax calculation PT: used in Perception Tax (for Argentina) calculation iFinancialCompany - Financial Company (I/E/PT (mandatory)) iTaxCountry - Tax Country (I/E/PT(mandatory)) iTaxCode - Tax Code. In case of a Notax tax code, an early return is done (I/E(mandatory)) iCurrency - Transaction Currency, the currency in which the amount is expressed. (I/PT (mandatory)) iInvoiceToBusinessPartner - Invoice-to Business Partner (PT) iSoldToBusinessPartner - Sold-to Business Partner (I/E) iBuyFromBusinessPartner - Buy-From BP; fill this field only in

Warehouse		case of Purchase Tax Origins. (I/E)
	iFinancialBPGroup	- Financial BP Group (E)
	iWarehouse	- Warehouse (E)
	iDepartment	- Department, fill this field only if the is not available (e.g. Manual Sales Invoice) (E)
	iCustomerOrderReference	- Customer Order Reference (E)
	iDeliveryTerms	- Delivery Terms (E)
	iPointOfTitlePassage	- Point of Title Passage (E)
	iExemptCertificate	- Exempt certificate (I/E)
	iExemptReason	- Exempt reason (I/E)
	iTypeOfAmount	- Specification of the characteristics of variable 'iTaxableOrderLineAmount'. * Gross: Amount includes tax * Net: Amount is without tax; true for calls from TD. * Gross over Hundred (I)
	iTaxableOrderLineAmount	- Taxable amount, the order line amount minus discounts (if applicable). (I/E/PT)
	iCustomsValue	- Customs Value (I)
	iOrderDate	- Order Date (I)
	iInvoiceDate	- Invoice Date (I/PT)
	iDeliveryDate	- Delivery Date One of these dates is used for determining the date for which the tax rates apply. If the invoice date is present, then this date is used. Else, if the delivery date is present, then this date is used. Else the order date is used. (I)
	iTaxDate	- Tax Date (E/PT)
	iOrderNumber	- Order Number (E)
	iOrderLine	- Order Line (E)
	iInvoiceNumber	- Invoice Number (E)
	iInvoiceLine	- Invoice Line (E)
	iLastInvoiceLine	- Indication if the current invoice line is the last one of the invoices. Only applicable if 'Update Tax Register' is set to 'Yes'. (E)

iTransactionType	- Transaction Type (currently not used) (E)
iOrderQuantity	- Order Quantity (E)
iSalesServiceRentalUsage	- Options are 'Sales', 'Service', 'Rental' or 'Purchase' (E)
iRelationType	- Product Relation. The field determines which of the following field must be taken into account for determination of the product category. (E)
iItem	- Item (for determining the product category) (E)
iItemGroup	- Item Group (for determining the product category) (E)
iLandedCostType	- Landed Cost Type (for determining the product category) (E)
iProject	- Project (for determining the product category) (E)
iStandardLabor	- Standard Labor (for determining the product category) (E)
iProjectLabor	- Project Labor (for determining the product category) (E)
iStandardSundryCosts	- Standard Sundry Costs (for determining the product category) (E)
iProjectSundryCosts	- Project Sundry Costs (for determining the product category) (E)
iServiceCostType	- Service Cost Type (for determining the product category) (E)
iContractType	- Contract Type (for determining the product category) (E)
iServiceType	- Service Type (for determining the product category) (E)
iAccountNumber	- Ledger Account (for determining the product category) (E)
iShipFromAddress	- Ship-from Address (E)
iShipToAddress	- Ship-to Address (E)
iAdminShipFromAddress	-Point of Acceptance Address. This is the place where the business is actually agreed / signed. Normally this is the buy-from address. (E)
iAdminShipToAddress	-Point of Order Origin. Normally this is the sold-to address, but it might differ due to the point of title passage.

	(E) iRoundTaxAmount - If 'Yes', then the output tax amount is rounded. If 'No', then the output tax amount is not rounded (I) iUpdateTaxRegister - If 'Yes' then the Vertex database is updated with new rates. If 'No' then the Vertex database is only used for retrieval of rates. (E) iUseAmountEnteredInFinancials - This value only applies to Financials. Integrations from other packages will have this field set to false. If the value is set, then the gross/net indicator of the tax code must be used when calculating the tax amount. (E) iPerceptionTax - If 'Yes', the tax calculation is done for Perception Tax (in Argentina). Perception Taxes are tax amounts that are applied to documents at header level. Therefore, the input variable iTaxableOrderLineAmount must be filled with the total net invoice amount. The Tax Result Node will contain the Perception Tax Codes that are applied. Mandatory input of this function: - iFinancialCompany - iTaxCountry - iCurrency - iInvoiceToBp - iTypeOfAmount must be tctax.toam.net - iTaxableOrderLineAmount - iInvoiceDate - iTaxDate If 'No' the regular tax retrieval applies. (PT) Output: oTotalTaxAmount - Total tax amount oTotalTaxPercentage - Total tax percentage oTaxCalcResultsNode - XML node which stores the calculation details. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 Success Error

Tax.GetBusinessPartnerTaxRegistrationID

DLL:	tcexttaxapi
	This function is available from KB2311239 .
Syntax:	<pre> long Tax.GetBusinessPartnerTaxRegistrationID(domain tcncmp iFinancialCompany, const domain tccom.bpid iBusinessPartner, const domain tcccty iCountry, const domain tcmcs.cste iState, const domain tcezy iEconomicZoneType, domain tcdate iReferenceDate, ref domain tctax.txnb oBusinessPartnerTaxRegistrationID, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function retrieves the business partner's tax registration ID. Pre: NA Post: NA Input: - iFinancialCompany - Financial Company (Mandatory) - iBusinessPartner - Business Partner (Mandatory) - iCountry - Country (Mandatory) - iState - State - iEconomicZoneType - Economic Zone Type - iReferenceDate - Reference Date (This is a UTC Date/Time.) Output: - oBusinessPartnerTaxRegistrationID - Business Partner Tax Registration ID - oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Success <> 0 - Error

Tax.GetDefaultsGoods

DLL:	tcexttaxapi
	This function is available from KB3525116 .
Syntax:	<pre> long Tax.GetDefaultsGoods(domain tcncmp iFinancialCompany, const domain tccom.bpid iShipFromBusinessPartner, const domain tccom.bpid iShipToBusinessPartner, const domain tccom.bpid iInvoiceFromBusinessPartner, const domain tccom.bpid iInvoiceToBusinessPartner, const domain tccom.bpid iBuyFromBusinessPartner, const domain tccom.bpid iSoldToBusinessPartner, domain tctax.oorg iOriginalTaxOrigin, </pre>

		domain	tctax.oorg	iCurrentTaxOrigin,
		domain	tcitr.scen	iIntercompanyTradeScenario,
		domain	tccls.scen	iClassificationScenario,
	const	domain	tcitem	iItem,
		domain	tckitm	iItemType,
	const	domain	tccitg	iItemGroup,
	const	domain	tcatse	iAttributeSet,
	const	domain	tcgeoc	iGeoDestination,
	const	domain	tcgeoc	iGeoOrigin,
	const	domain	tcgeoc	iGeoPostalAddress,
	const	domain	tcsite	iShipFromSite,
	const	domain	tcsite	iShipToSite,
	const	domain	tccom.cadr	iShipFromAddress,
	const	domain	tccom.cadr	iShipToAddress,
	const	domain	tccom.cadr	iBuyFromAddress,
	const	domain	tccom.cadr	iSoldToAddress,
	const	domain	tccom.cadr	iInvoiceFromAddress,
	const	domain	tccom.cadr	iInvoiceToAddress,
		domain	tcncmp	iOperationalCompany,
	const	domain	tcdec	iTermsOfDelivery,
	const	domain	tcptpa	iPointOfTitlePassage,
		domain	tctax.ttyp	iTradeType,
		domain	tcdate	iDateUTC,
		domain	tctax.vtbo	iVatBasedOn,
	const	domain	tccwar	iWarehouse,
	const	domain	tccwoc	iWorkCenter,
	const	domain	tcccty	iPurchaseTaxCountry,
	const	domain	tccvat	iPurchaseTaxCode,
	const	domain	tccwoc	iAdminDepartment,
	const	domain	tccwoc	iFinancialDepartment,
	const	domain	tccom.cadr	iFinancialDepartmentAddress,
	const	domain	tctax.bpcl	iTaxClassification,
	const	domain	tccsec	iBusinessSector,
	const	domain	tcccat	iCategory,
	const	domain	tccspa	iElement,
	const	domain	tccact	iActivity,
		domain	tcpcs.cotp	iCostType,
	const	domain	tcpcs.coob	iCostObject,
		domain	tcyesno	iTaxApplicable,
	ref	domain	tcccty	oTaxCountry,
	ref	domain	tcpcs.cste	oTaxState,
	ref	domain	tcezty	oTaxEconomicZoneType,
	ref	domain	tccvat	oTaxCode,
	ref	domain	tcccty	oBusinessPartnerTaxCountry,
	ref	domain	tcpcs.cste	oBusinessPartnerTaxState,
	ref	domain	tcezty	oBusinessPartnerTaxEconomicZoneType,
	ref	domain	tcyesno	oExempt,
	ref	domain	tcfovn	oExemptCertificate,
	ref	domain	tccdis	oExemptReason,
	ref	domain	tcpcs.s250m	oOverallMessage mb,
	ref	domain	tcpcs.s999m	oExceptionMessage mb,
	ref		long	oExceptionID)
Usage:	Expl:	This function is used to get the default tax related information based on the given input. This function determines the appropriate		
		* Tax Country * Tax State * Tax Economic Zone Type * Tax Code * Business Partner Tax Country * Business Partner Tax State * Business Partner Tax Economic Zone Type		

	* Exempt * Exempt Certificate * Exempt Reason to be used when calling from application. * The Tax Country is the Country where you as Company are responsible for Tax. * The Tax Code is the percentage on which the Tax amount is based on. * the Business Partner Tax Country is the Country where the Business Partner is responsible for Tax. Pre: All arguments should be filled whenever possible. Because input arguments are valid for both Sales and Purchase transactions, not all arguments can be filled in all cases. Post: NotApplicable Input: <table border="0"> <tr> <td>iFinancialCompany</td><td>- Financial Company</td></tr> <tr> <td>iShipFromBusinessPartner</td><td>- Ship-from Business Partner</td></tr> <tr> <td>iShipToBusinessPartner</td><td>- Ship-to Business Partner</td></tr> <tr> <td>iInvoiceFromBusinessPartner</td><td>- Invoice-from Business Partner</td></tr> <tr> <td>iInvoiceToBusinessPartner</td><td>- Invoice-to Business Partner</td></tr> <tr> <td>iBuyFromBusinessPartner</td><td>- Buy-from Business Partner</td></tr> <tr> <td>iSoldToBusinessPartner</td><td>- Sold-to Business Partner</td></tr> <tr> <td>iOriginalTaxOrigin</td><td>- Original Tax Origin</td></tr> <tr> <td>iCurrentTaxOrigin</td><td>- Current Tax Origin</td></tr> <tr> <td>iIntercompanyTradeScenario</td><td>- Intercompany Trade Order Scenario</td></tr> <tr> <td>iClassificationScenario</td><td>- Classification Scenario</td></tr> <tr> <td>iItem</td><td>- Item</td></tr> <tr> <td>iItemType</td><td>- Item Type</td></tr> <tr> <td>iItemGroup</td><td>- Item Group</td></tr> <tr> <td>iAttributeSet</td><td>- Arrtribute Set</td></tr> <tr> <td>iGeoDestination</td><td>- GEO Code Destination</td></tr> </table> Hierarchy necessary to determine GEO codes: Modules: Sales a. Use pstc.dl if non-blank b. If pstc.dl is blank, then use pstc.cs if non-blank. c. If pstc.cs is blank, then use the address of ship-to business partner Modules: Purchase a. Use pstc.dl if non-blank b. If pstc.dl is blank, then use pstc.wh if non-blank. c. If pstc.wh is non-blank, then use the address of the company associated on order. iGeoOrigin - GEO Code Origin Hierarchy necessary to determine GEO codes: Modules: Sales a. Use pstc.wh if non-blank b. If pstc.wh is blank, then use the address of the company associated on the order. Modules: Purchase a. Use pstc.cs if non-blank b. If pstc.cs is blank, then the address of the ship-from business partner. iGeoPostalAddress - GEO Code Postal Address Hierarchy necessary to determine GEO codes: Modules: Sales Modules: Purchase a. Use pstc.pa iShipFromSite - Ship-from Site	iFinancialCompany	- Financial Company	iShipFromBusinessPartner	- Ship-from Business Partner	iShipToBusinessPartner	- Ship-to Business Partner	iInvoiceFromBusinessPartner	- Invoice-from Business Partner	iInvoiceToBusinessPartner	- Invoice-to Business Partner	iBuyFromBusinessPartner	- Buy-from Business Partner	iSoldToBusinessPartner	- Sold-to Business Partner	iOriginalTaxOrigin	- Original Tax Origin	iCurrentTaxOrigin	- Current Tax Origin	iIntercompanyTradeScenario	- Intercompany Trade Order Scenario	iClassificationScenario	- Classification Scenario	iItem	- Item	iItemType	- Item Type	iItemGroup	- Item Group	iAttributeSet	- Arrtribute Set	iGeoDestination	- GEO Code Destination
iFinancialCompany	- Financial Company																																
iShipFromBusinessPartner	- Ship-from Business Partner																																
iShipToBusinessPartner	- Ship-to Business Partner																																
iInvoiceFromBusinessPartner	- Invoice-from Business Partner																																
iInvoiceToBusinessPartner	- Invoice-to Business Partner																																
iBuyFromBusinessPartner	- Buy-from Business Partner																																
iSoldToBusinessPartner	- Sold-to Business Partner																																
iOriginalTaxOrigin	- Original Tax Origin																																
iCurrentTaxOrigin	- Current Tax Origin																																
iIntercompanyTradeScenario	- Intercompany Trade Order Scenario																																
iClassificationScenario	- Classification Scenario																																
iItem	- Item																																
iItemType	- Item Type																																
iItemGroup	- Item Group																																
iAttributeSet	- Arrtribute Set																																
iGeoDestination	- GEO Code Destination																																

iShipToSite	- Ship-to Site
iShipFromAddress	- Ship-from Address
iShipToAddress	- Ship-to Address
iBuyFromAddress	- Buy-from Address
iSoldToAddress	- Sold-to Address
iInvoiceFromAddress	- Invoice-from Address
iInvoiceToAddress	- Invoice-to Address
iOperationalCompany	- Operational Company
iTermsOfDelivery	- Terms of Delivery
iPointOfTitlePassage	- Point of Title Passage
iTradeType	- Trade Type
* tctax.ttyp.dir.del (when called from tdpur)	
* tctax.ttyp.dir.del.int.inv (when called from whinh)	
* tctax.ttyp.triangular (when called from whinh)	
* tctax.ttyp.bilateral (when called from whinh)	
* tctax.ttyp.not.appl (in all other cases)	
iDateUTC	- UTC Date
iVatBasedOn	- VAT Based on
iWarehouse	- Warehouse
iWorkCenter	- Work Center
iPurchaseTaxCountry	- Purchase Tax Country
iPurchaseTaxCode	- Purchase Tax Code
iAdminDepartment	- Admin Department
iFinancialDepartment	- Financial Department
iFinancialDepartmentAddress	- Financial Department Address
iTaxClassification	- Tax Classification
iBusinessSector	- Business Sector
iCategory	- Category
iElement	- Element
iActivity	- Activity
iCostType	- Cost Type
iCostObject	- Cost Object
iTaxApplicable	- Tax Applicable
No:	- In case tds000.atin.4 (Apply Tax to Installments) = 'No' and the transaction is an installment.
	- In case of advance payment requests from Contract
Yes:	- In all other cases.
Output: oTaxCountry	- Tax Country
oTaxState	- Tax State
oTaxEconomicZoneType	- Tax Economic Zone Type
oTaxCode	- Tax Code
oBusinessPartnerTaxCountry	- Business Partner Tax Country
oBusinessPartnerTaxState	- Business Partner Tax State
oBusinessPartnerTaxEconomicZoneType	- Business Partner Economic Zone Type
oExempt	- Exempt
oExemptCertificate	- Exempt Certificate
oExemptReason	- Exempt Reason
oOverallMessage	- Overall Message
Return: 0/DALHOOKERROR	

Tax.GetRate

DLL:	tcexttaxapi
Syntax:	<pre> long Tax.GetRate(domain tcncmp iCompany, domain tcccty iTaxCountry, domain tccvat iTaxCode, domain tcdate iEffectiveDate, ref domain tcpvat oTaxRate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function retrieves the Tax Rate of a Tax Country/Tax Code combination. If the Tax Code is a Group Tax Code the VAT Tax Code of that group is used instead. Pre: na Post: na Input: iCompany - Company (mandatory) iTaxCountry - Tax Country (mandatory) iTaxCode - Tax Code (mandatory) iEffectiveDate - Effective date (mandatory) Output: oTaxRate - Tax Rate oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 succes <> 0 error

Tax.ProcessTestScenario

DLL:	tcexttaxapi
	This function is available from KB3605590 .
Syntax:	<pre> long Tax.ProcessTestScenario(const domain tclogn iLogonCode, domain tctax.seqn iTestNumber, ref domain tcccty oTaxCountry, ref domain tccvat oTaxCode, ref domain tcpvat oTaxRate, ref domain tcmcs.cste oTaxState, ref domain tcezy oTaxEconomicZoneType, ref domain tcccty oBusinessPartnerTaxCountry, ref domain tcmcs.cste oBusinessPartnerTaxState, ref domain tcezy oBusinessPartnerTaxEconomicZoneType, ref domain tcyesno oExempt, ref domain tcfovn oExemptCertificate, ref domain tccdis oExemptReason, ref domain tcmcs.s999m oExceptionMessage mb, </pre>

Public Interfaces for Taxation

	ref	long	oExceptionID)
Usage:	Expl:	This function processes the given tax scenario and returns the results.	
	Pre:	The test scenario to be processed must be available in session 'Test Tax Scenario' (tctax8100m000).	
	Post:	na	
	Input:	iLogonCode	- Logon Code (mandatory)
		iTestNumber	- Test Number (mandatory)
	Output:	oTaxCountry	- Tax Country
		oTaxCode	- Tax Code
		oTaxRate	- Tax Rate
		oTaxState	- Tax State
		oTaxEconomicZoneType	- Tax Economic Zone Type
		oBusinessPartnerTaxCountry	- Business Partner Tax Country
		oBusinessPartnerTaxState	- Business Partner Tax State
		oBusinessPartnerTaxEconomicZoneType	- Business Partner Tax Economic Zone Type
		oExempt	- Exempt
		oExemptCertificate	- Exempt Certificate
		oExemptReason	- Exempt Reason
		oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.
		oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.
	Return:	0	- succes
		<> 0	- otherwise

Chapter 30 Public Interfaces for BOD & BDE

Public Interfaces for BOD

The following functions are available:

[BOD.ActionsAfterProcessingIncomingRequest](#)

[BOD.ActionsAfterProcessingIncomingRequestWithAutomaticProcessing](#)

[BOD.ActionsBeforeProcessingIncomingRequest](#)

[BOD.ComposeLocationId](#)

[BOD.ConvertFromERPDateToISODate](#)

[BOD.ConvertFromERPItem](#)

[BOD.ConvertFromISODateToERP](#)

[BOD.ConvertFromISOUnitToERP](#)

[BOD.ConvertToERPItem](#)

[BOD.CreateCodeDefinitionReferenceNode](#)

[BOD.CreateIDNode](#)

[BOD.CreateLastModificationPersonNode](#)

[BOD.CreateNoteNode](#)

[BOD.ExecuteMethod](#)

[BOD.ExecuteOnAcknowledgeForChameleon](#)

[BOD.ExecuteOnLoadForChameleon](#)

[BOD.ExecuteOnProcessForChameleon](#)

[BOD.ExecuteOnSyncForChameleon](#)

[BOD.ExecuteOnUpdateForChameleon](#)

[BOD.ExecutePublishForChameleon](#)

[BOD.ExecuteRequest](#)

[BOD.ExecuteShowForChameleon](#)

[BOD.GetIdAccountingEntity](#)[BOD.GetIdLocation](#)[BOD.GetIdLogicalID](#)[BOD.GetPublishingAllowed](#)[BOD.GetTextNumberFromNoteNode](#)[BOD.GetValueFromNodeElement](#)[BOD.HandleStagingAfterProcessingIncomingRequest](#)[BOD.HandleStagingBeforeProcessingIncomingRequest](#)[BOD.InterruptPublishing](#)[BOD.Publish](#)[BOD.PublishLNMessage](#)[BOD.PublishWithBatch](#)[BOD.ResumePublishing](#)

BOD.ActionsAfterProcessingIncomingRequest

DLL:	ttextbodapi		
Syntax:	<pre>long BOD.ActionsAfterProcessingIncomingRequest (long iXMLRequest, ref long oResult, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>		
Usage:	Expl: This function publishes the staged BODs that were triggered from the processing of the incoming request and resets the parameter that enabled the staging of outgoing BODs in the BOD.ActionsBeforeProcessingIncomingRequest() function. Preferably, it should be called in the AfterExecuteHook of every OnEvent section, e.g. the OnProcess, OnLoad, OnSync, OnShow, OnAcknowledge etc. Pre: Before the incoming BOD is processed, function BOD.ActionsBeforeProcessingIncomingRequest() must be called. Post: NA Input: iXMLRequest - XML structure with request. Mandatory Output: oResult - 0 if succes, otherwise <> 0 oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information.		

	Return: 0 <> 0	- Staged BODs are published. - Staged BODs could not be published.

BOD.ActionsAfterProcessingIncomingRequestWithAutomaticProcessing

DLL:	ttextbodapi	
Syntax:	<pre> long BOD.ActionsAfterProcessingIncomingRequestWithAutomaticProcessing(long iXMLRequest, domain tcmcs.str132 iObjectType, domain tcmcs.str132 iObject, ref long oResult, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>	
Usage:	Expl: This function publishes the staged BODs that were triggered from the processing of the incoming request and resets the parameter that enabled the staging of outgoing BODs in the BOD.ActionsBeforeProcessingIncomingRequest() function. Furthermore, it starts the automatic processing (if any) for the object that was created/updated in LN from the incoming request. Preferably, this function should be called in the AfterExecuteHook of every OnEvent section, e.g. the OnProcess, OnLoad, OnSync, OnShow, OnAcknowledge etc. Pre: Before the incoming BOD is processed, function BOD.ActionsBeforeProcessingIncomingRequest() must be called. Post: NA Input: - iXMLRequest - XML structure with request. Mandatory - iObjectType - object type for which automatic processing must be started, e.g. "PurchaseOrder" - iObject - The identifier of the given ObjectType e.g. the purchase order if ObjectType is "PurchaseOrder" Output: - oResult - 0 if succes, otherwise <> 0 - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - Staged BODs are published. - Staged BODs could not be published.	

BOD.ActionsBeforeProcessingIncomingRequest

DLL:	tcextbodapi
Syntax:	<pre> long BOD.ActionsBeforeProcessingIncomingRequest (long iXMLRequest, ref boolean oCancel, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function validates the incoming request and sets parameters needed during the processing. It validates if the Accounting Entity and the Location (Location is not checked for master data BODs) of the incoming request matches the Accounting Entity and Location of the current company (in Tenant, Accounting Entity and Location setup or in BOD Parameters) and sets output argument oCancel to true in case of a mismatch. It extracts the LastModificationPerson/IDs/ID from the incoming request and determines the linked LN user. If the LN user can be determined, it switches user to the LN user. It sets a parameter that enables the staging of outgoing BODs that are triggered from the processing of the incoming request. Preferably, it should be called in the BeforeExecuteHook of every OnEvent section, e.g. the OnProcess, OnLoad, OnSync, OnShow, OnAcknowledge etc. Pre: NA Post: After the incoming request is processed, BOD.ActionsAfterProcessingIncomingRequest() or BOD.ActionsAfterProcessingIncomingRequestWithAutomaticProcessing() must be called. Input: iXMLRequest - XML structure with request. Mandatory Output: oCancel - true, if BOD must be cancelled oExceptionMessage - false, if BOD can be processed oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID Return: 0 - An ID that refers to all error information. Use the functions in Exception to get all relevant information. <> 0 - No error occurred while executing the actions. - An error occurred while executing the actions, or a mismatch was detected during accounting entity / location validation and BOD parameter Inbound Routing Error Handling is set to Return Error.

BOD.ComposeLocationId

DLL:	tcextbodapi

	This function is available from KB2275062 .	
Syntax:	<pre> long BOD.ComposeLocationId(domain tcmcs.str20 iType, domain tcncmp iOperationalCompany, domain tcbod.lctn iLocation, ref domain tcmcs.str50 oLocationIdentifier, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>	
Usage:	Expl: This function composes the location identifier to be used in the LocationBOD or in any Location/ID elements that reference the LocationBOD. Pre: NA Post: NA Input : iType - The type of the location. Mandatory. The type should be one of following values: "Department" "Project" "Warehouse" "Company" "Site" - iOperationalCompany - The operational company to which the location belongs. Only used for location type Site. If not supplied, the current company is assumed. - iLocation - The location (department, project, warehouse, company or site) in LN. Mandatory. - Output: oLocationIdentifier - The composed location identifier used in the BODs. - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Return: 0 - OK. <> 0 - Error occurred.	

BOD.ConvertFromERPDateToISODate

DLL:	tcextbodapi	
	This function is available from KB2308619 .	
Syntax:	<pre> long BOD.ConvertFromERPDateToISODate(domain tcdte iERPDate, long iDateConversion, long iDateFormat, boolean iERPDateIsUTC, ref domain tcmcs.str30 oISODate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>	

Usage:	<pre> Expl : This function converts a Date in LN (with Data Type "UTC Date/Time" or "Date") to ISO 8601 format. Additionally this function can set a date in ISO 8601 format if the Date in LN is 0/empty. Pre : - Post : - Input : iERPDate - The date in LN. iDateConversion - This determines the value of oISODate if iERPDate is equal to 0. Allowed values: 1 - Set to minimum date 2 - Set to maximum date 3 - Set to current date 4 - Set no date (oISODate will be empty) 5 - Set to current date without time part iDateFormat - This determines the time part of oISODate. Allowed values: 0 - Keep the standard time format: time is kept if iERPDateIsUTC is true and no time is set if iERPDateIsUTC is false 1 - Set the time part 2 - Set no time part iERPDateIsUTC - Mandatory. Allowed values: true: the Data Type for iERPDate is "UTC Date/Time". false: the Data Type for iERPDate is "Date". Output : oISODate - The date in ISO 8601 format. oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - OK. <> 0 - Conversion failed. </pre>

BOD.ConvertFromERPItem

DLL:	tcextbodapi
Syntax:	<pre> long BOD.ConvertFromERPItem(domain tcitem iERPItem, ref domain tcmcs.str50 oBODItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	<pre> Expl. : This function converts the ERP item to the external item used in the BODs. Pre : - Post : - Input : iERPItem - Mandatory. Output : oBODItem oExceptionMessage - The last message if the return value is not equal to 0. </pre>

	If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - iERPItem converted to oBODItem - Otherwise. Return : 0 <> 0

BOD.ConvertFromISODateToERP

DLL:	tcextbodapi This function is available from KB2308619.
Syntax:	<pre>long BOD.ConvertFromISODateToERP(domain tcmcs.str30 iISODate, ref domain tcdatetime oERPDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl : This function converts a string, representing a date[time] in ISO 8601 format to a Date in LN (with Data Type "UTC Date/Time" or "Date"). Pre : - Post : - Input : iISODate - The date in ISO 8601 format. Mandatory. The following formats are supported: "yyyy-mm-ddThh:mm:ssZ" (GMT) "yyyy-mm-ddThh:mm:ss+hh:mm" "yyyy-mm-ddThh:mm:ss-hh:mm" "yyyy-mm-ddThh:mm:ss" (interpreted as local date/time) "yyyy-mm-dd" - The date in LN. If iISODate contains the character T, the date is converted to a UTC value, which contains the date and time based on the LN Data Type "UTC Date/Time". The earliest date is 1970-01-01. If iISODate does not contain the character T, the date is converted to a local date value based on the LN Data Type "Date". This does not take the time zone into account. Output : oERPDate oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return : 0 <> 0 - OK. - Conversion failed.

BOD.ConvertFromISOUnitToERP

DLL:	tcextbodapi This function is available from KB2302511.
Syntax:	<pre> long BOD.ConvertFromISOUnitToERP(domain tcitem iItem, domain tcctc iUnit, boolean iCheckUnitSet, boolean iCheckConversionFactor, ref domain tccuni oERPUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function converts the ISO unit code, to the ERP unit code. When item is empty, first unit found with the correct ISO code, is returned; when no unit is found, unit is searched for that has empty ISO code and with unit equal to the ISO code. When item is filled, last unit found with the correct ISO code, and found unit is in unit set of item and has conversion factor to the item's inventory unit, is returned. When unit found equals inventory unit or weight unit of item, this unit overrules other units and is returned; when no unit is found, unit is searched for that has empty ISO code and with unit equal to the ISO code. Example 1: - units with correct ISO code and in unit set with valid conversion factor: A, B and C - inventory unit of item = B - B is always returned Example 2: - units with correct ISO code and in unit set with valid conversion factor: A, B and C - inventory unit of item = D - C is returned Example 3: - no units with correct ISO code - there is a unit X that equals the ISO code X - X is returned Pre: NA Post: NA Input: - iItem - ERP Item - iUnit - ISO unit code. Mandatory - iCheckUnitSet - unit set must be checked, true/false - iCheckConversionFactor - conv factor must be checked, true/false Output: - oERPUnit - ERP Unit code - oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - OK.

	<> 0	- Error occurred.

BOD.ConvertToERPItem

DLL:	tcextbodapi										
Syntax:	<pre> long BOD.ConvertToERPItem(domain tcmcs.str50 iBODItem, ref domain tcitem oERPItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>										
Usage:	Expl. : This function sets the ERP LN item, using the item as received from other product. (e.g. WMS) Process: Dynamic translation by project lookup. The first 9 characters of the item (or less dependent on item segmentation) are taken, and looked up as project code in the project table. If this code exists, the item code is assumed to represent a customized item and not converted. Otherwise it is treated as standard item and leading spaces are added. EXAMPLES: <table border="0"> <thead> <tr> <th>CBO item string</th><th>ERP item</th></tr> </thead> <tbody> <tr> <td>"PR0000001BIKE"</td><td>"PR0000001BIKE" Project = PR0000001</td></tr> <tr> <td>"PR1 BIKE"</td><td>"PR1 BIKE" Project = PR1</td></tr> <tr> <td>" BIKE"</td><td>" BIKE"</td></tr> <tr> <td>"BIKE"</td><td>" BIKE" Project = ""</td></tr> </tbody> </table> Pre : - Post : - Input : iBODItem - CBO item (e.g. "BIKE"). Mandatory. Output : oERPItem - LN item (e.g. " BIKE ") oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return : 0 - iBODItem converted to oERPItem <> 0 - Otherwise.	CBO item string	ERP item	"PR0000001BIKE"	"PR0000001BIKE" Project = PR0000001	"PR1 BIKE"	"PR1 BIKE" Project = PR1	" BIKE"	" BIKE"	"BIKE"	" BIKE" Project = ""
CBO item string	ERP item										
"PR0000001BIKE"	"PR0000001BIKE" Project = PR0000001										
"PR1 BIKE"	"PR1 BIKE" Project = PR1										
" BIKE"	" BIKE"										
"BIKE"	" BIKE" Project = ""										

BOD.CreateCodeDefinitionReferenceNode

DLL:	tcextbodapi
	This function is available from KB2292786 .
Syntax:	<pre> long BOD.CreateCodeDefinitionReferenceNode(domain tcncmp iCompany, domain tcmcs.str50 iListId, domain tcmcs.str100 iCode, domain tcmcs.str50 iTagName, ref domain tcmcs.long oXmlNode, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function creates a node with a reference to the CodeDefinitionBOD. Example for iListId is "Process Codes", iCode is "xxx" and iTagName is "ProcessCode": <pre> <ProcessCode listID="Process Codes" accountingEntity="aaa">xxx</ProcessCode> </pre> Pre: NA Post: NA Input: iCompany - Company. Mandatory. iListId - List ID name to be used for the listID attribute. Mandatory iCode - Code. Mandatory iTagName - Tag name. Mandatory Output: oXmlNode - Generated Node oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - OK. <> 0 - Error occurred.

BOD.CreateIDNode

DLL:	tcextbodapi
	This function is available from KB2292786 .
Syntax:	<pre> long BOD.CreateIDNode(domain tcncmp iCompany, domain tcbod.name iNoun, domain tcmcs.str256 iNounId, domain tcmcs.str100 iTagName, ref domain tcmcs.long oXmlNode, ref domain tcmcs.s999m oExceptionMessage mb, </pre>

	ref	long	oExceptionID)
Usage:	Expl:	This function creates an ID node with a reference to a BOD. The function determines the accounting entity, location and lid attributes for the NounId in the specified Noun and returns a Node with the NounId and related attributes. <pre> <ID accountingEntity="aaa" location="bbb" lid="ccc">iNounId</ID> </pre> If iTagName is filled, the node is created with the value of iTagName. Otherwise, the tag name is <ID>.	
	Pre:	NA	
	Post:	NA	
	Input:	iCompany	- Company. Mandatory.
		iNoun	- Noun name e.g. "SalesOrderBOD" or "ItemMasterCommonBOD". Mandatory
		iNounId	- Noun ID value. Mandatory
		iTagName	- Tag Name
	Output:	oXmlNode	- Generated Node for the NounId
		oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
		oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
	Return:	0	- OK.
		<> 0	- Error occurred.

BOD.CreateLastModificationPersonNode

DLL:	tcextbodapi This function is available from KB2292786 .		
Syntax:	<pre> long BOD.CreateLastModificationPersonNode (domain tcmcs.str50 iLogonCode, ref domain tcmcs.long oXmlNode, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>		
Usage:	Expl: This function creates a LastModificationPerson node based on the iLogonCode. The current LN user (logname\$) is used instead of iLogonCode if: - Input variable iLogonCode is empty. - Length of input variable iLogonCode is greater than length of domain 'tclogn' (string 16). - The iLogonCode does not exist as LN User. The LastModificationPerson node is not created if this function is called in company 0 or if no employee exists for iLogonCode in the Employee (tccom001) table. In this case oXmlNode is returned as 0 and the return value of the function is 0. XML structure: <pre> <LastModificationPerson> </pre>		

	<pre> <IDs> <ID accountingEntity="aaa" location="bbb" lid="ccc">xxx</ID> <Name> </LastModificationPerson> Pre: NA Post: NA Input: iLogonCode - The logon code of the LN User Output: oXmlNode - Generated Node oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - OK. <> 0 - Error occurred. </pre>

BOD.CreateNoteNode

DLL:	tctxbodapi This function is available from KB2292786.
Syntax:	<pre> long BOD.CreateNoteNode(domain tctxtn iTextNumber, domain tcmcs.str50 iNoteType, ref domain tcmcs.long ioXmlNode(), ref domain tcmcs.long ioSize, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Based on iTextNumber this function creates one or more Note nodes with iNoteType as attribute. The number of nodes created depends on the number of languages for which the text has been defined. <pre> <Note type="Header" languageID="en">English text</Note> <Note type="Header" languageID="nl">Dutch text</Note> <Note type="Header" languageID="de">German text</Note> </pre> This function can be called multiple times; each time with a different value for iTextNumber and iNoteType. New Note nodes will be appended to the existing array with Note nodes. Result for first call with iNoteType is "Header": <pre> <Note type="Header" languageID="en">Header text</Note> </pre> Result for second call with iNoteType is "Footer": <pre> <Note type="Header" languageID="en">Header text</Note> <Note type="Footer" languageID="en">Footer text</Note> </pre> Pre: The ioXmlNode array must be initialized as based before the function call: <pre> long ioXmlNode(1) based </pre> Post: NA

Input:	iTextNumber	- Text number. Mandatory.
	iNoteType	- Value for the type attribute in the generated Note nodes. This is mandatory if the function is called multiple times for the same ioXmlNode array. Otherwise, optional.
	ioXmlNode	- Existing Notes array; new nodes will be appended.
	ioSize	- Size of existing ioXmlNode array.
Output:	ioXmlNode	- Result with Note nodes.
	ioSize	- Size of ioXmlNode array.
	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
Return:	0	- OK.
	<> 0	- Error occurred.

BOD.ExecuteMethod

DLL:	tcextbodapi
Syntax:	<pre> long BOD.ExecuteMethod(domain tcbod.name iNoun, domain tcmcs.str30 iMethod, long iXMLRequest, ref long oXMLResponse, ref long oXMLResult, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function executes a non-batch method for a specified BOD and can only be called from the OnExecuteHook of the On<verb> (e.g. OnProcess or OnSync) method of an incoming noun. For example: if the actionCode of the incoming noun is "Add", call this function with iMethod is "Create" or if the actionCode is "Change", call this function with iMethod is "Change". Pre: NA Post: NA Input: iNoun - The (protected) Noun for which the method must be executed, e.g. "ReceiveDeliveryWarehousingBOD". Mandatory iMethod - The method to be executed, e.g. "Create" or "Change". Mandatory iXMLRequest - XML structure with request. Mandatory Output: oXMLResponse - XML structure with response (if method is executed successfully) oXMLResult - XML structure with result (in case of error) oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the

	oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Method is executed. <> 0 - Method could not be executed.

BOD.ExecuteOnAcknowledgeForChameleon

DLL:	ttextbodapi
Syntax:	<pre>long BOD.ExecuteOnAcknowledgeForChameleon(domain tcbod.name iProtectedNoun, long iXMLRequest, ref long oXMLResponse, ref long oXMLResult, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function executes the OnAcknowledge method of a protected noun of the chameleon BOD and can only be called from the OnExecuteHook of the OnAcknowledge method of an incoming public noun. The function is used to route an incoming BOD with Acknowledge verb to a protected noun. Pre: NA Post: NA Input: iProtectedNoun - The (protected) Noun for which the OnAcknowledge method must be executed, e.g. "SalesOrderInBOD". Mandatory iXMLRequest - XML structure with request. Mandatory Output: oXMLResponse - XML structure with response (if method is executed successfully) oXMLResult - XML structure with result (in case of error) oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - BOD is processed in LN. <> 0 - Error occurred while processing BOD.

BOD.ExecuteOnLoadForChameleon

DLL:	ttextbodapi

Syntax:	<pre> long BOD.ExecuteOnLoadForChameleon(domain tcbod.name iProtectedNoun, long iXMLRequest, ref long oXMLResponse, ref long oXMLResult, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function executes the OnLoad method of a protected noun of the chameleon BOD and can only be called from the OnExecuteHook of the OnLoad method of an incoming public noun. The function is used to route an incoming BOD with Load verb to a protected noun. Pre: NA Post: NA Input: iProtectedNoun - The (protected) Noun for which the OnLoad method must be executed, e.g. "SalesOrderInBOD". Mandatory Output: iXMLRequest - XML structure with request. Mandatory oXMLResponse - XML structure with response (if method is executed successfully) oXMLResult - XML structure with result (in case of error) oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - BOD is processed in LN. <> 0 - BOD could not be processed in LN

BOD.ExecuteOnProcessForChameleon

DLL:	tcextbodapi
Syntax:	<pre> long BOD.ExecuteOnProcessForChameleon(domain tcbod.name iProtectedNoun, long iXMLRequest, ref long oXMLResponse, ref long oXMLResult, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function executes the OnProcess method of a protected noun of the chameleon BOD and can only be called from the OnExecuteHook of the OnProcess method of an incoming public noun. The function is used to route an incoming BOD with Process verb to a protected noun. Pre: NA Post: NA Input: iProtectedNoun - The (protected) Noun for which the OnProcess method must be executed, e.g. "SalesOrderInBOD". Mandatory Output: iXMLRequest - XML structure with request. Mandatory oXMLResponse - XML structure with response (if method is

	executed successfully) oXMLResult - XML structure with result (in case of error) oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - BOD is processed in LN. <> 0 - BOD could not be processed in LN.

BOD.ExecuteOnSyncForChameleon

DLL:	ttextbodapi
Syntax:	<pre>long BOD.ExecuteOnSyncForChameleon(domain tcbod.name iProtectedNoun, long iXMLRequest, ref long oXMLResponse, ref long oXMLResult, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function executes the OnSync method of a protected noun of the chameleon BOD and can only be called from the OnExecuteHook of the OnSync method of an incoming public noun. The function is used to route an incoming BOD with Sync verb to a protected noun. Pre: NA Post: NA Input: iProtectedNoun - The (protected) Noun for which the OnSync method must be executed, e.g. "SalesOrderInBOD". Mandatory iXMLRequest - XML structure with request. Mandatory Output: oXMLResponse - XML structure with response (if method is executed successfully) oXMLResult - XML structure with result (in case of error) oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - BOD is processed in LN. <> 0 - BOD could not be processed in LN.

BOD.ExecuteOnUpdateForChameleon

DLL:	tcextbodapi
Syntax:	<pre> long BOD.ExecuteOnUpdateForChameleon(domain tcbod.name iProtectedNoun, long iXMLRequest, ref long oXMLResponse, ref long oXMLResult, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function executes the OnUpdate method of a protected noun of the chameleon BOD and can only be called from the OnExecuteHook of the OnUpdate method of an incoming public noun. The function is used to route an incoming BOD with Update verb to a protected noun. Pre: NA Post: NA Input: iProtectedNoun - The (protected) Noun for which the OnUpdate method must be executed, e.g. "SalesOrderInBOD". Mandatory Output: iXMLRequest - XML structure with request. Mandatory oXMLResponse - XML structure with response (if method is executed successfully) oXMLResult - XML structure with result (in case of error) oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - BOD is processed in LN. <> 0 - BOD could not be processed in LN.

BOD.ExecutePublishForChameleon

DLL:	tcextbodapi
Syntax:	<pre> long BOD.ExecutePublishForChameleon(domain tcbod.name iProtectedNoun, long iXMLRequest, ref long oXMLResponse, ref long oXMLResult, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function executes the publishing for a protected noun of the chameleon BOD and can only be called from the OnExecuteHook of the PublishEvent method of a protected noun. It translates the protected noun to the public noun and executes the PublishEvent method of the public noun Pre: NA Post: NA

	Input: iProtectedNoun - The protected Noun, which is the BOD that calls this function, e.g. "ProductionOrderSFCBOD". Mandatory iXMLRequest - XML structure with request. Mandatory Output: oXMLResponse - XML structure with response (if PublishEvent method is executed successfully) oXMLResult - XML structure with result (in case of error) oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - BOD is published from LN. <> 0 - BOD could not be published from LN.

BOD.ExecuteRequest

DLL:	tcextbodapi
Syntax:	<pre>long BOD.ExecuteRequest(long iXMLRequest, ref long oXMLAcknowledge, ref long oXMLConfirm, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function executes an incoming request for a public BOD. The incoming request should be a BOD XML file in the same format as a BOD XML file that is received from ION. This function can be used in a custom LN program that builds a BOD XML file (e.g a ProcessSalesOrder or a SyncPerson) and calls this function to process this BOD request in LN (e.g. create or update the sales order, or create an employee). The processing is similar to the Test Business Object Method (tlbct3232m000) session in which the Is BOD check box is selected. A request that is part of a batch is not supported. Tools TIV must be greater than or equal to 2430. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iXMLRequest - BOD XML structure with request. Mandatory. Output: oXMLAcknowledge - XML structure with Acknowledge BOD. This XML structure is returned if the incoming request has the Process verb. If the incoming request has another verb, 0 is returned. oXMLConfirm - XML structure with Confirm BOD. This XML structure is returned if the incoming BOD XML request was not processed successfully. It contains the errors that occurred during processing the incoming BOD XML request. If no errors occurred, 0 is returned. oExceptionMessage - A message if the return value is not equal

	oExceptionID Return: 0 <> 0	to 0. This message contains the root cause of the request failure. - An ID that refers to all error information if the return value is <> 0. It includes errors that originate from the Confirm BOD. Use the functions in Exception to get all relevant information. - The BOD request was processed successfully. - The BOD request was not processed successfully.

BOD.ExecuteShowForChameleon

DLL:	tcextbodapi
Syntax:	<pre>long BOD.ExecuteShowForChameleon(domain tcbod.name iProtectedNoun, long iXMLRequest, ref long oXMLResponse, ref long oXMLResult, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function executes the Show method of the public noun of the chameleon BOD and can only be called from the OnExecuteHook of the Show method of a protected noun. It translates the protected noun to the public noun and executes the Show method of the public noun. Then it renames the response XML to the response XML of the protected noun Pre: NA Post: NA Input: iProtectedNoun - The protected Noun, which is the BOD that calls this function, e.g. "ProductionOrderSFCBOD". Mandatory Output: iXMLRequest - XML structure with request. Mandatory oXMLResponse - XML structure with response (if method is executed successfully) oXMLResult - XML structure with result (in case of error) oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Method is executed successfully. <> 0 - Otherwise.

BOD.GetIdAccountingEntity

DLL:	tcextbodapi
Syntax:	<pre> long BOD.GetIdAccountingEntity(domain tcncmp iCompany, domain tcbod.name iNoun, long iEntityType, domain tcmcs.str30 iEntityCode, domain tcmcs.tabl iRootTable, ref domain tcbod.acen oAccountingEntity, ref boolean oAccountingEntityisSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function determines the Accounting Entity. Pre: NA Post: NA Input: iCompany - Company for which the accounting entity is determined. Mandatory iNoun - Noun: Mandatory iEntityType - Entity Type: Mandatory for transactional data BODs Possible values: 1 (Warehouse), 2 (Department), 3 (Project) iEntityCode - Entity Code: Mandatory for transactional data BODs Possible values: The warehouse, department or project. The Entity Type and Entity Code are used to determine the Tenant, AccountingEntity and Location. iRootTable - Root Table: Mandatory for master data BODs Output: oAccountingEntity - Accounting Entity oAccountingEntityisSet - Accounting Entity set (true or false). oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Accounting Entity is determined. <> 0 - Otherwise.

BOD.GetIdLocation

DLL:	tcextbodapi
Syntax:	<pre> long BOD.GetIdLocation(domain tcncmp iCompany, domain tcbod.name iNoun, long iEntityType, domain tcmcs.str30 iEntityCode, domain tcmcs.tabl iRootTable, ref domain tcbod.lctn oLocation, </pre>

	ref boolean oLocationisSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function determines the Location. Pre: NA Post: NA Input: iCompany - company for which the location is determined. Mandatory iNoun - Noun: Mandatory iEntityType - Entity Type: Mandatory for transactional data BODs Possible values: 1 (Warehouse), 2 (Department), 3 (Project) iEntityCode - Entity Code: Mandatory for transactional data BODs Possible values: The warehouse, department or project. The Entity Type and Entity Code are used to determine the Tenant, AccountingEntity and Location. iRootTable - Root Table: Mandatory for master data BODs Output: oLocation - Location oLocationisSet - Location set (true or false). oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Location is determined. <> 0 - Otherwise.

BOD.GetIdLogicalID

DLL:	tcextbodapi
Syntax:	long BOD.GetIdLogicalID(domain tcncmp iCompany, domain tcbod.name iNoun, long iEntityType, domain tcmcs.str30 iEntityCode, domain tcmcs.tabl iRootTable, ref domain tcbod.loid oLogicalID, ref boolean oLogicalIDisSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function determines the Logical ID. Pre: NA Post: NA Input: iCompany - company for which the logical Id is determined. Mandatory iNoun - Noun: Mandatory iEntityType - Entity Type: Mandatory for transactional data BODs Possible values: 1 (Warehouse), 2 (Department), 3 (Project)

	iEntityCode - Entity Code: Mandatory for transactional data BODs Possible values: The warehouse, department or project. The Entity Type and Entity Code are used to determine the Tenant, AccountingEntity and Location. iRootTable - Root Table: Mandatory for master data BODs Output: oLogicalID - LogicalID oLogicalIDisSet - LogicalID set (true or false). oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - LogicalId is determined. <> 0 - Otherwise.

BOD.GetPublishingAllowed

DLL:	tcextbodapi This function is available from KB2292786.
Syntax:	<pre>long BOD.GetPublishingAllowed(ref boolean oAllowed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function checks if BOD publishing is allowed in the current company. Pre: NA Post: NA Input: NA Output: oAllowed - true: publishing is allowed - false: publishing is not allowed oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - OK. <> 0 - Error occurred.

BOD.GetTextNumberFromNoteNode

DLL:	ttextbodapi This function is available from KB2292786 .
Syntax:	<pre> long BOD.GetTextNumberFromNoteNode(ref domain tcmcs.long iXmlNode(), domain tcmcs.long iSize, domain tcmcs.str50 iNoteType, domain tclang iSystemLanguage, ref domain tctxtn ioTextNumber, ref boolean oTextIsSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Based on iNoteType this function gets the text and creates a text number with text or updates an existing text number with the text provided in the Note elements in iXmlNode. Example: <pre> <Note type="Header" languageID="en">English text</Note> <Note type="Header" languageID="nl">Dutch text</Note> <Note type="Header" languageID="de">German text</Note> <Note type="Footer" languageID="en">English Footer text</Note> </pre> This function can be used to handle the texts of the inbound BOD. The inbound BOD will take care of the transaction handling. Pre: The iXmlNode array must be intialized and filled before the function call. Post: NA Input: - iXmlNode - Array with Note nodes. Mandatory. - iSize - Size of Notes array. Mandatory. - iNoteType - Value of the type attribute in the Notes array. If specified, only Notes with this value and Notes without a type attribute will be selected to get the text. - iSystemLanguage - If the languageID attribute is not present in the Notes array, the iSystemLanguage is used as fall back. - ioTextNumber - Text number. If specified, this text number is used. Otherwise a new text number is generated. Output: - ioTextNumber - The Text number that was updated or created. - oTextIsSet - Indicator if Text was updated/created or not. - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - OK. <> 0 - Error occurred.

BOD.GetValueFromNodeElement

DLL:	ttextbodapi		
	This function is available from KB2292786 .		
Syntax:	<pre> long BOD.GetValueFromNodeElement (long iXmlNode, domain tcmcs.str132 iPath, domain tcmcs.str14 iDomainCode, ref domain tcmcs.s512m oValue mb, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>		
Usage:	Expl: This function gets the value from a specific element in a node. It uses the path to locate the element in the node. If the domain code is supplied, the element value is checked against the domain definition and an error is returned if the value does not match the domain definition. Pre: NA Post: NA Input : iXmlNode - The xml node to search in. Mandatory iPath - The path in the xml node e.g. "<Shipment><ShipmentHeader><DocumentID><ID>". Mandatory iDomainCode - Domain code. If supplied, the element value is checked against the domain definition. Output: oValue - The value that is extracted from the node. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - OK. <> 0 - Error occurred.		

BOD.HandleStagingAfterProcessingIncomingRequest

DLL:	ttextbodapi		
Syntax:	<pre> long BOD.HandleStagingAfterProcessingIncomingRequest (domain tcbod.name iNoun, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>		
Usage:	Expl: This function publishes the staged BODs that were triggered from the processing of the incoming request and resets the parameter that enabled the staging of outgoing BODs in the BOD.HandleStagingBeforeProcessingIncomingRequest() function. Preferably, it should be called in the AfterExecuteHook of every OnEvent section, e.g. the OnProcess, OnLoad, OnSync,		

	OnShow, OnAcknowledge etc. Pre: Before the incoming BOD is processed, function BOD.HandleStagingBeforeProcessingIncomingRequest() must be called. Post: NA Input: iNoun - The noun of the incoming request e.g. "PurchaseOrderBOD" Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Staged BODs are published. <> 0 - Staged BODs could not published.

BOD.HandleStagingBeforeProcessingIncomingRequest

DLL:	tcextbodapi
Syntax:	<pre>long BOD.HandleStagingBeforeProcessingIncomingRequest(domain tcbod.name iNoun, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function sets a parameter that enables the staging of outgoing BODs that are triggered from the processing of the incoming request. Preferably, it should be called in the BeforeExecuteHook of every OnEvent section, e.g. the OnProcess, OnLoad, OnSync, OnShow, OnAcknowledge etc. Pre: NA Post: After the incoming BOD is processed, function BOD.HandleStagingAfterProcessingIncomingRequest() must be called Input: iNoun - The noun of the incoming request e.g. "PurchaseOrderBOD" Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Parameter that enables the staging is set. <> 0 - Otherwise.

BOD.InterruptPublishing

DLL:	tcextbodapi This function is available from KB2267552 .												
Syntax:	<pre> long BOD.InterruptPublishing(boolean iAllBods, domain tcbod.name iNoun, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>												
Usage:	Expl: This function interrupts the publishing of all BODs or one specified BOD in the current process. If parallel processing is implemented, the BOD publishing will be interrupted in the parallel processes as well. For example: session Calculate Standard Cost publishes the ItemMasterBOD for each item for which the cost price is updated. If you don't need the ItemMasterBOD from this session, you can call this function in the Before Command hook of the main processing with iNoun = "ItemMasterCommonBOD". You can resume the publishing by calling function BOD.ResumePublishing(). If a process can publish multiple BODs (e.g. SalesOrderBOD, CustomerReturnBOD and ShipmentBOD) and you want to interrupt the publishing of two BODs, you can call this function two times and specify iAllBods as false and iNoun as the BOD for which the publishing must be interrupted. Pre: N/A Post: Function BOD.ResumePublishing() must be called to resume the publishing. Output: <table border="0"> <tr> <td>iAllBods</td><td>- true: The publishing of all BODs in the current process is interrupted. false: The publishing of the specified BOD in the current process is interrupted.</td></tr> <tr> <td>iNoun</td><td>- The (protected) Noun for which the publishing must be interrupted. Example: ItemMasterCommonBOD. Mandatory if iAllBods is false. Not needed / ignored if iAllBods is true.</td></tr> <tr> <td>oExceptionMessage</td><td>- A message if the return value is not equal to 0. This message contains the root cause of the request failure.</td></tr> <tr> <td>oExceptionID</td><td>- An ID that refers to all error information if the return value is <> 0. Use the functions in Exception to get all relevant information.</td></tr> </table> Return: <table border="0"> <tr> <td>0</td><td>- The action to interrupt the publishing was set successfully.</td></tr> <tr> <td><> 0</td><td>- The action to interrupt the publishing can not be set.</td></tr> </table>	iAllBods	- true: The publishing of all BODs in the current process is interrupted. false: The publishing of the specified BOD in the current process is interrupted.	iNoun	- The (protected) Noun for which the publishing must be interrupted. Example: ItemMasterCommonBOD. Mandatory if iAllBods is false. Not needed / ignored if iAllBods is true.	oExceptionMessage	- A message if the return value is not equal to 0. This message contains the root cause of the request failure.	oExceptionID	- An ID that refers to all error information if the return value is <> 0. Use the functions in Exception to get all relevant information.	0	- The action to interrupt the publishing was set successfully.	<> 0	- The action to interrupt the publishing can not be set.
iAllBods	- true: The publishing of all BODs in the current process is interrupted. false: The publishing of the specified BOD in the current process is interrupted.												
iNoun	- The (protected) Noun for which the publishing must be interrupted. Example: ItemMasterCommonBOD. Mandatory if iAllBods is false. Not needed / ignored if iAllBods is true.												
oExceptionMessage	- A message if the return value is not equal to 0. This message contains the root cause of the request failure.												
oExceptionID	- An ID that refers to all error information if the return value is <> 0. Use the functions in Exception to get all relevant information.												
0	- The action to interrupt the publishing was set successfully.												
<> 0	- The action to interrupt the publishing can not be set.												

BOD.Publish

DLL:	tcextbodapi
Syntax:	<pre> long BOD.Publish(domain tcbod.name iNoun, domain tcmcs.tabl iRootTable, domain tcmcs.str30 iActionCode, long iEntityType, domain tcmcs.str30 iEntityCode, domain tcmcs.str100 iDocumentId, domain tcmcs.str30 iProcessingAction, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function publishes any BOD. Pre: Transaction handling is needed. Post: Transaction handling is needed: A commit or abort must be done. Input: iNoun - The (protected) Noun. Examples: XXXCustomBOD, ItemMasterCommonBOD. Mandatory iRootTable - Root Table: Mandatory for master data BODs iActionCode - Action Code: Mandatory Possible values: "Add", "Change", "Delete", "Replace", "Canceled". iEntityType - Entity Type: Mandatory for transactional data BODs Possible values: 1 (Warehouse), 2 (Department), 3 (Project) iEntityCode - Entity Code: Mandatory for transactional data BODs Possible values: The warehouse, department or project. The Entity Type and Entity Code are used to determine the Tenant, AccountingEntity and Location. iDocumentId - Document ID: Mandatory iProcessingAction - Processing Action: Mandatory Possible values: "OnlyStage", "StageOrPublish", "OnlyPublish". Variable arguments are Identifiers - The first Identifier is Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - BOD is published. <> 0 - BOD could not be published

BOD.PublishLNMessage

DLL:	tcextbodapi

Syntax:	<pre> long BOD.PublishLNMessage(domain tcmcs.str30m iMessageType mb, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function publishes the LnMessageBOD with a specific MessageType. The first three arguments are mandatory, in the given order. iMessageType can f.i. have value "PurchaseScheduleLine". The other arguments should be 'xml_elementname - xml_elementvalue' pairs, e.g.: <pre> "Description", tfgld011.desc, </pre> The number of input arguments can vary. The DocumentID is mandatory as one of the xml_element names: <pre> "DocumentID", table.key1, "DocumentID", table.key2, </pre> The maximum composed DocumentID_ID string length is 132. If the "ActionCode" is not set, the default "Add" is used. If "DisplayID" is not set the "DocumentID" is used to create the DisplayID. To create a user area node a UserAreaName, UserAreaType and UserAreaValue are mandatory. UserAreaDescription, UserAreaStartDate and UserAreaEndDate can also be used. --> In the function call, the same order must be used! Allowed values for UserAreaType are: - "DateTimeType" - "IndicatorType" - "NumericType" - "StringType" If an xml_elementname is not in the list of possible xml_elementnames, the xml_element value is added to the "MessageTypeDetails" node, e.g.: input arguments: <pre> "MessageType" "TestType", "Element.1" field.1, "Element.2" field.2, </pre> result node: <pre> <MessageTypeDetails> <TestTypeGroup> <Element.1>field.1</Element.1> <Element.2>field.2</Element.2> </TestTypeGroup> </MessageTypeDetails> </pre> xml_elementvalues should apply to the following rules: utc dates should be converted to string representation in ISO. e.g.: "EffectiveDateTime", utc.to.iso(datefield, UTC_ISO_Z) Use the english description associated with a specific value in an enumerated domain. e.g.: "TransactionCategory", enum.descr\$("tfgld.catg", tfgld011.catg, "2") Attributes of an xml_element are reflected using an @ and must be placed directly under the xml_element on which they are related, e.g.: <pre> "PurchaseInvoiceAmount", "10.000", "PurchaseInvoiceAmount@currencyID", "EUR", </pre>

	<pre> <PurchaseInvoiceAmount currencyID="EUR">10.000 </PurchaseInvoiceAmount> </pre> Possible xml_element names: - "ActionCode" - "Authorization" - "Description" - "DocumentReference" - "DocumentReference_ID" - "DocumentReference_LineNumber" - "DocumentReference_ScheduleLineNumber" - "DisplayID" - "DistributionGroup_Contact" - "DistributionGroup_ContactGroup" - "DistributionGroup_Email" - "DistributionGroup_Person" - "DistributionGroup_PersonGroup" - "DocumentID" - "Message_Description" - "Message_ID" - "MessageType" - "Note" (the xml_element value must be a text number) - "Status_Code" - "Status_EffectiveDateTime" - "Status_ArchiveIndicator" - "UserAreaDescription" - "UserAreaEndDate" - "UserAreaName" - "UserAreaStartDate" - "UserAreaType" - "UserAreaValue" Pre: Transaction handling is needed. Post: Transaction handling is needed: A commit or abort must be done. Input: iMessageType: Mandatory Variable arguments are pairs of two arguments with name and value. At least one argument with name "DocumentID" is Mandatory. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - BOD is published. <> 0 - BOD could not be publised.

BOD.PublishWithBatch

DLL:	tcextbodapi

	This function is available from KB3621405 .
Syntax:	<pre> long BOD.PublishWithBatch(domain tcbod.name iNoun, domain tcmcs.tabl iRootTable, domain tcmcs.str30 iActionCode, long iEntityType, domain tcmcs.str30 iEntityCode, domain tcmcs.str100 iDocumentId, domain tcmcs.str32 iBatch, domain tcmcs.long iBatchSequence, domain tcmcs.long iBatchSize, domain tcmcs.str30 iProcessingAction, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: This function publishes any BOD that is part of a Batch with BODs. Pre: Transaction handling is needed. Post: Transaction handling is needed: A commit or abort must be done. Input: iNoun - The (protected) Noun. Examples: XXXCustomBOD, ItemMasterCommonBOD. Mandatory iRootTable - Root Table: Mandatory for master data BODs iActionCode - Action Code: Mandatory Possible values: "Add", "Change", "Delete", "Replace", "Canceled". iEntityType - Entity Type: Mandatory for transactional data BODs Possible values: 1 (Warehouse), 2 (Department), 3 (Project) iEntityCode - Entity Code: Mandatory for transactional data BODs Possible values: The warehouse, department or project. The Entity Type and Entity Code are used to determine the Tenant, AccountingEntity and Location. iDocumentId - Document ID: Mandatory iBatch - Batch to which the BOD belongs. Mandatory iBatchSequence - The sequence number in the batch. Mandatory iBatchSize - The size (total number of BODs) of the batch. This must be provided in one of the BODs in the batch; normally with the last BOD of the batch. iProcessingAction - Processing Action: Mandatory Possible values: "OnlyStage", "StageOrPublish", "OnlyPublish". Variable arguments are Identifiers - The first Identifier is Mandatory Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - BOD is published. <> 0 - BOD could not be published

BOD.ResumePublishing

DLL:	tcextbodapi		
	This function is available from KB2267552 .		
Syntax:	long BOD.ResumePublishing(boolean domain ref ref tcbod.name tcmcs.s999m long iAllBods, iNoun, oExceptionMessage mb, oExceptionID)		
Usage:	Expl: Pre: Post: Output: iAllBods iNoun oExceptionMessage oExceptionID - true: The publishing of all BODs in the current process is resumed. false: The publishing of the specified BOD in the current process is resumed. Value false is not allowed if function BOD.InterruptPublishing() was called with iAllBods as true. - The (protected) Noun for which the publishing must be resumed. Mandatory if iAllBods is false. Not needed / ignored if iAllBods is true. - A message if the return value is not equal to 0. This message contains the root cause of the request failure. - An ID that refers to all error information if the return value is <> 0. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - The action to resume the publishing was set successfully. - The action to resume the publishing can not be set.		

Public Interfaces for BDE

The following functions are available:

[BDE.ExecuteMethod](#)

BDE.ExecuteMethod

DLL:	ttextbdeapi
Syntax:	<pre> long BDE.ExecuteMethod(domain tcbod.name iBusinessObject, domain tcmcs.str30 iMethod, long long iXMLRequest, ref long oXMLResponse, ref long oXMLResult, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function executes a method for a specified public BDE. Pre: If the method to be executed updates the database, a db.retry.point() must have been set in the calling program. Post: If the method to be executed updates the database, a commit.transaction() must be done by the calling program. However, in case of a non-zero return value, an abort.transaction() must be done, otherwise updates may have been done partially, which, once committed, may cause significant data corruption. Input: iBusinessObject - The Business Object for which the method must be executed, e.g. "Item_v3". Mandatory iMethod - The method to be executed, e.g. "Create" or "Change". Mandatory iXMLRequest - XML structure with request. Mandatory Output: oXMLResponse - XML structure with response (if method is executed successfully) oXMLResult - XML structure with result (in case of error) oExceptionMessage - A message if the return value is not equal to 0. This message contains the root cause of the of the method failure. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get the error messages. Note that more detailed error information is in oXMLResult. Return values: 0 - method is executed successfully <> 0 - on errors

Chapter 31 Public Interfaces for Factory Track

Public Interfaces for FactoryTrackBDE

The following functions are available:

[FactoryTrackBDE.CreateElement](#)

[FactoryTrackBDE.CreateNode](#)

FactoryTrackBDE.CreateElement

DLL:	brextbdeapi
Syntax:	<pre> long FactoryTrackBDE.CreateElement(const string iElementID(), const string iDomain(), long iParentNode, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: Use this public interface to create a data element in the input node. Public Interface FactoryTrackBDE.CreateNode can be used to create a parent node for this data element. Example: Create custom element "OrderNumber" in OutputNode. <pre> long ReturnValue domain tcorno OrderNumber ReturnValue = FactoryTrackBDE.CreateElement("OrderNumber", "tcqiv1", OutputNode, ExceptionMessage, ExceptionID, OrderNumber) </pre> This will add this element Node to the OutputNode (assuming

	<pre> OrderNumber = "AAA000001"): <OrderNumber Type="String" Domain="tcorno">AAA000001</OrderNumber> </pre> Pre: N.A. Post: N.A. Input: - iElementID - The name of the custom element: Mandatory. Dots are truncated. - iDomain - The domain of the custom element: Mandatory. - iParentNode - The parent XML node of the element that is being created: Mandatory. ... (1 argument) - The value of the XML element: Mandatory. Output: - oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. - oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Success <> 0 - Failure

FactoryTrackBDE.CreateNode

DLL:	brextbdeapi
Syntax:	<pre> long FactoryTrackBDE.CreateNode(const string iNodeID(), long iBDEResponseArgument, ref long oNode, ref domain tcmcs.s999m ref long oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this public interface to create a child in the response node of a custom Factory Track BDE. Public interface FactoryTrackBDE.CreateElement can be used to create elements in this node. Example: Custom BDE "FactoryTrackTransaction" has method "DoTransaction". The XML tree of the response node is by default <pre> <DoTransactionResponse> <DataArea> <FactoryTrackTransaction/> </DataArea> </DoTransactionResponse> </pre>

To create child node "ReqOutput", the public interface FactoryTrackBDE.CreateNode can be executed as follows.

```

long          ReturnValue
long          Node

```

```

ReturnValue = FactoryTrackBDE.CreateNode(
    "ReqOutput",
    bl.ResponseArgument,
    Node,
    ExceptionMessage,
    ExceptionID)

```

Then the XML tree of the response node (bl.ResponseArgument) is

```

<DoTransactionResponse>
  <DataArea>
    <FactoryTrackTransaction>
      <ReqOutput/>
    </FactoryTrackTransaction>
  </DataArea>
</DoTransactionResponse>

```

Pre: N.A.

Post: N.A.

Input:

```

iNodeID          - The name of the node to be created.
iBDEResponseArgument - The response argument of the BDE. This
                      is the XML node bl.ResponseArgument in
                      the business object layer of the BDE.

```

Output:

```

oNode            - A reference to the newly created node.
oExceptionMessage - The last message if the return
                    value is not equal to 0.
                    If more than one message is given,
                    these are present in the oExceptionID.
oExceptionID      - An ID that refers to all error
                    information. Use the functions in
                    Exception to get all relevant
                    information.

```

Return:

```

0                - Success
<> 0            - Failure

```

Public Interfaces for FactoryTrackQuery

The following functions are available:

[FactoryTrackQuery.CreateElement](#)

[FactoryTrackQuery.CreateRow](#)

[FactoryTrackQuery.GetElement](#)

FactoryTrackQuery.CreateElement

DLL:	brextqryapi
Syntax:	<pre>long FactoryTrackQuery.CreateElement(const string iElementID(), const string iDomain(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...)</pre>
Usage:	Expl: Use this public interface to create a custom element in the current query output row. This public interface can only be used in the context of process extension 'FactoryTrackQuery.QueryExtend'. If this public interface is used in the context of an extension on a standard query, the attribute EXT="yes" is linked to the custom element. Example: <pre> long ReturnValue ReturnValue = FactoryTrackQuery.CreateElement("InventoryOnHand", "tcqiv1", ExceptionMessage, ExceptionID, whinr140.qhnd) This will add the Element Node (assuming whinr140.qhnd = 10.5) <InventoryOnHand Type="Double" Domain="tcqiv1" EXT="yes">10.5</InventoryOnHand></pre> Example: <pre> domain tcmcs.str20 InventoryDate long ReturnValue InventoryDate = sprintf\$("%u(%04Y/%02m/%02d) ", whinr140.idat) ReturnValue = FactoryTrackQuery.CreateElement("InventoryDate", "tcmcs.str20", ExceptionMessage, ExceptionID, InventoryDate, whinr140.idat) This will add the Element Node (assuming whinr140.idat contains</pre>

	<pre> date 04/25/2018 13:51:40 in UTC-long format) <InventoryDate Type="String" Domain="tcmcs.str20" Date="04/25/2018" Time="13:51:40" EXT="yes">2018/04/25</InventoryDate> </pre>
	<pre> Pre: N.A. Post: N.A. Input: iElementID - The name of the custom element: Mandatory. Dots are truncated. iDomain - The domain of the custom element: Mandatory. ... (1 or 2 arguments) - The value of the custom element and optionally the date and time in long format. If the date and time argument is present, then date and time attributes will be added to the element. If the date and time argument equals zero, the date and time attributes will be empty. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Success <> 0 - Failure </pre>

FactoryTrackQuery.CreateRow

DLL:	brextqryapi
Syntax:	<pre> long FactoryTrackQuery.CreateRow(ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: Use this public interface to create a new output row. This public interface can only be used in the context of process extension 'FactoryTrackQuery.QueryExtend' for custom queries. Example: <pre> long ReturnValue ReturnValue = FactoryTrackQuery.CreateRow(ExceptionMessage, ExceptionID) </pre>

	<pre> Pre: N.A. Post: N.A. Input: N.A. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Success <> 0 - Failure </pre>

FactoryTrackQuery.GetElement

DLL:	brextqryapi
Syntax:	<pre> long FactoryTrackQuery.GetElement(const string iElementID(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...) </pre>
Usage:	Expl: Use this public interface to get the specified element node from the current query output row. This public interface can only be used in the context of process extension 'FactoryTrackQuery.QueryExtend'. Example: If the current query output row contains this element node, <pre> <itemcode Type="String" Domain="tcitem"> MAIN ITEM</itemcode> </pre> then this can be read by this public interface as follows. <pre> long ReturnValue domain tcitem Item ReturnValue = FactoryTrackQuery.GetElement("itemcode", ExceptionMessage, ExceptionID Item) Now Item = " MAIN ITEM". </pre> Example: If the current query output row contains this element node,

	<pre><outdate Type="String" Domain="tcmcs.str20" Date="04/25/2018" Time="13:51:40">2018/04/25</outdate></pre>												
	then this can be read by this public interface as follows.												
	<table><tr><td></td><td>long</td><td>ReturnValue</td></tr><tr><td>domain</td><td>tcmcs.str20</td><td>OutDate</td></tr><tr><td>domain</td><td>tcmcs.str10</td><td>DateAttribute</td></tr><tr><td>domain</td><td>tcmcs.str8</td><td>TimeAttribute</td></tr></table>		long	ReturnValue	domain	tcmcs.str20	OutDate	domain	tcmcs.str10	DateAttribute	domain	tcmcs.str8	TimeAttribute
	long	ReturnValue											
domain	tcmcs.str20	OutDate											
domain	tcmcs.str10	DateAttribute											
domain	tcmcs.str8	TimeAttribute											
	<pre>ReturnValue = FactoryTrackQuery.GetElement("outdate", ExceptionMessage, ExceptionID OutDate, DateAttribute, TimeAttribute)</pre>												
	Now OutDate = "2018/04/25", DateAttribute = "04/25/2018", and TimeAttribute = "13:51:40".												
	Pre: N.A. Post: N.A.												
	Input: <table><tr><td>iElementID</td><td>- Mandatory. The name of the element node to be read.</td></tr></table>	iElementID	- Mandatory. The name of the element node to be read.										
iElementID	- Mandatory. The name of the element node to be read.												
	Output: <table><tr><td>oExceptionMessage</td><td>- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.</td></tr><tr><td>oExceptionID</td><td>- An ID that refers to all error information. Use the functions in Exception to get all relevant information.</td></tr><tr><td>... (1 to 3 arguments)</td><td>- The data of the element node and optionally the date and time in string format. Date and time can only be read if the date and time attributes are present in the element node.</td></tr></table>	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.	... (1 to 3 arguments)	- The data of the element node and optionally the date and time in string format. Date and time can only be read if the date and time attributes are present in the element node.						
oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.												
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.												
... (1 to 3 arguments)	- The data of the element node and optionally the date and time in string format. Date and time can only be read if the date and time attributes are present in the element node.												
	Return: <table><tr><td>0</td><td>- Success</td></tr><tr><td><> 0</td><td>- Failure</td></tr></table>	0	- Success	<> 0	- Failure								
0	- Success												
<> 0	- Failure												

Chapter 32 Public Interfaces for Project

Public Interfaces for Project

The following functions are available:

[Project.Activate](#)

[Project.Start360](#)

[Project.StartCopy](#)

Project.Activate

DLL:	tpextpdmapi
Syntax:	<pre> long Project.Activate(domain tccprj iProject, boolean iUserInteraction, ref boolean oCanceledbyUser, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function will Activate the given Project. After the Project is Activated, the next Automatic Steps in the Project Procedure will be executed. This function must not be called within a logical transaction as this function will have its own transaction handling. Pre: There should be no pending logical transaction before calling this function. Post: No need to commit or abort the process, that is handled within the function. Input: iProject - The Project that needs to be Activated (Mandatory). iUserInteraction Possible values are: TRUE - The function is called from a LN Session, the user could be asked to confirm certain exceptions (like the project is not linked to a contract or a capital project has no assets). FALSE - The function is called from a background process, the project activation continues without user interaction.

	Note that, when this argument is true, the user may reject certain exceptions. As this is by decision of the user, it is not considered an error and the return value of the public interface is zero (no error) if this occurs. To allow the calling program to recognize and handle this situation, the output argument oCanceledbyUser will be set to true in this case. Output: oCanceledbyUser - If this public interface was called with iUserInteraction = true and the user rejects an exception, this argument is returned as true to indicate that the project activation was canceled by the user. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - The Project is Activated, or the activation was canceled by the user. <> 0 - An error occurred

Project.Start360

DLL:	tpextpdmap
Syntax:	<pre> long Project.Start360(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcyesno iSetFilterFields, domain tcemno iProjectManager, domain tccwoc iProjectManagementOffice, domain tccprj iProject, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl This function starts the Project 360 session (tpdpm6500m100). Input: iStartMode Specifies the start mode for the session (Mandatory). Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are:

	"AllProjects": All projects are displayed "OpenProjects": Only open projects are displayed iSessionIndex Specifies the session index that is to be used. iQueryExtend A specific query to be used when zooming to this session. iSetFilterFields: Used to set a filter on the projects for a Project Manager or a Project Management Office. Possible values are: Yes: A filter on the projects for a Project Manager or a Project Management Office can be applied. No: No filter for Project Manager or Project Management Office will be applied. Empty: If not set, the saved session defaults will be used and applied for these filter fields. iProjectManager: The Project Manager to be used with iSetFilterFields. iProjectManagementOffice: The Project Management Office to be used with iSetFilterFields. i.project: To filter on a specific Project. Make sure that the project is inrange of the iStartFilter, iQueryExtend and SetFilterFields selections. Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

Project.StartCopy

DLL:	tpextpdmapi
Syntax:	<pre>long Project.StartCopy(long iStartMode, domain tccprj iSourceProject, ref domain tccprj oTargetProject, ref domain tcdsca oTargetProjectDescription mb, ref boolean oProjectCopied, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...)</pre>
Usage:	Expl This function starts the Copy Project session (tppdm7840m000). The source Project and the most relevant fields for a new Project can be passed to the session. Other fields can be set in the session or one can use the Saved Defaults or use the Defaults Saved to Personalization.

	Input: iStartMode Specifies the start mode for the session (Mandatory). Possible values are: MODAL - The parent session is blocked until the child session exits. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iSourceProject The Project to be copied ... Additional input, specifies the option in a string, followed by the required value in the proper format. Possible options are: TargetProject A string containing the Project code of the new Project or a Project Series to generate a new code TargetProjectDescription A string for the description of the new Project SubProject A boolean to indicate a Sub Project: True when the new Project must be linked to a Main Project False when the new Project is a Single Project MainProject A string containing the Project code of the Main Project in case of a Sub Project ProjectAddress A string for the Project Address code ShipfromAddress A string for the standard Ship-from Address code for project deliveries ProjectManager A string containing the code of the Manager of the Project ProjectManagementOffice A string containing the code of the Office responsible for the Project EnterpriseUnit A string containing the code of the Enterprise Unit to which the Project belongs PrioritySupplyWarehouse A string containing the code of the general Warehouse for receiving and for deliveries to the Project StartDate A long UTC value for Project Start Date FinishDate A long UTC value for Project Finish Date Example: The Copy Project session will be started and has the source Project filled, with a target project series for a new 'Single Project', with a description and a start date: Project.StartCopy(* Fixed arguments: start.mode, - input source.project, - input target.project, - output target.project.description, - output project.copied, - output exception.message, - output exception.id, - output * Variable arguments: "TargetProject", - input project.series, - input "TargetProjectDescription", - input project.description, - input "SubProject", - input true, - input "StartDate", - input

	<pre> date.time.utc) - input </pre>
	Output: Note: the session must be started modal to have the copy results returned in the output. oTargetProject - The created Project oTargetProjectDescription - The Description of the created Project oProjectCopied - True, if the Project has been copied - False, otherwise oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - Error

Public Interfaces for ProjectContract

The following functions are available:

[ProjectContract.StartMultiMain](#)

ProjectContract.StartMultiMain

DLL:	tpextctmapi
Syntax:	<pre> long ProjectContract.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccono iProjectContract, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the Multi-Main session "Contract" (tpctm1600m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the

	child session exists. The session will be started as a zoom session. Parent and child are parallel sessions that can be manipulated simultaneously. MODELESS - iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend A specific query to be used when zooming to this session. iProjectContract Contract Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - An error occurred

Public Interfaces for ProjectBid

The following functions are available:

[ProjectBid.StartMultiMain](#)

ProjectBid.StartMultiMain

DLL:	tpextestapi
Syntax:	<pre> long ProjectBid.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccprj iProject, domain tpest.vers iVersion, domain tpest.bnum iBidNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the Project Multi-Main session "Bid"

	<pre> (tpest3600m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend A specific query to be used when zooming to this session. iProject Project iVersion Version iBidNumber Bid Number Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - An error occurred </pre>

Public Interfaces for ProjectEstimate

The following functions are available:

[ProjectEstimate.GenerateStructuralElements](#)

ProjectEstimate.GenerateStructuralElements

DLL:	tpextestapi
Syntax:	long ProjectEstimate.GenerateStructuralElements(domain tccprj iProject,

	<pre> domain tpest.vers iEstimateVersionFrom, domain tpest.vers iEstimateVersionTo, domain tpest.esid iEstimateStructureFrom, domain tpest.esid iEstimateStructureTo, domain tccpcp iCostComponentFrom, domain tccpcp iCostComponentTo, domain tpptc.cstl iExtensionFrom, domain tpptc.cstl iExtensionTo, domain tppdm.cspa iBudgetTopElement, boolean iUpdateFreeVersions, boolean iUpdateActualVersions, boolean iUpdateOnlyUsedStructures, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function updates the Estimate Structures for a given Project and its Estimate Versions as in session "Generate Structural Elements" (tpest1220m000). The Structural Elements of the Estimate Structures are generated from the applicable sources like Activities, Elements, Cost Components, Extensions and User Defined Structures. Pre: This function has its own transaction management. There should be no pending logical transaction before calling this function. Post: There is no need to execute an abort or commit after calling this function, that is handled within the function. Input: - iProject - Project: Mandatory - iEstimateVersionFrom - Estimate Version From - iEstimateVersionTo - Estimate Version To - iEstimateStructureFrom - Estimate Structure From - iEstimateStructureTo - Estimate Structure To - iCostComponentFrom - Cost Component From Only relevant when Estimate Structures of type 'Cost Component' are selected. - iCostComponentTo - Cost Component To Only relevant when Estimate Structures of type 'Cost Component' are selected. - iExtensionFrom - Extension From Only relevant when Estimate Structures of type 'Extension' are selected. - iExtensionTo - Extension To Only relevant when Estimate Structures of type 'Extension' are selected. - iBudgetTopElement - Budget Top Element: Optional The Top Element for structures of type Element. Only relevant when only one Estimate Version is selected, and Estimate Structures of type 'Element' are selected. - iUpdateFreeVersions - Update Free Versions: Mandatory Update Estimate Structures of Estimate Versions with Status 'Free'. Possible values: True : Update Free Versions False: Do not update Free versions - iUpdateActualVersions - Update Actual Versions: Mandatory Update Estimate Structures of Estimate Versions with Status 'Actual'. Possible values: True : Update Actual Versions False: Do not update Actual Versions - iUpdateOnlyUsedStructures - Update Only Used Structures: Mandatory

	Only update Estimate Structures which are used in an Estimate Version. Possible values: True : Update only Estimate Structures which are used in an Estimate Version False: Update all Estimate Structures Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Successful <> 0 - An error occurred

Public Interfaces for ProjectContractDeliverable

The following functions are available:

[ProjectContractDeliverable.AddCPQConfiguration](#)

ProjectContractDeliverable.AddCPQConfiguration

DLL:	tpextpdmapi
Syntax:	<pre>long ProjectContractDeliverable.AddCPQConfiguration(domain tccono iContract, domain tpctm.cnln iContractLine, domain tcpono iDeliverable, domain tcitem iConfigurableItem, domain tcmcs.str60 iConfigurationID, domain tcmcs.str60 iConfigurationDetailID, ref domain tccpva oProductVariant, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function creates a Product Variant in LN based on the supplied CPQ Configuration for the given Contract Deliverable. The created Product Variant will be updated on the Contract Deliverable. Checks: Before allowing the Contract Deliverable to be updated with a

	Product Variant some checks are done. These checks are mostly similar to the checks done when configuring an item manually in session 'Contract Deliverables' (tppdm7100m100) using the form command "Configure Product Variant". The checks include: - The Infor Configurator (CPQ) must be implemented - The Contract Deliverable may not be checked-out for OCM / Workflow - The supplied item in input argument iConfigurableItem must be equal to the item on the Contract Deliverable - The item on the Contract Deliverable must not be configured yet, meaning that the value of the Product Variant must be empty (zero) - The item must: - have Item Type Generic - have property "With PCS" set to No - not be an Assembly item - have CPQ assigned as Configurator - Authorization and Security settings for the Contract Deliverable must allow the USE of: - Contract - Project - Item - The Contract Deliverable must: - have status Free - not have a Schedule - not be a Backorder - not be a Return Deliverable Restrictions: This function must only be called from a BOD context. Pre: Retry point must be set. The Contract Deliverable to be updated must already exist. Post: Commit or abort transaction. Input: iContract - Contract: Mandatory iContractLine - Contract Line: Mandatory iDeliverable - Deliverable: Mandatory iConfigurableItem - Configurable Item: Mandatory The Item code to be configured. iConfigurationID - Configuration ID: Mandatory The Configuration ID with which the configuration is stored in CRM/EQ (Enterprise Quoting). iConfigurationDetailID - Configuration Detail ID: Mandatory The Configuration Detail ID with which the configuration is stored in CRM/EQ. Output: oProductVariant - Product Variant The new created Product Variant. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Successful <> 0 - An error occurred

Public Interfaces for ProjectActivityBudget

The following functions are available:

[ProjectActivityBudget.AddCPQConfiguration](#)

ProjectActivityBudget.AddCPQConfiguration

DLL:	tpextptcapi
Syntax:	<pre> long ProjectActivityBudget.AddCPQConfiguration(domain tccprj iProject, domain tppss.cpla iPlan, domain tppdm.cact iActivity, domain tppdm.serd iSequenceNumber, domain tcitem iConfigurableItem, domain tcmcs.str60 iConfigurationID, domain tcmcs.str60 iConfigurationDetailID, ref domain tccpva oProductVariant, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function creates a Product Variant in LN based on the supplied CPQ Configuration for the given Activity Budget Line (Material). The created Product Variant will be updated on the Activity Budget Line. Checks: Before allowing the Activity Budget Line to be updated with a Product Variant some checks are done. These checks are mostly similar to the checks done when configuring an item manually in session 'Activity Budget (Material)' (tpptc2110m000) using the form command "Configure". The checks include: - The Infor Configurator (CPQ) must be implemented - The supplied item in input argument iConfigurableItem must be equal to the item on the Activity Budget Line - The item on the Activity Budget Line must not be configured yet, meaning that the value of the Product Variant must be empty (zero) - The item must: - have Item Type Generic - have CPQ assigned as Configurator - Authorization and Security settings for the Activity Budget Line must allow USE of: - Project - Item

	Restrictions: This function must only be called from a BOD context. Pre: Retry point must be set. The Activity Budget Line to be updated must already exist. Post: Commit or abort transaction. Input: iProject - Project: Mandatory iPlan - Plan: Optional If the Plan is not specified, the default or actual Plan of the Project will be used. iActivity - Activity: Mandatory iSequenceNumber - Sequence Number: Mandatory The Line number of the Budget Line. iConfigurableItem - Configurable Item: Mandatory The Item code to be configured. iConfigurationID - Configuration ID: Mandatory The Configuration ID with which the configuration is stored in CRM/EQ (Enterprise Quoting). iConfigurationDetailID - Configuration Detail ID: Mandatory The Configuration Detail ID with which the configuration is stored in CRM/EQ. Output: oProductVariant - Product Variant The new created Product Variant. oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Successful <> 0 - An error occurred

Public Interfaces for ProjectActivity

The following functions are available:

[ProjectActivity.SetInsertMode](#)

ProjectActivity.SetInsertMode

DLL:	tpextpssapi
	This function is available from KB2271119 .
Syntax:	<pre> long ProjectActivity.SetInsertMode(domain tcmcs.str30 iInsertMode, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function sets the Insert Mode when creating new activities for a Project. The Insert Mode indicates if the new Activity is for a Top Activity, a Child Activity or a Milestone and allows proper Activity DAL handling. - To create a TOP activity, call the interface with iInsertMode set to "TopActivity". The parent activity (tppss200.pact) must remain empty in this case. - To create a child activity, call the interface with iInsertMode set to "ChildActivity". A parent activity must exist and be specified as such (in tppss200.pact) while filling the tppss200 record in this case. - To create a Milestone, call the interface with iInsertMode set to "Milestone". A parent activity must exist and be specified as such (in tppss200.pact) while filling the tppss200 record in this case. To make sure that the selected insert mode is taken into account properly, this public interface must be called before calling dal.new.object() or dal.copy.object() for tppss200. Note that there is no need to reset this insert mode after the Activity has been created. The DAL of the Activities table handles this automatically as it saves the data. This also means that the insert mode must be set before the creation of each activity separately. Example: <pre> if isspace(ParentActivity) then ReturnValue = ProjectActivity.SetInsertMode("TopActivity", ExceptionMessage, ExceptionID) else ReturnValue = ProjectActivity.SetInsertMode("ChildActivity", ExceptionMessage, ExceptionID) endif if ReturnValue <> 0 then return(ReturnValue) endif ReturnValue = dal.new.object("tppss200") if ReturnValue <> 0 then return(ReturnValue) endif dal.set.field("tppss200.cprj", MyProject) dal.set.field("tppss200.cpla", MyPlan) dal.set.field("tppss200.cact", NewActivity) ... if not isspace(ParentActivity) then dal.set.field("tppss200.pact", ParentActivity) endif </pre>

	<pre>ReturnValue = dal.save.object("tppss200") if ReturnValue <> 0 then return(ReturnValue) endif</pre>
Pre: None	
Post: None	
Input: iInsertMode	Indicates what kind of Activity is being created, mandatory. Possible values are: - TopActivity - ChildActivity - Milestone
Output: oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
Return: 0	- Successful
<> 0	- An error occurred

Chapter 33 Public Interfaces for Authorization and Security

Public Interfaces for DataAuthorization

The following functions are available:

[DataAuthorization.DocumentActionAllowed](#)

[DataAuthorization.DocumentFilter](#)

DataAuthorization.DocumentActionAllowed

DLL:	tcextsecapi										
Syntax:	<pre> long DataAuthorization.DocumentActionAllowed(domain tcsec.auth.doc iDocumentType, domain tcsec.docm iDocument, domain tcncmp iDocumentCompany, const string iDocumentAction(), boolean iSkipActionNotAllowedMessage, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>										
Usage:	Expl: This function checks if the user has authorization to perform a given action for a specific document. The authorization check is based on applied rules that have been set up in the Authorization and Security module in Common. Supported Document actions are: <table> <thead> <tr> <th>iDocumentAction</th><th>Description</th></tr> </thead> <tbody> <tr> <td>"VIEW"</td><td>Check if view is allowed</td></tr> <tr> <td>"USE"</td><td>Check if use is allowed</td></tr> <tr> <td>"MODIFY"</td><td>Check if modify is allowed</td></tr> </tbody> </table> Supported Document Types: <table> <thead> <tr> <th>iDocumentType</th><th>Description</th></tr> </thead> <tbody> </tbody> </table>	iDocumentAction	Description	"VIEW"	Check if view is allowed	"USE"	Check if use is allowed	"MODIFY"	Check if modify is allowed	iDocumentType	Description
iDocumentAction	Description										
"VIEW"	Check if view is allowed										
"USE"	Check if use is allowed										
"MODIFY"	Check if modify is allowed										
iDocumentType	Description										

Contract		
tcsec.auth.doc.contract		Contract
Project		
tcsec.auth.doc.project		Project
Procurement Documents		
tcsec.auth.doc.requisition		Requisition
tcsec.auth.doc.rfq		Request for Quotation
tcsec.auth.doc.purch.order		Purchase Order
tcsec.auth.doc.purch.contract		Purchase Contract
tcsec.auth.doc.pur.price.book		Purchase Price Book
tcsec.auth.doc.trgt.price.book		Target Price Book
Sales Documents		
tcsec.auth.doc.opportunity		Opportunity
tcsec.auth.doc.sls.quote		Sales Quote
tcsec.auth.doc.customer.order		Received Customer Order
tcsec.auth.doc.sls.order		Sales Order
tcsec.auth.doc.sls.contract		Sales Contract
tcsec.auth.doc.sls.price.book		Price Book
tcsec.auth.doc.sls.catalog		Sales Catalog
Production		
tcsec.auth.doc.work.center		Work Center
tcsec.auth.doc.work.cell		Work Cell
tcsec.auth.doc.pcs.project		Project (PCS)
Service Documents		
tcsec.auth.doc.service.contrct		Service Contract
tcsec.auth.doc.contract.quote		Contract Quote
tcsec.auth.doc.srv.order.quote		Service Order Quote
tcsec.auth.doc.mnt.sls.quote		Service Quote
tcsec.auth.doc.quote.request		Quote Request
tcsec.auth.doc.service.call		Service Call
tcsec.auth.doc.service.order		Service Order
tcsec.auth.doc.mnt.sls.order		Maintenance Sales Order
tcsec.auth.doc.work.order		Work Order
tcsec.auth.doc.customer.claim		Customer Claim
tcsec.auth.doc.supplier.claim		Supplier Claim
tcsec.auth.doc.installtn.group		Installation Group
Warehouses		
tcsec.auth.doc.warehouse		Warehouse
Items		
tcsec.auth.doc.item		Item
Business Partners		
tcsec.auth.doc.bus.partner		Business Partner
Invoice Documents		
tcsec.auth.doc.man.sls.invoice		Manual Sales Invoice
Financial Documents		
tcsec.auth.doc.transactn.type		Transaction Type
tcsec.auth.doc.ledger.account		Ledger Account
tcsec.auth.doc.gl.code		General Ledger code
tcsec.auth.doc.bank.relation		Bank Relation
tcsec.auth.doc.budget		Budget (FBS)
tcsec.auth.doc.fin.statement		Financial Statement

Input argument iDocumentCompany

Argument iDocumentCompany can be used to specify a different company than the current company.

The support for this input argument depends on both Document Type and Document action, and is limited to below list.

A "y" means: supported for the combination of Document Type and Action.

iDocumentType	iDocumentAction		
	VIEW	USE	MODIFY

Project			
tcsec.auth.doc.project	V	V	
Production			
tcsec.auth.doc.work.center	V	V	
tcsec.auth.doc.work.cell	V	V	
tcsec.auth.doc.pcs.project	V	V	
Service Documents			
tcsec.auth.doc.service.contrct	V	V	
tcsec.auth.doc.contract.quote	V	V	
tcsec.auth.doc.srv.order.quote	V	V	
tcsec.auth.doc.mnt.sls.quote	V	V	
tcsec.auth.doc.quote.request	V	V	
tcsec.auth.doc.service.call	V	V	
tcsec.auth.doc.service.order	V	V	
tcsec.auth.doc.mnt.sls.order	V	V	
tcsec.auth.doc.work.order	V	V	
tcsec.auth.doc.customer.claim	V	V	
tcsec.auth.doc.supplier.claim	V	V	
tcsec.auth.doc.installtn.group	V	V	
Warehouses			
tcsec.auth.doc.warehouse	V		
Items			
tcsec.auth.doc.item	V		
Invoice Documents			
tcsec.auth.doc.man.sls.invoice	V	V	V
Financial Documents			
tcsec.auth.doc.transactn.type	V	V	
tcsec.auth.doc.ledger.account	V	V	
tcsec.auth.doc.gl.code	V	V	
tcsec.auth.doc.bank.relation	V	V	
tcsec.auth.doc.budget	V	V	
tcsec.auth.doc.fin.statement	V	V	

Example:

To check if updating a PCS Project is allowed for the current company.

domain tcmcs.s999m exception.message

	<pre> long exception.id #pragma used dll "otcextsecapi" if DataAuthorization.DocumentActionAllowed(/* Document Type: PCS Project tcsec.auth.doc.pcs.project, /* PCS Project to check tipcs020.cprj, /* Current company get.compnr(), /* Document action: Check Modify is /* allowed "MODIFY", /* Skip the authorization error message true, /* Return arguments with error /* information if applicable exception.message, exception.id) = 0 then /* Update record or allow an action command else /* Handle the returned error(s) endif </pre>
Pre:	None
Post:	None
Input:	iDocumentType - Document Type: Mandatory - Document code: Optional Typically this argument is filled with a Document number, e.g. a Contract or Project. If the Document is empty the action is allowed (the function will return a value 0). iDocumentCompany - Document Company: Mandatory The support for this input field depends on the Document Type and Document Action (see table above). When not supported, the field value must be set to the current company (use function get.compnr()), else the current or another company can be specified. Company 0 is not allowed. iDocumentAction - The action that must be checked: Mandatory Possible actions: "VIEW", "USE and "MODIFY". iSkipActionNotAllowedMessage - Do not show an error message when the Document Action is not allowed: mandatory. The purpose of this input argument is to be able to suppress the error message that is set when the user is

	not allowed to perform the action. This may be useful when printing or processing Documents and the Documents for which the user has no authorization can just be skipped. If the value is true the error message not set. When false an error message could be set and returned in output argument oExceptionMessage. Output: oExceptionMessage - The last message if the return value of the function is not equal to 0. When input argument iSkipActionNotAllowedMessage is set to true, the returned message in this argument is the authorization error message. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - The specified Document action is allowed - An Error occurred or the user does not have authorization for the action specified.

DataAuthorization.DocumentFilter

DLL:	tcextsecapi				
Syntax:	<pre>long DataAuthorization.DocumentFilter(domain tcsec.auth.doc iDocumentType, const string iMainTableDocumentField(), const string iMainTableCompanyField(), ref string oDocumentFilter(), ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>				
Usage:	Expl: This function creates a filter that can be used in sessions to filter out records for which the user is not authorized according to applied rules set up in the Authorization and Security module in Common. By applying the filter, only records for which the user has authorization are visible. Being authorized means that the authorization level must be View, Use or Modify for the document defined by the main table of the session, e.g. a PCS Project, Sales Order or a Contract. The filter is created for the given Document Type (iDocumentType) and main table document field (iMainTableDocumentField), and must be added to the standard SQL query of the session using the query.extend.where function. Document Types supported: <table> <thead> <tr> <th>iDocumentType</th><th>Description</th></tr> </thead> <tbody> <tr> <td>-----</td><td>-----</td></tr> </tbody> </table>	iDocumentType	Description	-----	-----
iDocumentType	Description				
-----	-----				

Contract	
tcsec.auth.doc.contract	Contract
Project	
tcsec.auth.doc.project	Project
Procurement Documents	
tcsec.auth.doc.requisition	Requisition
tcsec.auth.doc.rfq	Request for Quotation
tcsec.auth.doc.purch.order	Purchase Order
tcsec.auth.doc.purch.contract	Purchase Contract
tcsec.auth.doc.pur.price.book	Purchase Price Book
tcsec.auth.doc.trgt.price.book	Target Price Book
Sales Documents	
tcsec.auth.doc.opportunity	Opportunity
tcsec.auth.doc.sls.quote	Sales Quote
tcsec.auth.doc.customer.order	Received Customer Order
tcsec.auth.doc.sls.order	Sales Order
tcsec.auth.doc.sls.contract	Sales Contract
tcsec.auth.doc.sls.price.book	Price Book
tcsec.auth.doc.sls.catalog	Sales Catalog
Production	
tcsec.auth.doc.work.center	Work Center
tcsec.auth.doc.work.cell	Work Cell
tcsec.auth.doc.pcs.project	Project (PCS)
Service Documents	
tcsec.auth.doc.service.contrct	Service Contract
tcsec.auth.doc.contract.quote	Contract Quote
tcsec.auth.doc.srv.order.quote	Service Order Quote
tcsec.auth.doc.mnt.sls.quote	Service Quote
tcsec.auth.doc.quote.request	Quote Request
tcsec.auth.doc.service.call	Service Call
tcsec.auth.doc.service.order	Service Order
tcsec.auth.doc.mnt.sls.order	Maintenance Sales Order
tcsec.auth.doc.work.order	Work Order
tcsec.auth.doc.customer.claim	Customer Claim
tcsec.auth.doc.supplier.claim	Supplier Claim
tcsec.auth.doc.installtn.group	Installation Group
Warehouses	
tcsec.auth.doc.warehouse	Warehouse
Items	
tcsec.auth.doc.item	Item
Business Partners	
tcsec.auth.doc.bus.partner	Business Partner
Invoice Documents	
tcsec.auth.doc.man.sls.invoice	Manual Sales Invoice
Financial Documents	
tcsec.auth.doc.transactn.type	Transaction Type
tcsec.auth.doc.ledger.account	Ledger Account
tcsec.auth.doc.gl.code	General Ledger code
tcsec.auth.doc.bank.relation	Bank Relation
tcsec.auth.doc.budget	Budget (FBS)
tcsec.auth.doc.fin.statement	Financial Statement

Input argument iMainTableCompanyField

Argument iMainTableCompanyField can be used to specify a company field of the main table in the filter. This company field holds the actual company where the Document exists. This can be another (logistic) company or the current company. If iMainTableCompanyField is not specified, only Documents of the current company are considered.

Input argument iMainTableCompanyField is optional, with one exception: it is mandatory for for Document Type "Manual Sales Invoice".

Also, the argument is only supported for a subset of Document Types.

Supported Document Types for argument iMainTableCompanyField:

iDocumentType	Description

Project	
tcsec.auth.doc.project	Project
Production	
tcsec.auth.doc.work.center	Work Center
tcsec.auth.doc.work.cell	Work Cell
tcsec.auth.doc.pcs.project	Project (PCS)
Service Documents	
tcsec.auth.doc.service.contrct	Service Contract
tcsec.auth.doc.contract.quote	Contract Quote
tcsec.auth.doc.srv.order.quote	Service Order Quote
tcsec.auth.doc.mnt.sls.quote	Service Quote
tcsec.auth.doc.quote.request	Quote Request
tcsec.auth.doc.service.call	Service Call
tcsec.auth.doc.service.order	Service Order
tcsec.auth.doc.mnt.sls.order	Maintenance Sales Order
tcsec.auth.doc.work.order	Work Order
tcsec.auth.doc.customer.claim	Customer Claim
tcsec.auth.doc.supplier.claim	Supplier Claim
tcsec.auth.doc.installtn.group	Installation Group
Warehouses	
tcsec.auth.doc.warehouse	Warehouse
Items	
tcsec.auth.doc.item	Item
Invoice Documents	
tcsec.auth.doc.man.sls.invoice	Manual Sales Invoice Note: mandatory
Financial Documents	
tcsec.auth.doc.transactn.type	Transaction Type
tcsec.auth.doc.ledger.account	Ledger Account
tcsec.auth.doc.gl.code	General Ledger code
tcsec.auth.doc.bank.relation	Bank Relation
tcsec.auth.doc.budget	Budget (FBS)
tcsec.auth.doc.fin.statement	Financial Statement

Usage examples:

Example without company field:

```

        string          document.filter(4096)
domain tcmcs.s999m    exception.message
        long           exception.id

#pragmama used dll      "otcextsecapi"

if DataAuthorization.DocumentFilter(
    /* Document Type: Project
    tcsec.auth.doc.project,

    /* Main table field name holding the
    /* Project
    "tppdm600.cprj",

    /* No main table field for company
    /* specified
    "",

    /* Return argument with the filter
    document.filter,

    /* Return arguments with error
    /* information if applicable
    exception.message,
    exception.id) = 0 then

    /* No errors: apply the filter
    query.extend.where(
        document.filter,
        EXTEND_APPEND)
else
    /* Handle the returned error(s)
endif

```

Example with company field:

```

if DataAuthorization.DocumentFilter(
    tcsec.auth.doc.warehouse,
    "whwmd241.cwar",
    "whwmd241.ncmp",
    document.filter,
    exception.message,
    exception.id) = 0 then

    /* No errors: apply the filter
    query.extend.where(
        document.filter,
        EXTEND_APPEND)
else
    /* Handle the returned error(s)
endif

```

Pre: None

Post: None

Input: iDocumentType - Document Type: Mandatory
iMainTableDocumentField - The name of the main table field
containing the Document: Mandatory
Note: this argument must be filled

	<code>iMainTableCompanyField</code> - with a quoted string of the table field name: e.g. "tipcs020.cprj". - The name of the main table field containing a Company: Optional, with one exception: Mandatory for Document Type "Manual Sales Invoice". If set, the Company in the field is used instead of the current company. Note: the input of a company field is not supported by all Document Types. If supported this argument must be filled with a quoted string of the table field name: e.g. "whwmd241.ncmp" else with an empty string: "". Output: <code>oDocumentFilter</code> - A string containing a SQL filter to filter records based on authorization settings for the user. <code>oExceptionMessage</code> - The last message if the return value is not equal to 0. <code>oExceptionID</code> - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Successful. <> 0 - An error occurred.

Chapter 34 Public Interfaces for Quality Management

Public Interfaces for OrderInspection

The following functions are available:

[OrderInspection.Close](#)

[OrderInspection.Complete](#)

[OrderInspection.Process](#)

OrderInspection.Close

DLL:	qmextptcapi								
Syntax:	<pre>long OrderInspection.Close(domain qmptc.orgn iOrderOrigin, domain tcorno iOrderNumber, domain qmptc.pono iOrderPosition, domain tcopno iOperation, domain qmrpt.wstt iWorkStation, domain tcpono iOrderSequence, domain tcorno iInspection, domain tcpono iInspectionLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>								
Usage:	Expl: This function closes the given order inspection and can be called for all origins except Storage Inspections, Service, Maintenance Sales, Batch Repair, Maintenance Work and Not Applicable. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input:<table><tr><td>iOrderOrigin</td><td>- Order Origin: Mandatory</td></tr><tr><td>iOrderNumber</td><td>- Order Number: Mandatory for all origins other than Inventory Inspection</td></tr><tr><td>iOrderPosition</td><td>- Order Position: Mandatory for all Origins</td></tr></table>			iOrderOrigin	- Order Origin: Mandatory	iOrderNumber	- Order Number: Mandatory for all origins other than Inventory Inspection	iOrderPosition	- Order Position: Mandatory for all Origins
iOrderOrigin	- Order Origin: Mandatory								
iOrderNumber	- Order Number: Mandatory for all origins other than Inventory Inspection								
iOrderPosition	- Order Position: Mandatory for all Origins								

	other than Routing, Production, Purchase Schedule, Production Repetitive & Inventory Inspection iOperation - Operation: Mandatory for Origin Routing iWorkStation - Work Station: Mandatory for Origin Repetitive Routing iOrderSequence - Order Sequence: Mandatory for all Origins other than Purchase Schedule & Inventory Inspection iInspection - Inspection: Mandatory for all Origins other than Routing & Repetitive Routing iInspectionLine - Inspection Line: Mandatory for Origins other than Routing & Repetitive Routing Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Order has been closed successfully <> 0 - Error

OrderInspection.Complete

DLL:	qmextptcapi
Syntax:	<pre>long OrderInspection.Complete(domain qmptc.orgn iOrderOrigin, domain tcorno iOrderNumber, domain qmptc.pono iOrderPosition, domain tcopno iOperation, domain qmrpt.wstt iWorkStation, domain tcpono iOrderSequence, domain tcorno iInspection, domain tcpono iInspectionLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function completes the given order inspection and this can be called for all origins except Storage Inspections, Service, Maintenance Sales, Batch Repair, Maintenance Work and Not Applicable. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iOrderOrigin - Order Origin: Mandatory iOrderNumber - Order Number: Mandatory for all origins other than Inventory Inspection iOrderPosition - Order Position: Mandatory for all Origins other than Routing, Production, Purchase Schedule, Production Repetitive & Inventory Inspection iOperation - Operation: Mandatory for Origin Routing iWorkStation - Work Station: Mandatory for Origin

	Inspection iOrderSequence iInspection iInspectionLine Output: oExceptionMessage oExceptionID Return: 0 <> 0	Repetitive Routing - Order Sequence: Mandatory for all Origins other than Purchase Schedule & Inventory - Inspection: Mandatory for all Origins other than Routing & Repetitive Routing - Inspection Line: Mandatory for Origins other than Routing & Repetitive Routing - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Order has been completed successfully - Error

OrderInspection.Process

DLL:	qmextptcapi		
Syntax:	<pre>long OrderInspection.Process(domain qmptc.orgn iOrderOrigin, domain tcorno iOrderNumber, domain qmptc.pono iOrderPosition, domain tcopno iOperation, domain qmrpt.wstt iWorkStation, domain tcpono iOrderSequence, domain tcorno iInspection, domain tcpono iInspectionLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>		
Usage:	Expl: This function processes the given order inspection and this can be called for all origins except Storage Inspections, Service, Maintenance Sales, Batch Repair, Maintenance Work and Not Applicable. And this function must not be called within a logical transaction as this function will have its own transaction handling. Pre: NA Post: NA Input: iOrderOrigin - Order Origin: Mandatory iOrderNumber - Order Number: Mandatory for all origins other than Inventory Inspection iOrderPosition - Order Position: Mandatory for all Origins other than Routing, Production, Purchase Schedule, iOperation - Operation: Mandatory for Origin Routing iWorkStation - Work Station: Mandatory for Origin Repetitive Routing iOrderSequence - Order Sequence: Mandatory for all Origins		

	Inspection iInspection iInspectionLine Output: oExceptionMessage oExceptionID Return: 0 <> 0	other than Purchase Schedule & Inventory - Inspection: Mandatory for all Origins other than Routing & Repetitive Routing - Inspection Line: Mandatory for Origins other than Routing & Repetitive Routing - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Order has been processed succesfully - Error

Public Interfaces for InspectionOrder

The following functions are available:

[InspectionOrder.Print](#)

InspectionOrder.Print

DLL:	qmextptcapi	
Syntax:	<pre> long InspectionOrder.Print(domain qmptc.orgn iOrderOrigin, domain tcorno iOrderNumber, domain qmptc.pono iOrderPosition, domain tcopno iOperation, domain qmrpt.wstt iWorkStation, domain tcpono iOrderSequence, domain tcorno iInspection, domain tcpono iInspectionLine, domain qmptc.iorn iInspectionOrder, domain tcitem iItem, domain tcyesno iPrintPrintedLines, domain tcyesno iPrintStorageInspInv, domain tcmcs.str15 iPrintDevice, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>	
Usage:	Expl:	This function prints the given inspection order and this can be called for all origins except Service, Maintenance Sales, Batch Repair, Maintenance Work and Not Applicable. And this

	function must not be called within a logical transaction as this function will have its own transaction handling.																
Pre:	NA																
Post:	NA																
Input:																	
	iOrderOrigin - Order Origin: Mandatory when Inspection Order is not given iOrderNumber - Order Number: Mandatory for all origins other than Inventory Inspection and when Inspection Order is not given iOrderPosition - Order Position: Mandatory for all Origins other than Routing, Production, Production Repetitive & Inventory Inspection and when Inspection Order is not given iOperation - Operation: Mandatory for Origin Routing when Inspection Order is not given iWorkStation - Work Station: Mandatory for Origin Repetitive Routing and when Inspection Order is not given iOrderSequence - Order Sequence: Mandatory for all Origins other than Inventory Inspection and when Inspection Order is not given iInspection - Inspection: Mandatory for all Origins other than Routing & Repetitive Routing and when Inspection Order is not given iInspectionLine - Inspection Line: Mandatory for Origins other than Routing & Repetitive Routing and when Inspection Order is not given iInspectionOrder - QM Inspection Order: Mandatory when Order details are not given iItem - Item: Mandatory when Origin is Storage Inspection and Inspection Order is not given iPrintPrintedLines - Print Already Printed Lines Indicator: Mandatory iPrintStorageInspInv - Print Storage Inspection Inventory Lines: Mandatory iPrintDevice - Printing Device: Not Mandatory If not Provided asked for device when report is opened. <tr> <td>Output:</td><td>oExceptionMessage</td></tr> <tr> <td></td><td>- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID</td></tr> <tr> <td></td><td>oExceptionID</td></tr> <tr> <td></td><td>- An ID that refers to all error information. Use the functions in Exception to get all relevant information.</td></tr> <tr> <td>Return:</td><td>0</td></tr> <tr> <td></td><td><> 0</td></tr> <tr> <td></td><td>- Order has been printed successfully</td></tr> <tr> <td></td><td>- Error</td></tr>	Output:	oExceptionMessage		- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID		oExceptionID		- An ID that refers to all error information. Use the functions in Exception to get all relevant information.	Return:	0		<> 0		- Order has been printed successfully		- Error
Output:	oExceptionMessage																
	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID																
	oExceptionID																
	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.																
Return:	0																
	<> 0																
	- Order has been printed successfully																
	- Error																

Public Interfaces for CorrectiveActionPlan

The following functions are available:

[CorrectiveActionPlan.SetStatus](#)

[CorrectiveActionPlan.StartMultiMain](#)

CorrectiveActionPlan.SetStatus

DLL:	qmextcplapi
Syntax:	<pre>long CorrectiveActionPlan.SetStatus(domain tcorno iCorrectiveActionPlan, domain qmcpl.caps iCorrectiveActionPlanStatus, ref domain tcmcs.s99m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This public interface will set Corrective Action Plan status. Pre: Not Applicable Post: Not Applicable Input: iCorrectiveActionPlan - Corrective Action Plan; Mandatory iCorrectiveActionPlanStatus - Corrective Action Plan Status; Mandatory qmcpl.caps.open - Open (Reset) qmcpl.caps.submitted - Submitted qmcpl.caps.approved - Approved qmcpl.caps.complete - Completed qmcpl.caps.cancelled - Canceled qmcpl.caps.closed - Closed Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Corrective Action Plan Status changed to given status. <> 0 - Corrective Action Plan Status could not be changed.

CorrectiveActionPlan.StartMultiMain

DLL:	qmextcplapi
Syntax:	<pre>long CorrectiveActionPlan.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(),</pre>

	<pre> domain tccono iCorrectiveActionPlan, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the Multi-Main session "Corrective Action Plan" (qmcpl1100m900). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend Not Used iCorrectiveActionPlan Corrective Action Plan, Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - An error occurred

Public Interfaces for CorrectiveActionPlanTask

The following functions are available:

[CorrectiveActionPlanTask.SetStatus](#)

CorrectiveActionPlanTask.SetStatus

DLL:	qmextcplapi
Syntax:	<pre> long CorrectiveActionPlanTask.SetStatus(domain tcorno iCorrectiveActionPlan, </pre>

	domain tcpono iCorrectiveActionPlanTask, domain qmcpl.tsks iCorrectiveActionPlanTaskStatus, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This public interface will set Corrective Action Plan Task status. Pre: Not Applicable Post: Not Applicable Input: iCorrectiveActionPlan - Corrective Action Plan; Mandatory iCorrectiveActionPlanTask - Corrective Action Plan Task; Mandatory iCorrectiveActionPlanTaskStatus - Corrective Action Plan Task Status ; Mandatory qmcpl.tsks.inprogress - In Progress qmcpl.tsks.completed - Completed qmcpl.tsks.closed - Closed qmcpl.tsks.cancelled - Canceled Output: oExceptionMessage - The last message if the return value is not equal to 0. - If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Corrective Action Plan Task Status changed to given status. <> 0 - Corrective Action Plan Task Status could not be changed.

Public Interfaces for FRACASHeader

The following functions are available:

[FRACASHeader.StartMultiMain](#)

FRACASHeader.StartMultiMain

DLL:	qmextpqmapi
Syntax:	long FRACASHeader.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iFRACASNumber, ref domain tcmcs.s999m oExceptionMessage mb,

	ref	long	oExceptionID)
Usage:	Expl:	This function starts the Multi-Main session "Failure Reporting Analysis and Corrective Action System Header" (qmpqm2600m100).	
	Input:	iStartMode	Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously.
		iStartFilter	Not Used
		iSessionIndex	Not Used
		iQueryExtend	Not Used
		iFRACASNumber	FRACAS Number, Mandatory
	Output:	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID
		oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
	Return:	0	- Session started
		<> 0	- An error occurred

Public Interfaces for TestingCombination

The following functions are available:

[TestingCombinations.StartOverview](#)

TestingCombinations.StartOverview

DLL:	qmextptcapi
Syntax:	long TestingCombinations.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter,

	<pre> const long iSessionIndex, string iQueryExtend(), domain qmptc.orgn iOrderOrigin, domain tcmcs.qitg iQualityGroup, domain tcitem iItem, domain tcuef.effn iItemEffectivityUnit, domain tcedm.revi iItemRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the overview session Testing Combinations (qmptc0119m000). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Specifies the session index that is to be used. iQueryExtend A specific query to be used when zooming to this session. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the oExceptionMessage and the session will not be started. Primary Key Fields: - iOrderOrigin - Order Origin; Mandatory - iQualityGroup - Quality Group - iItem - Item - iItemEffectivityUnit - Item Effectivity Unit - iItemRevision - Item Revision Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - An error occurred

Public Interfaces for StandardTestProcedure

The following functions are available:

[StandardTestProcedure.StartMultiMain](#)

StandardTestProcedure.StartMultiMain

DLL:	qmextptcapi
Syntax:	<pre> long StandardTestProcedure.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain qmptc.quid iStandardTestProcedure, domain qmptc.srno iStandardTestProcedureVersion, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts the Multi-Main session "Standard Test Procedure" (qmptc0110m100). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits. The session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Not Used. iSessionIndex Not Used. iQueryExtend Not Used iStandardTestProcedure Standard Test Procedure, Mandatory iStandardTestProcedureVersion Standard Test Procedure Version, Mandatory Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Session started <> 0 - An error occurred

Public Interfaces for NonConformanceReport

The following functions are available:

[NonConformanceReport.StartDetail](#)

NonConformanceReport.StartDetail

DLL:	qmextncmapi
Syntax:	<pre> long NonConformanceReport.StartDetail(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcorno iNonConformanceReport, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session Non-Conformance Report (qmncml100m000). Input: <i>iStartMode</i> Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, in case of a multi-occurrence the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. <i>iStartFilter</i> Not Used. <i>iSessionIndex</i> Not Used. <i>iQueryExtend</i> Not Used. Following input variables form the primary key, these fields are mandatory, if the primary key cannot be found an API error will be set in the <i>oExceptionMessage</i> and the session will not be started. Primary Key Fields: <i>iNonConformanceReport</i> - Non-Conformance Report; Mandatory Output: <i>oExceptionMessage</i> - The last message if the return value is not equal to 0.

	oExceptionID Return: 0 <> 0	If more than one message is given, these are present in the oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Session started - An error occurred

Chapter 35 Public Interfaces for Cash Management

Public Interfaces for PaymentAdvice

The following functions are available:

[PaymentAdvice.StartProcessPayments](#)

PaymentAdvice.StartProcessPayments

DLL:	tfextcmgapi
Syntax:	long PaymentAdvice.StartProcessPayments(long iStartMode, domain tfgld.user iUserFrom, domain tfgld.user iUserTo, domain tfgld.btno iPaymentBatchFrom, domain tfgld.btno iPaymentBatchTo, domain tfcmg.stpp iPaymentBatchStatusFrom, domain tfcmg.stpp iPaymentBatchStatusTo, domain tfgld.date iBatchCreationDateFrom, domain tfgld.date iBatchCreationDateTo, domain tfcmg.paym iPaymentProcessDefaultID, domain tfcmg.bank iDefaultBankRelation, domain tfgld.desc iTextForRemittance mb, domain tcyesno iIncludeRemittanceIDInReference, domain tcmcs.str3 iSuffix, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This function starts session Process Payments (tfcmg1240m000) Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. iUserFrom - User From Not Mandatory iUserTo - User To Not Mandatory iPaymentBatchFrom - Payment Batch From Not Mandatory iPaymentBatchTo - Payment Batch To Not Mandatory iPaymentBatchStatusFrom - Payment Batch Status From Not Mandatory iPaymentBatchStatusTo - Payment Batch Status To Not Mandatory

	Mandatory	iBatchCreationDateFrom	- Batch Creation Date From	Not Mandatory
		iBatchCreationDateTo	- Batch Creation Date To	Not Mandatory
		iPaymentProcessDefaultID	-Payment Process Default ID	Not
	Not Mandatory	iDefaultBankRelation	- Default Bank Relation	Not Mandatory
		iTextForRemittance	- Text For Remittance	Not Mandatory
		iIncludeRemittanceIDInReference	- Include Remittance ID in Reference	
	Output: for iStartMode MODAL:	iSuffix	-Suffix	Not Mandatory
		oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID	
		oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.	
	Return: 0 <> 0		- Session started	
			- Error	

Chapter 36 Public Interfaces for Accounts Receivable

Public Interfaces for AccountsReceivable

The following functions are available:

[AccountsReceivable.DetermineDueDate](#)

[AccountsReceivable.StartOverview](#)

AccountsReceivable.DetermineDueDate

DLL:	tfextacapi		
	This function is available from KB2300364 .		
Syntax:	long AccountsReceivable.DetermineDueDate(domain tcncmp const domain tccom.bpid const domain tccpay domain tfgld.amnt domain tfgld.date ref domain tfgld.date ref domain tcmcs.s999m ref long iFinancialCompany, iPayByBusinessPartner, iPaymentTerms, iInvoiceAmount, iDocumentDate, oDueDate, oExceptionMessage mb, oExceptionID)		
Usage:	Expl: This function determines a due date based on the given input parameters. Here the document date and due date are non-UTC dates. Pre: NA Post: NA Input: iFinancialCompany iPayByBusinessPartner iPaymentTerms iInvoiceAmount iDocumentDate - Financial Company - Pay-by Business Partner - Payment Terms - Invoice Amount - (Not used yet) - Document Date eg. Invoice date Output: oDueDate oExceptionMessage - Determined Due Date - The last message if any message is found. If more than one message is		

	given, these are present in the oExceptionID. - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 <> 0 - OK - on errors

AccountsReceivable.StartOverview

DLL:	tfextacapi
	This function is available from KB3524683 .
Syntax:	<pre> long AccountsReceivable.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tccom.bpid iInvoiceToBusinessPartner, domain tccwoc iDepartment, domain tcseak iSearchKey mb, domain tccom.cadr iAddress, domain tcemno iCreditAnalyst, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts Overview session Accounts Receivable 360 - (tfacr2560m000). Pre: na Post: na Input: iStartMode - Start Mode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are:"" iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a iStartFilter the session index will match the value of this variable. iQueryExtend A specific query to be used when zooming to this session. iInvoiceToBusinessPartner - Invoice-To Business Partner iDepartment - Department iSearchKey - Search Key iAddress - Address

Public Interfaces for Accounts Receivable

	iCreditAnalyst Output: oExceptionMessage oExceptionID Return: 0 <> 0	- Credit Analyst - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. - succes otherwise

Chapter 37 Public Interfaces for General Ledger

Public Interfaces for GeneralLedger

The following functions are available:

[GeneralLedger.CheckBatchLineIsBalanced](#)

[GeneralLedger.CheckDocumentIsBalanced](#)

GeneralLedger.CheckBatchLineIsBalanced

DLL:	tfextgldapi																		
	This function is available from KB2301011 .																		
Syntax:	<pre>long GeneralLedger.CheckBatchLineIsBalanced(domain tcncmp iFinancialCompany, domain tfgld.year iFiscalYear, domain tfgld.btno iBatch, domain tfgld.ttyp iTransactionType, ref boolean oNonFinalizedTransactionPresent, ref boolean oBatchLineIsBalanced, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>																		
Usage:	Expl: This function checks whether all documents of the given batch and transaction type are balanced. A batch line becomes balanced when all documents within the batch line are balanced. If one of the documents within the batch line is not balanced then oBatchLineIsBalanced will be false. In case of Batch is not found or has been already finalized, then an error message is given. So, only Non-Finalized batches should be used. Pre: NA Post: NA <table border="0"> <tr> <td>Input:</td> <td>iFinancialCompany</td> <td>- Financial Company</td> <td>- Mandatory</td> </tr> <tr> <td></td> <td>iFiscalYear</td> <td>- Fiscal Year</td> <td>- Mandatory</td> </tr> <tr> <td></td> <td>iBatch</td> <td>- Batch</td> <td>- Mandatory</td> </tr> <tr> <td></td> <td>iTransactionType</td> <td>- Transaction type</td> <td>- Mandatory</td> </tr> </table> Output: oNonFinalizedTransactionPresent -			Input:	iFinancialCompany	- Financial Company	- Mandatory		iFiscalYear	- Fiscal Year	- Mandatory		iBatch	- Batch	- Mandatory		iTransactionType	- Transaction type	- Mandatory
Input:	iFinancialCompany	- Financial Company	- Mandatory																
	iFiscalYear	- Fiscal Year	- Mandatory																
	iBatch	- Batch	- Mandatory																
	iTransactionType	- Transaction type	- Mandatory																

	Non-finalized transaction present (True or False) - Batch Line is balanced indicator (True or False) oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - OK <> 0 - on errors

GeneralLedger.CheckDocumentIsBalanced

DLL:	tfextgldapi This function is available from KB2285516.
Syntax:	<pre>long GeneralLedger.CheckDocumentIsBalanced(domain tcncmp iFinancialCompany, domain tfgld.ttyp iTransactionType, domain tfgld.docn iDocument, ref boolean oDocumentIsBalanced, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function checks if the given input document is balanced. Pre: NA Post: NA Input: iFinancialCompany - Financial Company iTransactionType - Transaction Type iDocument - Document Output: oDocumentIsBalanced - Document is balanced indicator oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - OK <> 0 - on errors

Public Interfaces for Account

The following functions are available:

[Account.CrossValidateDimensions](#)

Account.CrossValidateDimensions

DLL:	tfextgldapi		
	This function is available from KB3525960 .		
Syntax:	<pre>long Account.CrossValidateDimensions(domain tcncmp iFinancialCompany, domain tcdate iDateForCheck, domain tfgld.leac iLedgerAccount, domain tfgld.dimn iDimensionNumber, const domain tfgld.dimx iDimensionArray() fixed, ref boolean oValidDimensions, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>		
Usage:	Expl.: This function checks whether the dimensions are valid according to the cross validation rules. Pre: - Post: - Input: iFinancialCompany - Financial Company: Mandatory iDateForCheck - Date for check in UTC format: Mandatory iLedgerAccount - Ledger Account: Mandatory iDimensionNumber - Until (and including) which dimension of the array is to be checked: Mandatory iDimensionArray - Array with dimensions Output: oValidDimensions - Dimensions are valid according to cross validation rules oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Data read <> 0 - An error occurred Return: 0/DALHOOKERROR		

Chapter 38 Public Interfaces for Fixed Assets

Public Interfaces for Asset

The following functions are available:

[Asset.GetDefaultLedgerAccountAndDimensions](#)

[Asset.StartAdjustAssets](#)

[Asset.StartTransferAssets](#)

Asset.GetDefaultLedgerAccountAndDimensions

DLL:	ttextfamapi		
	This function is available from KB2331941 .		
Syntax:	<pre> long Asset.GetDefaultLedgerAccountAndDimensions (domain tcncmp iAssetCompany, domain tffam.mcod iAssetNumber, domain tffam.mcod iAssetExtension, domain tffam.code iAssetBook, domain tccom.bpid iBusinessPartner, const domain tffam.lcod iLocationSegments() fixed, domain tcidty iIntegrationDocumentType, domain tffam.dtty iDepreciationType, domain tffam.dtyp iDisposalType, domain tffam.code iReason, domain tfgld.dbcr iDebitCredit, domain tfgld.date iTransactionDate, long iProcessingOptionSet, ref domain tfgld.leac oLedgerAccount, ref domain tfgld.dimx oDimensions() fixed, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>		
Usage:	Expl.: This function retrieves ledger account and dimensions from the mapping scheme for a fixed asset. Pre: - Post: - Input: iAssetCompany - Asset Company: Mandatory iAssetNumber - Asset Number: Mandatory		

iAssetExtension	- Asset Extension: Mandatory	
iAssetBook	- Asset Book: Mandatory	
iBusinessPartner	- Business Partner: Optional	
iLocationSegments	- Array with Location Segments: Optional	
iIntegrationDocumentType	- Integration Document Type: Mandatory	
iDepreciationType	- Depreciation Type: Optional	
iDisposalType	- Disposal Type: Optional	
iReason	- Reason: Optional	
iDebitCredit	- Debit/Credit Indicator: Mandatory	
iTransactionDate	- Transaction Date: Mandatory	
iProcessingOptionSet	- Optional, if 0, the optional values are retrieved from the asset and/or asset book. A Processing Option Set can be created via a call to ProcessingOptionSet.Create().	
NAME	TYPE	DEFAULT
OwningCompany	domain tcncomp	from Asset
OwningDepartment	domain tccwoc	from Asset
Category	domain tffam.code	from Asset
Subcategory	domain tffam.code	from Asset
Group	domain tffam.code	from Asset
Frequency	domain tffam.code	from Asset Book
PropertyType	domain tffam.code	from Asset Book
DepreciationCode	domain tffam.meth	from Asset Book
Output: oLedgerAccount	- Ledger Account	
oDimensions	- Array with Dimensions	
oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.	
oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.	
Return: 0	- Data read	
<> 0	- An error occurred	

Asset.StartAdjustAssets

DLL:	ttextfamapi This function is available from KB3603659 .
Syntax:	<pre> long Asset.StartAdjustAssets(domain tffam.mcod iAssetNumber, domain tffam.mcod iAssetExtension, long iStartMode, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session 'Adjust Assets' (tffam8220m000). Input: iAssetNumber - Asset Number: Mandatory

iAssetExtension	- Asset Extension: Mandatory
iStartMode	- Specifies the start mode for the session. Not used.
iProcessingOptionSet	- Optional, if 0, the session is started with regular defaulting logic (user defaults or session defaults), except selection ranges for Asset Number and Asset Extension, which are defaulted with iAssetNumber and iAssetExtension. A Processing Option Set can be created via a call to ProcessingOptionSet.Create().

Processing Options have a direct relationship with the form fields on session Adjust Assets (tffam8220m000) and are not explained in further detail here. Please refer to the session help for additional information.

Options which are not available as Processing Options will get defaulted with the value in column DEFAULT below.

Processing Options that are set while a required Implemented Software Component is not available are ignored.

Explanation about setting of defaults:

Minimum Value: Minimum value of domain is taken as default value.

Maximum Value: Maximum value of domain is taken as default value.

NAME	TYPE	DEFAULT
AssetNumberFrom	domain tffam.mcod	iAssetNumber
AssetNumberTo	domain tffam.mcod	iAssetNumber
AssetExtensionFrom	domain tffam.mcod	iAssetExtension
AssetExtensionTo	domain tffam.mcod	iAssetExtension
CategoryFrom	domain tffam.code	minimum value
CategoryTo	domain tffam.code	maximum value
SubcategoryFrom	domain tffam.code	minimum value
SubcategoryTo	domain tffam.code	maximum value
GroupFrom	domain tffam.code	minimum value
GroupTo	domain tffam.code	maximum value
ITCMethodFrom	domain tffam.itcc	minimum value
ITCMethodTo	domain tffam.itcc	maximum value
VintageGroupAccountFrom	domain tffam.lcod	minimum value
VintageGroupAccountTo	domain tffam.lcod	maximum value
InServiceDateFrom	domain tfgld.date	minimum value
InServiceDateTo	domain tfgld.date	current date
CurrentQuantityFrom	domain tffam.pzlg	minimum value
CurrentQuantityTo	domain tffam.pzlg	999999999
Automobile	domain tffam.aynb	tffam.aynb.both
Listed	domain tffam.aynb	tffam.aynb.both
New	domain tffam.aynb	tffam.aynb.both
Location1From	domain tffam.lcod	minimum value
Location1To	domain tffam.lcod	maximum value
Location2From	domain tffam.lcod	minimum value
Location2To	domain tffam.lcod	maximum value
Location3From	domain tffam.lcod	minimum value
Location3To	domain tffam.lcod	maximum value
Location4From	domain tffam.lcod	minimum value
Location4To	domain tffam.lcod	maximum value
Location5From	domain tffam.lcod	minimum value
Location5To	domain tffam.lcod	maximum value
Location6From	domain tffam.lcod	minimum value
Location6To	domain tffam.lcod	maximum value
Location7From	domain tffam.lcod	minimum value
Location7To	domain tffam.lcod	maximum value
Location8From	domain tffam.lcod	minimum value

Location8To	domain	tffam.lcod	maximum value
LocationChange	domain	tcyesno	tcyesno.no
EffectiveFrom	domain	tffam.date	tffam.date.eff
AdjustmentDate	domain	tfgld.date	current date
Reason	domain	tffam.code	minimum value
Suppress	domain	tcyesno	tcyesno.no
PrintReport	domain	tcyesno	tcyesno.yes
ActionCategory	domain	tffam.oper2	tffam.oper2.no.change
Category	domain	tffam.code	minimum value
Subcategory	domain	tffam.code	minimum value
ActionGroup	domain	tffam.oper2	tffam.oper2.no.change
Group	domain	tffam.code	minimum value
ActionInServiceDate	domain	tffam.oper	tffam.oper2.no.change
InServiceDate	domain	tfgld.date	minimum value
ActionOwner	domain	tffam.oper2	tffam.oper2.no.change
Owner	domain	tffam.ownc	minimum value
ActionITCMethod	domain	tffam.oper2	tffam.oper2.no.change
ITCMethod	domain	tffam.itcc	tffam.itcc.none
ActionAuto	domain	tffam.oper2	tffam.oper2.no.change
Auto	domain	tffam.ajyn	tffam.ajyn.no
ActionListed	domain	tffam.oper2	tffam.oper2.no.change
Listed	domain	tffam.ajyn	tffam.ajyn.no
ActionNew	domain	tffam.oper2	tffam.oper2.no.change
New	domain	tffam.ajyn	tffam.ajyn.no
ActionAssetDistribtionBy	domain	tffam.oper2	tffam.oper2.no.change
AssetDistributionBy	domain	tffam.byqp	tffam.byqp.quantity
ActionVintageAccount	domain	tffam.vint.act	tffam.vint.act.nothing
VintageAccount	domain	tffam.lcod	minimum value
ActionBusinessPercentage	domain	tffam.oper2	tffam.oper2.no.change
BusinessPercentage	domain	tffam.pcnt	minimum value
Output: oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.		
oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.		
Return: 0	- Session started		
<> 0	- Error		

Asset.StartTransferAssets

DLL:	ttextfamapi		
	This function is available from KB2327954 .		
Syntax:	<pre>long Asset.StartTransferAssets(domain tffam.mcod iAssetNumber, domain tffam.mcod iAssetExtension, long iStartMode, long iProcessingOptionSet, ref domain tcmcs.s999m oExceptionMessage mb,</pre>		

Public Interfaces for Fixed Assets

	ref	long	oExceptionID)
Usage:	Expl:	This function starts session 'Transfer Assets' (tffam821lm000).	
	Input:	iAssetNumber	- Asset Number: Mandatory
		iAssetExtension	- Asset Extension: Mandatory
		iStartMode	- Specifies the start mode for the session. Not used.
		iProcessingOptionSet	- Optional, if 0, the session is started with regular defaulting logic (user defaults or session defaults), except selection ranges for Asset Number and Asset Extension, which are defaulted with iAssetNumber and iAssetExtension. A Processing Option Set can be created via a call to ProcessingOptionSet.Create().
	Processing Options have a direct relationship with the form fields on session Transfer Assets (tffam821lm000) and are not explained in further detail here. Please refer to the session help for additional information. Options which are not available as Processing Options will get defaulted with the value in column DEFAULT below. Processing Options that are set while a required Implemented Software Component is not available are ignored. Explanation about setting of defaults: Minimum Value: Minimum value of domain is taken as default value. Maximum Value: Maximum value of domain is taken as default value.		
	NAME	TYPE	DEFAULT
	AssetNumberFrom	domain tffam.mcod	iAssetNumber
	AssetNumberTo	domain tffam.mcod	iAssetNumber
	AssetExtensionFrom	domain tffam.mcod	iAssetExtension
	AssetExtensionTo	domain tffam.mcod	iAssetExtension
	CategoryFrom	domain tffam.code	minimum value
	CategoryTo	domain tffam.code	maximum value
	SubcategoryFrom	domain tffam.code	minimum value
	SubcategoryTo	domain tffam.code	maximum value
	GroupFrom	domain tffam.code	minimum value
	GroupTo	domain tffam.code	maximum value
	ITCMethodFrom	domain tffam.itcc	minimum value
	ITCMethodTo	domain tffam.itcc	maximum value
	VintageGroupAccountFrom	domain tffam.lcod	minimum value
	VintageGroupAccountTo	domain tffam.lcod	maximum value
	InServiceDateFrom	domain tfgld.date	minimum value
	InServiceDateTo	domain tfgld.date	current date
	CurrentQuantityFrom	domain tffam.pzlg	minimum value
	CurrentQuantityTo	domain tffam.pzlg	999999999
	Automobile	domain tffam.aynb	tffam.aynb.both
	Listed	domain tffam.aynb	tffam.aynb.both
	New	domain tffam.aynb	tffam.aynb.both
	Location1From	domain tffam.lcod	minimum value
	Location1To	domain tffam.lcod	maximum value
	Location2From	domain tffam.lcod	minimum value
	Location2To	domain tffam.lcod	maximum value
	Location3From	domain tffam.lcod	minimum value
	Location3To	domain tffam.lcod	maximum value
	Location4From	domain tffam.lcod	minimum value
	Location4To	domain tffam.lcod	maximum value
	Location5From	domain tffam.lcod	minimum value
	Location5To	domain tffam.lcod	maximum value
	Location6From	domain tffam.lcod	minimum value
	Location6To	domain tffam.lcod	maximum value

Location7From	domain	tffam.lcod	minimum value
Location7To	domain	tffam.lcod	maximum value
Location8From	domain	tffam.lcod	minimum value
Location8To	domain	tffam.lcod	maximum value
EffectiveFrom	domain	tffam.date	tffam.date.eff
TransferDate	domain	tfgld.date	current date
Reason	domain	tffam.code	minimum value
TransferPercentage	domain	tfgld.perc	100.0
TransferAmount	domain	tfgld.amnt	0.0
TransferDistribution	domain	tcyesno	tcyesno.no
Suppress	domain	tcyesno	tcyesno.no
PrintReport	domain	tcyesno	tcyesno.yes
InServDateEqualToTransferDate	domain	tcyesno	tcyesno.no
TargetCompany	domain	tcncmp	current company
BasedOnBookValue	domain	tcyesno	tcyesno.no
TargetVintageGroupAccount	domain	tffam.lcod	minimum value
TargetBusinessPartner	domain	tccom.bpid	minimum value
CopyAsset	domain	tcyesno	tcyesno.yes
TargetAsset	domain	tffam.mcod	minimum value
CopyExtension	domain	tcyesno	tcyesno.no
TargetExtension	domain	tffam.mcod	minimum value
FirstFreeAssetExtension	domain	tcyesno	tcyesno.no
CopyCategory	domain	tcyesno	tcyesno.yes
TargetCategory	domain	tffam.code	minimum value
TargetSubcategory	domain	tffam.code	minimum value
CopyGroup	domain	tcyesno	tcyesno.yes
TargetGroup	domain	tffam.code	minimum value
Output: oExceptionMessage	- The last message if any message is found. If more than one message is given, these are present in the oExceptionID.		
oExceptionID	- An ID that refers to the exception information. Use the functions in Exception to get all relevant information.		
Return: 0	- Session started		
<> 0	- Error		

Public Interfaces for AssetBook

The following functions are available:

[AssetBook.CalculateRemainingDepreciationLife](#)

AssetBook.CalculateRemainingDepreciationLife

DLL:	tfextfamapi This function is available from KB2287701.																				
Syntax:	<pre> long AssetBook.CalculateRemainingDepreciationLife(domain tcncmp iAssetCompany, domain tffam.mcod iAssetNumber, domain tffam.mcod iAssetExtension, domain tffam.code iAssetBook, ref domain tcmcs.double oRemainingDays, ref domain tcmcs.double oRemainingPeriods, ref domain tffam.life oRemainingLifeYears, ref domain tffam.life oRemainingLifePeriods, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>																				
Usage:	Expl.: This function calculates the remaining depreciation life of an asset book. The remaining depreciation life takes the average convention of the used depreciation code into account. Next output arguments are returned: - oRemainingDays: Used for depreciation calculation when depreciation mode in the FAM parameters is set to Daily. - oRemainingPeriods: Unrounded value of remaining periods, used for depreciation calculation when depreciation mode in the FAM parameters is set to Periodically. - oRemainingLifeYears / oRemainingLifePeriods: Rounded number of entire years and periods, derived from oRemainingPeriods and used for display purposes in several sessions, because in fixed asset management it is common practice to express the life in years and periods. Pre: - Post: - Input: <table border="0"> <tr> <td>iAssetCompany</td><td>- Asset Company: Mandatory</td></tr> <tr> <td>iAssetNumber</td><td>- Asset Number: Mandatory</td></tr> <tr> <td>iAssetExtension</td><td>- Asset Extension: Mandatory</td></tr> <tr> <td>iAssetBook</td><td>- Asset Book: Mandatory</td></tr> </table> Output: <table border="0"> <tr> <td>oRemainingDays</td><td>- Remaining days to be depreciated till the end of the asset life.</td></tr> <tr> <td>oRemainingPeriods</td><td>- Remaining periods to be depreciated till the end of the asset life.</td></tr> <tr> <td>oRemainingLifeYears</td><td>- Entire number of years still to be depreciated.</td></tr> <tr> <td>oRemainingLifePeriods</td><td>- Rounded number of periods still to be depreciated.</td></tr> <tr> <td>oExceptionMessage</td><td>- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.</td></tr> <tr> <td>oExceptionID</td><td>- An ID that refers to all error information. Use the functions in Exception to get all relevant information.</td></tr> </table> Return: 0 - Data read <> 0 - An error occurred	iAssetCompany	- Asset Company: Mandatory	iAssetNumber	- Asset Number: Mandatory	iAssetExtension	- Asset Extension: Mandatory	iAssetBook	- Asset Book: Mandatory	oRemainingDays	- Remaining days to be depreciated till the end of the asset life.	oRemainingPeriods	- Remaining periods to be depreciated till the end of the asset life.	oRemainingLifeYears	- Entire number of years still to be depreciated.	oRemainingLifePeriods	- Rounded number of periods still to be depreciated.	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
iAssetCompany	- Asset Company: Mandatory																				
iAssetNumber	- Asset Number: Mandatory																				
iAssetExtension	- Asset Extension: Mandatory																				
iAssetBook	- Asset Book: Mandatory																				
oRemainingDays	- Remaining days to be depreciated till the end of the asset life.																				
oRemainingPeriods	- Remaining periods to be depreciated till the end of the asset life.																				
oRemainingLifeYears	- Entire number of years still to be depreciated.																				
oRemainingLifePeriods	- Rounded number of periods still to be depreciated.																				
oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.																				
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.																				

Chapter 39 Public Interfaces for Product Lifecycle Management

Public Interfaces for PLMObject

The following functions are available:

[PLMObject.CheckIn](#)

[PLMObject.CheckOut](#)

[PLMObject.UndoCheckIn](#)

[PLMObject.UndoCheckOut](#)

PLMObject.CheckIn

DLL:	pdextpdmapi
Syntax:	<pre> long PLMObject.CheckIn(domain pdobjt iObjectType, domain pdokey iKey, domain pdrevi iRevision, domain pdproj iProject, domain pdyesno iIgnoreProjectConfig, domain pdyesno iUnlockFilesOnCheckIn, ref domain tcguid.extend oTransactionGUID, ref domain tcmcs.s999m oErrorMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface for Revision Control Check-In. All the parameters are mandatory. Pre: The transaction must have been already committed prior to calling this function. This function internally performs transaction handling. Post: None Input: iObjectType - Object Type Possible values are: - pdobjt.item (Item) - pdobjt.document (Document) - pdobjt.folder (Folder) iKey - Object Key

	iRevision - Object Revision iProject - Project iIgnoreProjectConfig - Ignore Project Configuration Parameters when the Object is Check-In. iUnlockFilesOnCheckIn - Indicates if files shall be automatically unlocked as part of Check-In operation. Output: oTransactionGUID - Transaction GUID. The Revision Control Mechanism processing objects will be inserted into the tables pdcom013- Revision Control Objects and pdcom017- Revision Control Validations with the oTransaction as guid. The calling method gets the objects and it's validations information using this field. oErrorMessage - The error message if the return value is not equal to 0. A warning message if filled and the return value is 0. oExceptionID - An ID that refers to all error information. Use the XML functions in Exception to get all relevant information. Return: long - 0 if success - <> 0 if fail

PLMObject.CheckOut

DLL:	pdextpdmapi
Syntax:	<pre> long PLMObject.CheckOut(domain pobjt iObjectType, domain pdokey iKey, domain pdrevi iRevision, domain pdrevi iManualRevision, domain pdyesno iIsMajor, domain pdproj iProject, domain pdyesno iIgnoreProjectConfig, domain pdyesno iLockFilesOnCheckOut, ref domain tcguid.extend oTransactionGUID, ref domain tcmcs.s999m oErrorMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface for Revision Control Check-Out. All the parameters are mandatory. Pre: The transaction must have been already committed prior to calling this function. This function internally performs transaction handling. Post: None Input: - iObjectType - Object Type Possible values are: - pobjt.item (Item) - pobjt.document (Document) - pobjt.folder (Folder) - iKey - Object Key - iRevision - Object Revision - iManualRevision - Object Manual Revision - iIsMajor - Major Check-Out - iProject - Project - iIgnoreProjectConfig - Ignore Project Configuration Parameters when the Object is Check-Out.

	<code>iLockFilesOnCheckOut</code> - Indicates if files shall be automatically locked as part of Check-Out operation Output: <code>oTransactionGUID</code>- Transaction GUID. The Revision Control Mechanism processing objects will be inserted into the tables <code>pdcom013</code>- Revision Control Objects and <code>pdcom017</code>- Revision Control Validations with the <code>oTransaction</code> as guid. The calling method gets the objects and it's validations information using this field. <code>oErrorMessage</code> - The error message if the return value is not equal to 0. A warning message if filled and the return value is 0. <code>oExceptionID</code> - An ID that refers to all error information. Use the XML functions in Exception to get all relevant information. Return: long - 0 if success - <> 0 if fail

PLMObject.UndoCheckIn

DLL:	pdextpdmapi
Syntax:	<pre>long PLMObject.UndoCheckIn(domain pdobjt iObjectType, domain pdokey iKey, domain pdrevi iRevision, ref domain tcguid.extend oTransactionGUID, ref domain tcmcs.s999m oErrorMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This Public Interface for Undo Check-In. All the parameters are mandatory. Note that only the input object is affected. Pre: The transaction must have been already committed prior to calling this function. This function internally performs transaction handling. Post: None Input: <code>iObjectType</code> - Object Type Possible values are: - <code>pdobjt.item</code> (Item) - <code>pdobjt.document</code> (Document) - <code>pdobjt.folder</code> (Folder) <code>iKey</code> - Object Key <code>iRevision</code> - Object Revision Output: <code>oTransactionGUID</code>- Transaction GUID. The Revision Control Mechanism processing objects will be inserted into the tables <code>pdcom013</code>- Revision Control Objects and <code>pdcom017</code>- Revision Control Validations with the <code>oTransaction</code> as guid. The calling method gets the objects and it's validations information using this field. <code>oErrorMessage</code> - The error message if the return value is not equal to 0. A warning message if filled and the return value is 0. <code>oExceptionID</code> - An ID that refers to all error information. Use the XML functions in Exception to get all relevant

	information. Return: long - 0 if success - <> 0 if fail

PLMObject.UndoCheckOut

DLL:	pdextpdmapi
Syntax:	<pre> long PLMObject.UndoCheckOut (domain pobjt iObjectType, domain pdokey iKey, domain pdrevi iRevision, ref domain tcguid.extend oTransactionGUID, ref domain tcmcs.s999m oErrorMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This Public Interface for Undo Check-Out. All the parameters are mandatory. Note that apart from the input object, the linked objects (based on project configuration parameters) also gets affected. Pre: The transaction must have been already committed prior to calling this function. This function internally performs transaction handling. Post: None Input: iObjectType - Object Type Possible values are: - pobjt.item (Item) - pobjt.document (Document) - pobjt.folder (Folder) iKey - Object Key iRevision - Object Revision Output: oTransactionGUID - Transaction GUID. The Revision Control Mechanism processing objects will be inserted into the tables pdcom013- Revision Control Objects and pdcom017- Revision Control Validations with the oTransaction as guid. The calling method gets the objects and it's validations information using this field. oErrorMessage - The error message if the return value is not equal to 0. A warning message if filled and the return value is 0. oExceptionID - An ID that refers to all error information. Use the XML functions in Exception to get all relevant information. Return: long - 0 if success - <> 0 if fail

Chapter 40 Public Interfaces for Object Configuration Management

Public Interfaces for Workflow

The following functions are available:

[Workflow.DisableSubmit](#)

[Workflow.EnableSubmit](#)

[Workflow.Recall](#)

[Workflow.RecallByTypedObjectID](#)

[Workflow.Submit](#)

Workflow.DisableSubmit

DLL:	ttextocmap This function is available from KB2220087 .
Syntax:	<pre>long Workflow.DisableSubmit(domain tcmcs.str6 iObjectType, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: A Workflow Model contains Object Types which contain tables. This function disables the Workflow action Submit for the tables related to the Object Type. The related tables are checked out, but are not submitted for approval. The Workflow action Submit is disabled for updates in the current process. This function can be used in a process that updates multiple tables and the update in one table is automatically submitted for approval. Then another update on that table in the same process results in an error message: <Object> has the 'Pending' status. To avoid this error, the function Workflow.DisableSubmit() should be called before the first update on the table and function Workflow.EnableSubmit() should be called after the first update on the table. Example:

	Updating the Name of a Business Partner (tccom100.nama) will automatically update the Search Key of all roles of that Business Partner. If the Invoice-from role is active for the BP, tccom122.seak is updated and the tccom122 record is automatically checked out and submitted if Object Type "TCIFBP" is enabled. If the same process wants to make an update on the Invoice-from role for the BP (tccom122), the error Invoice-from Business Partner [<BP>] has the 'Pending' status is returned. The Workflow.DisableSubmit() function can be used before updating tccom100.nama to disable the submit for tccom122 (Object Type "TCIFBP"). The Workflow.EnableSubmit() function can be used after updating tccom100.nama and before updating Invoice-from BP data (tccom122) to enable the submit for tccom122 and other tables. Coding example: <pre> long return.value domain tcmcs.s999m exception.message long exception.id return.value = Workflow.DisableSubmit("TCIFBP", exception.message, exception.id) if return.value <> 0 then dal.set.error.message("@ " & exception.message) return(return.value) endif * Update tccom100.nama * dal.save.object("tccom100") * Automatically tccom122.seak is updated, object is checked out * and submit function is called; submit is not executed. return.value = Workflow.EnableSubmit(exception.message, exception.id) if return.value <> 0 then dal.set.error.message("@ " & exception.message) return(return.value) endif * Update tccom122 data * dal.save.object("tccom122") * Checked out Object is updated and submitted for Workflow approval. Pre: none Post: Function Workflow.EnableSubmit() must be called to enable the Workflow action Submit of all Object Types . Submitting the checked out Objects is required as well if this is not yet done. Input: iObjectType - object type (mandatory) Output: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - Success <> 0 - An error occurred </pre>

Workflow.EnableSubmit

DLL:	tcextocmapi				
	This function is available from KB2220087 .				
Syntax:	<pre>long Workflow.EnableSubmit(ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>				
Usage:	Expl: This function enables the Workflow action Submit for all Object Types that were disabled through function Workflow.DisableSubmit(). See Description of function Workflow.DisableSubmit() for further explanation. Pre: Function Workflow.DisableSubmit() was called one or more times (for one or multiple object types) in the current process. Post: Submitting the checked out Objects is required if this is not automatically done. Input: - Output: <table border="0"> <tr> <td>oExceptionMessage</td><td>- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.</td></tr> <tr> <td>oExceptionID</td><td>- An ID that refers to all error information. Use the functions in Exception to get all relevant information.</td></tr> </table> Return: 0 - Success <> 0 - An error occurred	oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.	oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.
oExceptionMessage	- The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID.				
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.				

Workflow.Recall

DLL:	tcextocmapi
	This function is available from KB2289885 .
Syntax:	<pre>long Workflow.Recall(domain tcmcs.tabl iRootTable, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>
Usage:	Expl: This function recalls the current record in an LN session that had been submitted for approval through ION Workflow. It publishes a Process.WorkflowBOD with actionCode "Replace" to interrupt the related Workflow in ION. This function should be called in the on.recall hook of the DAL of the table specified by iRootTable. Pre: An Object Type must be configured in the OCM Model for the specified iRootTable and this Object Type must be deployed.

	The related Workflow must be activated in ION. Post: NA Input: iRootTable Output: oExceptionMessage oExceptionID Return: 0 <> 0	- Root Table. Only tables in package "tx" are allowed. Mandatory. - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. - Success - An error occurred

Workflow.RecallByTypedObjectID

DLL:	tcextocmap This function is available from KB2310367 .	
Syntax:	<pre>long Workflow.RecallByTypedObjectID(domain ttocm.toid iTypedObjectID, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: This function recalls the record belonging to the specified Typed Object ID that had been submitted for approval through ION Workflow. It publishes a Process.WorkflowBOD with actionCode "Replace" to interrupt the related Workflow in ION. Example: an external application wants to update a sales order. The update fails because the workflow status of the sales order is 'Pending'. This function can be called to recall the sales order so that the sales order can be updated after the related Workflow was successfully interrupted in ION. Pre: An Object Type must be configured in the OCM Model for the table belonging to the specified Typed Object ID and this Object Type must be deployed. The related Workflow must be activated in ION. Retry point must be set. Post: Transaction must be committed or aborted. Input: iTypedObjectID Output: oExceptionMessage oExceptionID Return: 0 <> 0 - Workflow Typed Object ID belonging to the record that must be recalled. Mandatory. - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. - Success - An error occurred	

Workflow.Submit

DLL:	tcextocmapi		
	This function is available from KB2289885 .		
Syntax:	<pre>long Workflow.Submit(domain tcmcs.tabl iRootTable, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>		
Usage:	Expl: This function submits the current record in an LN session for approval through ION Workflow. It publishes a Process.WorkflowBOD with actionCode "Add" to initiate the related Workflow in ION. This function should be called in the on.submit hook of the DAL of the table specified by iRootTable. Pre: An Object Type must be configured in the OCM Model for the specified iRootTable and this Object Type must be deployed. The related Workflow must be activated in ION. Post: NA Input: iRootTable - Root Table. Only tables in package "tx" are allowed. Mandatory. Output: oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: 0 - Success <> 0 - An error occurred		

Chapter 41 Public Interfaces for Time Management

Public Interfaces for ProductionOrderHours

The following functions are available:

[ProductionOrderHours.CloseActiveLine](#)

[ProductionOrderHours.InsertActiveLine](#)

[ProductionOrderHours.StartOverview](#)

ProductionOrderHours.CloseActiveLine

DLL:	bpexttmmapi																			
	This function is available from KB2272425 .																			
Syntax:	long ProductionOrderHours.CloseActiveLine(domain tciemno iEmployee, domain tcccp.ynro iYear, domain tcccp.peri iPeriod, domain bpmdm.serd iSequence, domain tcdate iEndDateTime, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)																			
Usage:	Expl: This Public Interface is used to close the inserted active production order hours line resulting in update of the hours line with the spent time and possible new hours lines for next period(s). Pre: Caller must set retry-point Post: Caller must commit/abort transaction Input: <table><tr><td>iEmployee</td><td>- Employee</td><td>(Mandatory)</td></tr><tr><td>iYear</td><td>- Year</td><td>(Mandatory)</td></tr><tr><td>iPeriod</td><td>- Period</td><td>(Mandatory)</td></tr><tr><td>iSequence</td><td>- Sequence</td><td>(Mandatory)</td></tr><tr><td>iEndDateTime</td><td>- End Date and Time</td><td>(Mandatory)</td></tr></table> Output: oExceptionMessage <table><tr><td>- The last message if the return value is not equal to 0.</td></tr><tr><td>If more than one message is given, these are present in the oExceptionID</td></tr></table>			iEmployee	- Employee	(Mandatory)	iYear	- Year	(Mandatory)	iPeriod	- Period	(Mandatory)	iSequence	- Sequence	(Mandatory)	iEndDateTime	- End Date and Time	(Mandatory)	- The last message if the return value is not equal to 0.	If more than one message is given, these are present in the oExceptionID
iEmployee	- Employee	(Mandatory)																		
iYear	- Year	(Mandatory)																		
iPeriod	- Period	(Mandatory)																		
iSequence	- Sequence	(Mandatory)																		
iEndDateTime	- End Date and Time	(Mandatory)																		
- The last message if the return value is not equal to 0.																				
If more than one message is given, these are present in the oExceptionID																				

	oExceptionID Return: 0 <> 0	- An ID that refers to all error information. Use the functions in Exception to get all relevant information. - Active Production Order Hours Line has been closed. - Error while closing active Production Order Hours Line.

ProductionOrderHours.InsertActiveLine

DLL:	bpexttmmapi		
	This function is available from KB2272425 .		
Syntax:	long ProductionOrderHours.InsertActiveLine(domain tcncmp iLogisticCompany, domain tcorno iProductionOrder, domain tcopno iOperation, domain tcemno iEmployee, domain tcdate iStartDateTime, domain tcyesno iSetupTimeIndicator, domain tctano iTask, domain bptmm.mcno iMachine, domain tccwoc iWorkCenter, ref domain tcccp.yrno oYear, ref domain tcccp.peri oPeriod, ref domain bpmdm.serd oSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)		
Usage:	Expl: This function is used to create a production hours line with status Active. Pre: Caller must set retry-point. Post: Caller must commit/abort transaction. Input: iLogisticCompany - Logistic Company (Mandatory) iProductionOrder - Production Order (Mandatory) iOperation - Operation (Mandatory) iEmployee - Employee (Mandatory) iStartDateTime - Start Date and Time (Mandatory) iSetupTimeIndicator - Setup Time Indicator (Mandatory) Possible values : Yes (tcyesno.yes) No (tcyesno.no) iTask - Task (Mandatory) iMachine - Machine (Not Mandatory) iWorkCenter - Work Center (Mandatory) Output: oYear - Year oPeriod - Period oSequence - Sequence oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error		

	information. Use the functions in Exception to get all relevant information. Return: 0 <> 0	information. Use the functions in Exception to get all relevant information. - Active Production Order Hours Line Inserted. - Active Production Order Hours Line not inserted.

ProductionOrderHours.StartOverview

DLL:	bpexttmmapi This function is available from KB2274312.	
Syntax:	<pre>long ProductionOrderHours.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcemno iEmployee, domain tcccp.yrno iYear, domain tcccp.peri iPeriod, domain bptmm.regd iRegistrationDate, domain tcncmp iLogisticCompany, domain tcorno iProductionOrder, domain tcopno iOperation, domain tcppl.cgrp iTeam, ref domain tcemno oEmployee, ref domain tcccp.yrno oYear, ref domain tcccp.peri oPeriod, ref domain bpmdm.serid oSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)</pre>	
Usage:	Expl: This function starts session Production Order Hours - bptmm1120m000 (Overview). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "" (Session has no start filter) iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. Note: - Session starts on Index 2.	

	If specific session index is to be used, below Indices are only applicable:	
	Index 2	
	Index 3	
	Index 4	
	Index 5	
	Index 6	
iQueryExtend	Optional	
	A specific query to be used when zooming to this session.	
iEmployee	- Employee	(Not Mandatory)
iYear	- Year	(Not Mandatory)
iPeriod	- Period	(Not Mandatory)
iRegistrationDate	- Registration Date	(Not Mandatory)
iLogisticCompany	- Logistic Company	(Not Mandatory)
iProductionOrder	- Production Order	(Not Mandatory)
iOperation	- Operation	(Not Mandatory)
iTeam	- Team	(Not Mandatory)
Output: for iStartMode MODAL:		
oEmployee	- Selected Employee	
oYear	- Selected Year	
oPeriod	- Selected Period	
oSequence	- Selected Sequence	
oExceptionMessage	- The last message if the return value is not equal to 0.	
	If more than one message is given, these are present in the oExceptionID	
oExceptionID	- An ID that refers to all error information. Use the functions in Exception to get all relevant information.	
Return: 0	- Session started	
<> 0	- Otherwise.	

Public Interfaces for GeneralHours

The following functions are available:

[GeneralHours.StartOverview](#)

GeneralHours.StartOverview

DLL:	bpexttmapi
	This function is available from KB2274132 .

Syntax:	<pre> long GeneralHours.StartOverview(long iStartMode, domain tcmcs.st30 iStartFilter, long iSessionIndex, const string iQueryExtend(), domain tcemno iEmployee, domain tcccp.yrno iYear, domain tcccp.peri iPeriod, domain tcppl.cgrp iTeam, ref domain tcemno oEmployee, ref domain tcccp.yrno oYear, ref domain tcccp.peri oPeriod, ref domain bptmm.type oTransactionType, ref domain bpmdm.serd oSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>
Usage:	Expl: This function starts session General Hours - bptmm111lm200 (Overview). Input: iStartMode Specifies the start mode for the session. Possible values are: MODAL - The parent session is blocked until the child session exits, the session will be started as a zoom session. MODELESS - Parent and child are parallel sessions that can be manipulated simultaneously. iStartFilter Specifies the start filter that is to be applied to the started session. Possible values are: "" (Session has no start filter) iSessionIndex Specifies the session index that is to be used. Please be aware that the iStartFilter will overrule the index passed in this argument. So when not using a startfilter the session index will match the value of this variable. Note: - Session starts on Index 1. If specific session index is to be used, below Indices are only applicable: Index 1 Index 4 iQueryExtend Optional A specific query to be used when zooming to this session. iEmployee - Employee (Not Mandatory) iYear - Year (Not Mandatory) iPeriod - Period (Not Mandatory) iTeam - Team (Not Mandatory) Output: for iStartMode MODAL: oEmployee - Selected Employee oYear - Selected Year oPeriod - Selected Period oTransactionType - Selected Transaction Type oSequence - Selected Sequence oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present in the oExceptionID oExceptionID - An ID that refers to all error

	information. Use the functions in Exception to get all relevant information. - Session started - Otherwise. Return: 0 <> 0

Chapter 42 Public Interfaces for ZWF AG Shipment and Customs

Public Interfaces for SanctionList

The following functions are available:

[SanctionList.Check](#)

SanctionList.Check

DLL:	ttextzfcapi		
	This function is available from KB3637797 .		
Syntax:	<pre> domain tcmcs.long SanctionList.Check(domain tzzfc.sanc.doc iBaseType, domain tcncmp iFinancialComp, domain tzzfc.sanc.ain iAdditionalIntegration, domain tctran iTransactionID, domain tcorno iCode, boolean iShowPopupMessage, boolean iIsAdressCheck, boolean iIsUpdate, ref domain tcdate oDateOfCheck, ref domain tcmcs.long oCheckSequence, ref domain tcyesno oBlockProzess, ref domain tcmcs.str300m oMessage mb, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID) </pre>		
Usage:	Functions starts the ZWFAG Sanction List check. A 3GL Session is called. The function checks the sanction list information for the given address or document information and returns if the process should be blocked or not. Within the zwf module the whole checking process is logged. Function can send E-Mails depending on Sanction list parameter. This function is only usable with a licensed and working ZWFAG Sanction List Check. Pre: - Post: - Input: iBaseType - Base document type iFinancialComp - Financial company (for invoices)		

	iAdditionalIntegration - additional integration string iTransactionID - Transaction type (for invoices) iCode - Document or Address ID iShowPopupMessage - Show popupmessage with result y/n iIsAddressCheck - Is check for LN address code iIsUpdate - True for DAL_UPDATE - False for DAL_NEW Output: oDateOfCheck - Date of the check oCheckSequence - Checking sequence generated oBlockProzess - Blocking indicator oMessage - Message when function was successful oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: domain tcmcs.long - 0 success - DALHOOKERROR Error occurred in check

Public Interfaces for ExportControl

The following functions are available:

[ExportControl.Check](#)

ExportControl.Check

DLL:	tzextzfcapi
	This function is available from KB3637797 .
Syntax:	domain tcmcs.long ExportControl.Check(domain tcorno iCode, domain tzzfc.epc.doc iBaseType, domain tzzfc.sanc.ain iAdditionalIntegration, boolean iShowPopupMessage, ref domain tcdate oDateOfCheck, ref domain tzzfc.epc.did oCheckID, ref domain tcyesno oBlockProzess, ref domain tcmcs.str300m oMessage mb, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Functions starts the ZWFAG Export Control check. A 3GL Session is called. The function checks the export control information for the given document

	information and returns if the process should be blocked or not. Within the zwf module the whole checking process is logged. Function can send E-Mails depending on export control parameter. This function is only usable with a licensed and working ZWFAG Export Control Check. Pre: - Post: - Input: iCode - Document or Address ID iBaseType - Base document type iAdditionalIntegration - Additional Customised Integration Type iShowPoupMessage - Show poupmesssage with result y/n Output: oDateOfCheck - Date of the check oCheckID - Checking ID generated oBlockProzess - Blocking indicator oMessage - Message when function was successful oExceptionMessage - The last message if any message is found. If more than one message is given, these are present in the oExceptionID. oExceptionID - An ID that refers to the exception information. Use the functions in Exception to get all relevant information. Return: domain tcmcs.long - 0 success -1 error occurred in check

Chapter 43 Process Extensions

Process Extensions for AccountsReceivable

The following process extension(s) is/are available:

[AccountsReceivable.GetEffectiveDateForAging](#)

AccountsReceivable.GetEffectiveDateForAging

Get Effective Date to be used for calculating age of invoice or schedule.

This process extension is available from [KB3536486](#).

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to get a custom defined effective date in session Calculate Receivables Aging Analysis (tfacr2511m000). The session Calculate Receivables Aging Analysis (tfacr2511m000) will then calculate the age of an invoice/schedule based on this custom defined effective date.

To implement this process extension, you need to implement the following method(s):

tfext.acr0001.get.effective.date.for.aging.analysis

Syntax:	<pre>long tfext.acr0001.get.effective.date.for.aging.analysis(boolean i.schedule, ref domain tfgld.date io.effective.date)</pre>
Usage:	Expl: Use this Process Extension method to get a custom defined effective date in session Calculate Receivables Aging Analysis (tfacr2511m000). The session Calculate Receivables Aging Analysis (tfacr2511m000) will then calculate the age of an invoice/schedule based on this custom defined effective date. Only if an implementation is done for this Process Extension, and the Process Extension method is returning 0,

	and the input/output parameter io.effective.date is filled will the Document Date field (tfacr200.docd) and the Due date fields (tfacr200.dued & tfacr201.recd) be filled with this custom defined effective date. Then in session Calculate Receivables Aging Analysis (tfacr251lm000) this custom defined effective date is then used to calculate the age of an invoice/schedule. Pre: N.A. Post: N.A. Input: i.schedule IO: io.effective.date Return: 0 DALHOOKERROR	- This field indicates whether the function is called from an invoice or schedule line. If false this function is called from an invoice and the tfacr200 record is current. If true this function is called from a schedule line and the tfacr201 record is current. - This field is standard filled already. If the aging calculation is based on Document Date this field is filled with tfacr200.docd (Document Date). If the aging calculation is based on Due Date this field is filled with tfacr200.dued or tfacr201.recd (Due Date). This field can be influenced by the process extension based on own logic e.g. a CDF on invoice or schedule line. - Success - When an error occurs in setting the effective date.

Process Extensions for AssemblyOrder

The following process extension(s) is/are available:

[AssemblyOrder.SkipDelete](#)

[AssemblyOrder.SkipGenerate](#)

AssemblyOrder.SkipDelete

Skips Assembly Orders when deleting.

This process extension is available from [KB2230993](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension AssemblyOrder.SkipDelete can be used to skip certain Assembly Orders when requesting the deletion of a range of Assembly

	Orders. Sessions where this Process Extension can be implemented: - Delete Assembly Orders (tiasc2200m000) Fields that are available to be used in this Process Extension: - tiasc200.pvar - tiasc200.ncmp - tiasc200.asso - tiasc200.asst - tiasc200.block - tiasc200.porn - tiasc200.posi - tiasc200.asln - tiasc200.smcs - tiasc200.plod

AssemblyOrder.SkipGenerate

Skips Generation of Assembly Orders.

Technical information for this process extension:

Usage:	Process Extension AssemblyOrder.SkipGenerate can be used to skip generation of Assembly Orders for product variants. Extender may specify a message, to present information about the skip decision on the report. Sessions where this Process Extension can be implemented: - Generate Assembly Orders (tiapl3201m000) Fields that are available to be used in this Process Extension: - Key Fields from table: Product Variant (tiapl300) 1) tiapl300.cpva 2) tiapl300.item 3) tiapl300.smsc 4) tiapl300.reft 5) tiapl300.refo 6) tiapl300.apsv Note: (1)Tables must also be declared in the Process Extension. Pseudocode: Below you can find an example. Hook: Declarations <pre>table ttiapl300</pre> Hook: ext.skip.with.reason <pre>function extern boolean ext.skip.with.reason(ref string o.reason) { if <condition on tiapl300 = true> then o.reason = "Product Variant skipped because ..." return(true) endif return(false) }</pre>

Process Extensions for AssemblyPart

The following process extension(s) is/are available:

[AssemblyPart.SkipBackflush](#)

AssemblyPart.SkipBackflush

Skips backflushing of Assembly Parts.

This process extension is available from [KB2267502](#).

Technical information for this process extension:

Usage:	Process Extension AssemblyPart.SkipBackflush can be used to skip backflushing of Assembly Parts. Extender may specify a message, to present information about the skip decision on the report. Sessions where this Process Extension can be implemented: - Backflush Requirements (tiasc724lm000) Fields that are available to be used in this Process Extension: - All fields of table: Assembly Part Requirements (tiasc740) - All fields of table: Items (tcibd001) - The Warehouse Order in table: Clustered Line Station Orders (tiasc730) - tiasc730.worn Note: Tables must also be declared in the Process Extension. Pseudocode: Below you can find an example. Hook: Declarations <pre>table tiasc740</pre> Hook: ext.skip.with.reason <pre>function extern boolean ext.skip.with.reason(ref string o.reason) { if <condition on tiasc740 = true> then o.reason = "Assembly Part skipped because ..." return(true) endif return(false) }</pre>

Process Extensions for BillableLine

The following process extension(s) is/are available:

[BillableLine.CheckConfirmedAllowed](#)

[BillableLine.SkipComposing](#)

BillableLine.CheckConfirmedAllowed

Check whether, at adding Billable Line, the status may be set to Confirmed.

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to have additional checks before a Billable Line is set to 'Confirmed' at creating a Billable Line.

To implement this process extension, you need to implement the following method(s):

ciext.sli0002.check.billable.line.status.confirmed.allowed

Syntax:	<code>long ciext.sli0002.check.billable.line.status.confirmed.allowed(ref boolean o.line.status.confirmed.allowed)</code>
Usage:	Expl: Use this method to check whether, at adding Billable Line, the line status may be set to Confirmed. At moment of adding a Billable Line, LN will do own checks first. Only if the Billable Line status may be set to Confirmed according to the standard logic in LN, this method in the Process Extension is called. In this method, own checks can be implemented and the variable <code>o.line.status.confirmed.allowed</code> can be set. If the variable is set to false, the Billable Line status will be set to On Hold; otherwise, the status will be set to Confirmed. Fields that are available to be used in this Process Extension: although the Billable Line is to be added and not yet committed, the Billable Line fields from <code>cisli810</code> are already available. Note that <code>dal</code> messages set in this function will be ignored by

	the standard. Pre: N.A. Post: N.A. Input: N.A. Output: o.line.status.confirmed.allowed - Indicates whether the billable line status may be set to 'Confirmed'. Return: 0 - Success <> 0 - When an error occurs during checking; status will not be set to 'Confirmed'.

BillableLine.SkipComposing

Skip Billable Line when composing.

This process extension is available from [KB3520951](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension BillableLine.SkipComposing can be used to skip billable line when it is being composed. Sessions where this Process Extension can be implemented: - Compose/Print/Post (cisli2200m000) - Billable Lines (cisli8110m000) Fields that are available to be used in this Process Extension: - Billable Line (cisli810) Pseudocode: In the code below, the billable line from cisli810 whose Billable Amount (cisli810.amti) is greater than 1000.00 will be skipped during the composing process. Hook: Declarations <pre> table tcisli810 </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { if cisli810.amti > 1000.00 then /* If Billable line amount is greater than /* 1000.00 then skip this billable line during /* the composing. return(true) endif return (false) } </pre>

Process Extensions for BillOfLading

The following process extension(s) is/are available:

[BillOfLading.SkipShipmentLine](#)

BillOfLading.SkipShipmentLine

Skips Shipment Lines when Printing Bill of Lading.

To implement this process extension, you can use the information below:

Usage:	Process Extension BillOfLading.SkipShipmentLine can be used to skip Shipment Lines when Printing the Bill of Lading. Sessions where this Process Extension can be implemented: - Print Bills of Lading (whinh4470m000) - Automatic Processing of Shipping Documents when Shipment is Confirmed and the Print Bills of Lading is an automatic step. Fields that are available to be used in this Process Extension: - All fields of tables - Shipment (whinh430) - Shipment Lines (whinh431) - Load (whinh440) - Addresses (tccom130) - Carriers/LSP (tcmcs080) Note: tables must also be declared in the Process Extension. Pseudocode: Below you can find an example. Hook: Declarations <pre> table twhinh430 </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { if <condition on whinh430 = true> then return(true) endif return (false) } </pre>

Process Extensions for BlockedSalesOrderOrSalesOrderLine

The following process extension(s) is/are available:

[BlockedSalesOrderOrSalesOrderLine.SkipFirmRelease](#)

[BlockedSalesOrderOrSalesOrderLine.SkipSoftRelease](#)

BlockedSalesOrderOrSalesOrderLine.SkipFirmRelease

Skips Blocked Sales Order or Sales Order Line when executing Form Command 'Firm Release'.

This process extension is available from [KB3553809](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension BlockedSalesOrderOrSalesOrderLine.SkipFirmRelease can be used to skip specific Blocked Sales Orders or Sales Order Lines when executing Form Command "Firm Release". Session where this Process Extension can be implemented: - Blocked Sales Order (Lines) (tdsls4520m000) Fields that are available to be used in this Process Extension: - All fields of table Order (Line) Blocking (tdsls420) Note: Table must also be declared in the Process Extension. So skip conditions can be built on current tdsls420 data as instructed below. Pseudocode: Here you can find an example how to handle the conditions for blocked sales order (line)s. Hook: Declarations <pre>table ttdsls420</pre> Hook: ext.skip <pre>function extern boolean ext.skip() { if <condition on tdsls420 = true> then return(true) endif return (false) }</pre>

BlockedSalesOrderOrSalesOrderLine.SkipSoftRelease

Skips Blocked Sales Order or Sales Order Line when executing Form Command 'Soft Release'.

This process extension is available from [KB3553809](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension BlockedSalesOrderOrSalesOrderLine.SkipSoftRelease can be used to skip specific Blocked Sales Orders or Sales Order Lines when executing Form Command "Soft Release". Session where this Process Extension can be implemented: - Blocked Sales Order (Lines) (tdsls4520m000) Fields that are available to be used in this Process Extension: - All fields of table Order (Line) Blocking (tdsls420) Note: Table must also be declared in the Process Extension. So skip conditions can be built on current tdsls420 data as instructed below. Pseudocode: Here you can find an example how to handle the conditions for blocked sales order (line)s. Hook: Declarations <pre> table ttdsls420 Hook: ext.skip function extern boolean ext.skip() { if <condition on tdsls420 = true> then return(true) endif return (false) } </pre>

Process Extensions for BOD

The following process extension(s) is/are available:

[BOD.PublishCustomBOD](#)

BOD.PublishCustomBOD

Publish Custom BOD.

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to publish already existing data through an initial load for custom BODs. The session Publish Custom BODs (tcbod0299m000) is available as a framework to publish the custom BODs. By implementing the functions of this process extension, the session knows which BODs must be published, how their data must be selected from the database, which function must be called to publish the BOD for the selected data and how metadata has to be filled in the BODs. The following methods are present and must be implemented: - <code>tcext.bod0001.get.custom.bodnames()</code> This method is called when session Publish Custom BODs is started. With this method you specify which custom BODs need to be published. You can specify upto 50 BODs. For each BOD the session will display a separate tab, just like the standard BOD publishing sessions (for example Publish Order Management Transactional Data - tdbod0200m000). You can define custom fields in the process extension to add selection criteria to the tabs (see below). - <code>tcext.bod0001.get.initial.load.query()</code> This method is called once for each selected custom BOD when the processing is started in session Publish Custom BODs. Use this method to return the SQL query to read the root table of the custom BOD. In this query you can use the custom fields you defined in the process extension. For each selected record during the execution of this query, the BOD is created and published according the common options in the first tab of the Publish Custom BODs session. The following methods are present and are mutually exclusive - one of these two methods must be implemented for a custom BOD: - <code>tcext.bod0001.get.custom.bod.publish.function()</code> This method must be used if the SQL query in the <code>tcext.bod0001.get.initial.load.query()</code> method contains a GROUP BY clause or the SQL query returns more records than required for publishing the BODs and a custom function is needed to further decide if a selected record should publish a BOD or not. The custom function should contain functionality to decide if a BOD must be published, should provide metadata for the BOD, and should call the <code>BOD.Publish()</code> function to actually publish the BOD. This method is called for each record that is selected by the SQL query. Use this method to return the name of the function. - <code>tcext.bod0001.get.custom.bod.properties()</code> This method must be used if a BOD must be published for each record that is returned by the SQL query in the <code>tcext.bod0001.get.initial.load.query()</code> method and the SQL query does not contain a GROUP BY clause. A standard publish BOD function is called for each record that is selected by the SQL query. This method is called by this standard publish function during the BOD creation. Use this method to provide metadata for the BOD, like the DocumentID and the entity type and entity code that are used to set the accounting entity and location attributes in the BOD. For most custom BODs the standard BOD publish function can be used and the <code>tcext.bod0001.get.custom.bod.properties()</code> method should be implemented. Implementation of the

	<code>ttext.bod0001.get.custom.bod.publish.function()</code> method is an exception. Custom fields You can define custom fields in the process extension for selection criteria in the query to read the root table of the custom BODs. To include the custom fields in the session Publish Custom BODs, you need to create a session extension for this session and add the custom fields in the extension. Use the Add>Custom Field>Field from Process Extension option to add the custom fields. The fields will be added to the Configurable Fields tab of session Publish Custom BODs. With Form Personalization you can move the custom fields to the tab for the custom BOD to which the custom fields apply. Summary of steps: - Create process extension BOD.PublishCustomBOD - Implement function <code>ttext.bod0001.get.custom.bodnames()</code> - Create custom fields for the selection ranges - Implement function <code>ttext.bod0001.get.initial.load.query()</code> - Implement either function <code>ttext.bod0001.get.custom.bod.properties()</code> or implement function <code>ttext.bod0001.get.custom.bod.publish.function()</code> - If function <code>ttext.bod0001.get.custom.bod.publish.function()</code> is implemented, program publishing logic in the custom BOD publish function - Create session extension Publish Custom BODs (tcbod0299m000) - Add the custom fields from process extension BOD.PublishCustomBOD - Run session Publish Custom BODs (tcbod0299m000) - Personalize form to move the custom fields from the Configurable Fields tab to the BOD specific tab - Test the publishing of the custom BOD

To implement this process extension, you need to implement the following method(s):

ttext.bod0001.get.custom.bod.properties

Syntax:	<pre>long ttext.bod0001.get.custom.bod.properties(domain tcbod.name i.bodname, ref domain tcmcs.str100 o.documentID, ref domain tcmcs.str8 o.action.code, ref domain tcmcs.long o.bod.entity.type, ref domain tcmcs.str9 o.bod.entity.code)</pre>
Usage:	Use this method to return meta data fields that are needed in the BOD. Input: - <code>i.bodname</code> - custom BOD name Output: - <code>o.documentID</code> - documentID - <code>o.action.code</code> - BOD action code, possible values "Add", "Change", "Delete", "Replace" or "Canceled" - <code>o.bod.entity.type</code> - entity type used to set the accounting entity and location attributes. Possible values 0 (no type), 1 (Warehouse), 2 (Department), 3 (Project). For Master Data BODs value 0 is allowed. For other BOD types value must be 1, 2 or 3. - <code>o.bod.entity.code</code> - code related to the entity type.

	<pre> if entity type = 0, code is not used if entity type = 1, code must be a warehouse if entity type = 2, code must be a department if entity type = 3, code must be a project return: - 0 - OK - DALHOOKERROR - not OK Example of implementation: long retval retval = 0 o.documentID = "" o.action.code = "" o.bod.entity.type = 0 o.bod.entity.code = "" on case trim\$(i.bodname) case "CustomEmployeeBOD": o.documentID = tccom001.emno o.action.code = "Add" o.bod.entity.type = 0 o.bod.entity.code = "" break case "ColorBOD": o.documentID = "CO "& trim\$(txcpr001.colr) o.action.code = "Add" o.bod.entity.type = 0 o.bod.entity.code = "" break default: retval = DALHOOKERROR break endcase return(retval) </pre>

tcext.bod0001.get.custom.bod.publish.function

Syntax:	<pre> long tcext.bod0001.get.custom.bod.publish.function(domain tcbod.name i.bodname, ref domain tcmcs.str100 o.functionname, ref domain tcmcs.long o.instruction) </pre>
Usage:	Use this method to return the function and instruction that is needed to publish the BOD. When this method is not present or not implemented, default values will be used. The default values are: <pre> functionname = "tcbod.dll0005.publish.custom.bod" instruction = 1 </pre> Input: - i.bodname - custom BOD name Output: - o.functionname - the function name which has been created by the extension team to publish the BOD. The name of the

	DLL wherein this function exists, is derived from the function name. The function name must start with the package (tx), module, hardcoded ".dll", and the DLL number (4 characters). E.g. function name "txbod.dll0005.publish.custom.bod.tax.cty" must exist in DLL txboddll0005. - o.instruction - The instruction code. Possible values: 1 (PD_NO_SPECIFIC_COMMAND): the logic of the standard publish BOD function is used for counting and for publishing data. 2 (PD_CALL_FUNCTION_IN_SELECTDO_COUNT): the logic for counting and publishing data is programmed in the function of output argument o.functionname. In the current implementation, the value of o.instruction must be set to 2. return: - 0 - OK - DALHOOKERROR - not OK Example of implementation: <pre> o.functionname = "" o.instruction = 0 on case trim\$(i.bodname) case "CustomTaxCountryBOD": o.functionname = "txbod.dll0005.publish.custom.bod.tax.cty" o.instruction = 2 break default: break endcase return(0) </pre> Example of a custom BOD publish function: Following function could be programmed in txboddll0005 for an example where o.functionname = "txbod.dll0005.publish.custom.bod.tax.cty" and o.instruction = 2. The first argument and the last two arguments are fixed; in between each value returned by the initial load SQL query must exist as input argument <pre> function extern long txbod.dll0005.publish.custom.bod.tax.cty(const domain tcbod.mode i.mode, /* mandatory argument domain tcccty i.ccty, /*<<< Use one or more identifiers here ref long io.nr.bods.to.publish, /* mandatory argument ref long io.nr.bods.published) /* mandatory argument { long return.value domain tcmcs.s999m error.message long error.id if db.retry.hit() = 0 then io.nr.bods.to.publish = io.nr.bods.to.publish + 1 endif if i.mode <> tcbod.mode.count then return.value = BOD.Publish(</pre>

	<pre> "CustomTaxCountryBOD", "tcmcs036", "Add", 0, /* bod.entity.type ", /* bod.entity.code i.ccty, /* documentID "OnlyPublish", error.message, error.id, i.ccty) if return.value = 0 then io.nr.bods.published = io.nr.bods.published + 1 else dal.set.error.message("@&error.message) return(return.value) endif endif return(0) } </pre>

ttext.bod0001.get.custom.bodnames

Syntax:	<pre> long ttext.bod0001.get.custom.bodnames(ref domain tcbod.name o.bodnames() fixed, ref domain tcmcs.str70m o.labels() fixed mb, ref domain tcmcs.str8 o.root.tables() fixed, ref long o.indx) </pre>
Usage:	Use this method to define the custom BOD names for publishing. You can define upto 50 BODs. Each BOD must exist in the session BOD Implementation Registration (bobod1100m000) and the check box for the field "Standard" must not be selected. The BOD root table is mandatory for Master Data BODs and is optional for other BOD types. Input: N.A. Output: - o.bodnames - array with custom BOD names (max.50, length 70) - o.labels - array with custom BOD labels (max.50, length 70 multi-byte) - o.root.tables - array with custom BOD root tables (max.50, length 8) - o.indx - number of custom BODs in stack Return: - 0 - OK - DALHOOKERROR - not OK Example of implementation: <pre> o.indx = 3 alloc.mem(o.bodnames, 70, o.indx) alloc.mem(o.labels, 70, o.indx) alloc.mem(o.root.tables, 8, o.indx) /* CustomEmployeeBOD o.bodnames(1, 1) = "CustomEmployeeBOD" /* bodname </pre>

	<pre> o.labels(1, 1) = "Employee" /* label on session o.root.tables(1, 1) = "tccom001" /* root table /* ColorBOD o.bodnames(1, 2) = "ColorBOD" /* bodname o.labels(1, 2) = "Color" /* label on session o.root.tables(1, 2) = "txcpr001" /* root table /* CustomTaxCountryBOD o.bodnames(1, 3) = "CustomTaxCountryBOD" /* bodname o.labels(1, 3) = "Tax Country" /* label on session o.root.tables(1, 3) = "tcmcs036" /* root table return(0) </pre>

tcext.bod0001.get.initial.load.query

Syntax:	<pre> long tcext.bod0001.get.initial.load.query(domain tcbod.name i.bodname, ref string o.query()) </pre>
Usage:	Use this method to return the SQL query to read the index fields of the root table for each custom BOD. In the query you can use the values of the custom fields you defined in the process extension. The select statement of the query must only select the index fields of the root table. The values of the selected fields of the select statement are forwarded as input arguments for the standard publish BOD function or are forwarded as input arguments for the function that is defined in the <code>tcext.bod0001.get.custom.bod.publish.function()</code> method. Never implement a statement like "select <table>.*" Input: - i.bodname - custom BOD name Output: - o.query - dynamical query (max. length 2048) return: - 0 - OK - DALHOOKERROR - not OK Example of implementation: <pre> long retval retval = 0 o.query = "" on case trim\$(i.bodname) case "CustomEmployeeBOD": o.query = "select tccom001.emno " & "from tccom001 " & "where tccom001.emno inrange " & "{" & quoted.string(ext.empl.f) & "}" and " & "{" & quoted.string(ext.empl.t) & "}" " break case "ColorBOD": o.query = "select txcpr001.colr " & "from txcpr001 " & </pre>

	<pre> "where txcpr001.colr inrange " & "{" & quoted.string(ext.colr.f) & "} and " & "{" & quoted.string(ext.colr.t) & "} " break case "CustomTaxCountryBOD": o.query = "select tcmcs036.ccty " & "from tcmcs036 " & "where tcmcs036._index1 inrange " & "{" & quoted.string(ext.tax.country.f) & "} and " & "{" & quoted.string(ext.tax.country.t) & "} " & "group by tcmcs036.ccty " default: retval = DALHOOKERROR break endcase return(retval) </pre>

Process Extensions for CommissionsRebates

The following process extension(s) is/are available:

[CommissionsRebates.DetermineTaxCountry](#)

[CommissionsRebates.SkipCalculate](#)

CommissionsRebates.DetermineTaxCountry

Determine customer specific tax country for Commissions and Rebates.

This process extension is available from [KB3635923](#).

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to set a customer determined tax country from an extension to a value other than the value that was determined by standard LN logic. This tax country is then used in the CMS invoicing flow, as well as some master-data related sessions.

To implement this process extension, you need to implement the following method(s):

tdext.cms0001.commission.rebates.determine.tax.country

Syntax:	<pre> long tdext.cms0001.commission.rebates.determine.tax.country(domain tdcms.type i.type, domain tcorno i.sales.order, domain tcpono i.sales.order.line, domain tcpono i.sales.order.line.sequence, domain tcpono i.actual.delivery.line.sequence, domain tcpono i.invoice.line, domain tccom.bpid i.relation, domain tcpono i.serial.number, domain tcccty i.relation.country, domain tcccty i.tax.country.from.standard.logic, ref domain tcccty o.tax.country) </pre>
Usage:	Expl: Use this method to get a customer determined tax country from an extension. This tax country will be used in Commissions and Rebates. Note: This method allows you to return different tax countries as needed. Please be aware that during the invoice creation process, each time a different tax country is returned (compared to the previous one), a new invoice will be generated. If your custom tax country determination logic produces alternating or widely varying tax countries, this may result in the creation of many invoices. Invocation Contexts: The method can be invoked from two distinct contexts: * CMS Invoicing flow The first 8 input arguments uniquely identify a record from table tdcms050. These arguments can be used to read additional data from tdcms050, or from any of the sales order related tables. * Master-data context When the method is invoked in a master-data context, the first eight input arguments are left empty except for the input argument i.relation, which will be populated if available. This means that, in this context, only i.relation among the initial eight arguments may contain a value; all others remain unset. Fallback Logic: If either of the following situations occurs: * The extension returns a non-zero return-value, or * The tax-country is left empty Then, LN will automatically revert to the standard tax-country determination logic. Pre: NA Post: NA Input: i.type - Type Possible value(s): * tdcms.type.commission (Commission) * tdcms.type.rebate (Rebate) i.sales.order - Sales Order i.sales.order.line - Sales Order Line i.sales.order.line.sequence - Sales Sequence number i.actual.delivery.line.sequence - Sales Actual Delivery Sequence number i.invoice.line - Invoice Line

	<pre> i.relation i.serial.number i.relation.country i.tax.country.from.standard.logic Output: o.tax.country Return: 0 DALHOOKERROR </pre>	<pre> - Relation - Serial Number - Relation Country - Tax Country, as determined by the standard logic - Tax Country, as determined by the extension logic - Success - When an error occurs in the determination of the tax country </pre>

CommissionsRebates.SkipCalculate

Skips Commissions/Rebates when Calculating.

This process extension is available from [KB3620998](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension CommissionsRebates.SkipCalculate can be used to skip Commissions/Rebates when calculating the Commissions/Rebates. Session where this Process Extension can be implemented: - Calculate Commissions and Rebates (tdcms0240m000) Fields that are available to be used in this Process Extension: - All fields of table "Commissions/Rebates" (tdcms050) when removing non paid commissions and rebates. - Primary key fields of table "Sales Order Line History" (tdsls451) when calculation method 'Sales Order' is used. - Primary key fields of table "Sales Order Actual Delivery Line History" (tdsls456) when calculation method 'Sales Invoice' or 'Paid Sales Invoice' is used. External variable that is available to be used in this Process Extension: - proc_ext_skip_commissions_rebates_based_on [type: string(40)]. Supported values are: - remove_non_paid_commissions_rebates (for table tdcms050) - calculate_by_sales_order (for table tdsls451) - calculate_by_sales_invoice (for table tdsls456) Note: tables and external variable must also be declared in the Process Extension. The process extension must be applied to these three tables. Pseudocode: Below you can find an example how to handle the conditions for commissions/rebates, sales order history lines and sales order actual delivery history lines. Hook: Declarations <pre> table ttdcms050 table ttdsls451 table ttdsls456 </pre>

	<pre> extern string proc_ext_skip_commissions_rebates_based_on(40) Hook: ext.skip function extern boolean ext.skip() { on case proc_ext_skip_commissions_rebates_based_on case "remove_non_paid_commissions_rebates": if <condition on tdcms050 = true> then return(true) endif So in case of remove_non_paid_commissions_rebates commissions/rebates can be skipped based on tdcms050 data. break case "calculate_by_sales_order": if <condition on tdsls451 = true> then return(true) endif So in case of calculate_by_sales_order sales order history lines can be skipped based on tdsls451 primary key data. break case "calculate_by_sales_invoice": if <condition on tdsls456 = true> then return(true) endif So in case of calculate_by_sales_invoice sales order actual delivery history lines can be skipped based on tdsls456 primary key data. break default: break endcase return (false) } </pre>

Process Extensions for Common

The following process extension(s) is/are available:

[Common.DeterminePurchaseType](#)

Common.DeterminePurchaseType

Determine customer specific default Purchase Type.

This process extension is available from [KB2313055](#).

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to set a customer determined Purchase Type from an extension to a value other than the defaulted value based on Standard LN logic. This Purchase Type if valid is then used in the calling places where the purchase type is defaulted.

To implement this process extension, you need to implement the following method(s):

tcext.mcs0001.get.customer.defined.purchase.type

Syntax:	<pre> long tcext.mcs0001.get.customer.defined.purchase.type(domain tccom.bpid i.invoice.from.bp, domain tcorig i.order.origin, domain tccwoc i.department, domain tccitg i.item.group, domain tccprj i.project, domain tcitem i.item, domain tcccat i.project.category, domain tcynna i.direct.delivery, domain tcncmp i.financial.company, domain tcpsty i.purchase.type.from.standard, ref domain tcpsty o.purchase.type) </pre>																										
Usage:	Expl: This process extension gets the customer determined Purchase Type. The process extension is only called when the standard defaulting logic is executed correctly. The customer determined Purchase Type should exists in session 'Purchase Type (tcmcs2101m000)'. If no purchase type has been determined by customer then the standard defaulted purchase type is returned. Pre: NA Post: NA Input: <table border="0"> <tr> <td>i.invoice.from.bp</td><td>- Invoice-from Business Partner</td></tr> <tr> <td>i.order.origin</td><td>- Order Origin</td></tr> </table> Possible Values: <table border="0"> <tr> <td>- tcorig.pur</td><td>- Purchase</td></tr> <tr> <td>- tcorig.int.inv</td><td>- Internal Invoice</td></tr> <tr> <td>- tcorig.freight</td><td>- Freight</td></tr> <tr> <td>- tcorig.project</td><td>- Project</td></tr> <tr> <td>- tcorig.all</td><td>- All Origins</td></tr> <tr> <td>- tcorig.services.procm</td><td>- Service Procurement Order</td></tr> </table> <table border="0"> <tr> <td>i.department</td><td>- Department</td></tr> <tr> <td>i.item.group</td><td>- Item Group</td></tr> <tr> <td>i.project</td><td>- Project</td></tr> <tr> <td>i.item</td><td>- Item</td></tr> <tr> <td>i.project.category</td><td>- Project Category</td></tr> </table>	i.invoice.from.bp	- Invoice-from Business Partner	i.order.origin	- Order Origin	- tcorig.pur	- Purchase	- tcorig.int.inv	- Internal Invoice	- tcorig.freight	- Freight	- tcorig.project	- Project	- tcorig.all	- All Origins	- tcorig.services.procm	- Service Procurement Order	i.department	- Department	i.item.group	- Item Group	i.project	- Project	i.item	- Item	i.project.category	- Project Category
i.invoice.from.bp	- Invoice-from Business Partner																										
i.order.origin	- Order Origin																										
- tcorig.pur	- Purchase																										
- tcorig.int.inv	- Internal Invoice																										
- tcorig.freight	- Freight																										
- tcorig.project	- Project																										
- tcorig.all	- All Origins																										
- tcorig.services.procm	- Service Procurement Order																										
i.department	- Department																										
i.item.group	- Item Group																										
i.project	- Project																										
i.item	- Item																										
i.project.category	- Project Category																										

	<pre> i.direct.delivery - Direct Delivery Possible Values: - tcynna.yes - Yes - tcynna.no - No - tcynna.not.app- Not Applicable i.financial.company - Financial Company i.purchase.type - Purchase Type (Defaulted from Standard LN logic) Output: o.purchase.type - Purchase Type (Determined from Extension) Return: 0 - Success <> 0 - This value will be ignored and the standard defaulting logic will be executed. </pre>

Process Extensions for ComposeShippingStructure

The following process extension(s) is/are available:

[ComposeShippingStructure.CommandHandling](#)

ComposeShippingStructure.CommandHandling

Allows to disable commands and to define additional actions before and after execution within the sh.

Technical information for this process extension:

Usage:	With this Process Extension it is possible to disable a command and to execute additional actions before and/or after one of the following commands is executed in the compose shipping structure tree: - Freeze - Reopen - Confirm - Select Carrier/LSP - New Load For disabling commands the process extension is called when selecting an object in the compose shipping structure tree and for which the standard has decided the command should be enabled. Upon execution of a command, the process extension is called twice, once before the action will be performed and once when the command has been executed for each of the selected entities in the compose shipping structure tree.

To implement this process extension, you need to implement the following method(s):

whext.dll0007.compose.shipping.structure.command.is.allowed

Syntax:	<pre>boolean whext.dll0007.compose.shipping.structure.command.is.allowed(const string i.action(), domain whinh.load i.load, domain whinh.cntr i.shipping.container, domain whinh.shpm i.shipment, domain tcpono i.shipment.line)</pre>
Usage:	Expl: This function allows for disabling of commands that are present within the compose shipping structure tree. The enabling of the standard will first be evaluated, if the standard allows the execution of the command this function is called to evaluate if the command should also be allowed based on extension logic. It is not possible to enable a command which is disabled by the standard. Pre: N.a. Post: N.a. Input: i.action - This argument determines the action which is performed. Possible values are: - FreezeLoad - FreezeShippingContainer - FreezeShipment - FreezeShipmentLine - ReopenLoad - ReopenShippingContainer - ReopenShipment - ReopenShipmentLine - ConfirmLoad - ConfirmShippingContainer - ConfirmShipment - ConfirmShipmentLine - SelectCarrierLoad - NewLoad i.load - Load that is selected in the compose shipping structure tree. Only filled when either the Load or Shipping Container is selected i.shipping.container - Shipping Container that is selected in the compose shipping structure tree. Only filled when a Shipping Container is selected. i.shipment - Shipment that is selected in the compose shipping structure tree. Only filled when either the Shipment or Shipment Line is selected. i.shipment.line - Shipment Line that is selected in the compose shipping structure tree. Only filled when a Shipment Line is selected Output: Return: true --> command is allowed false --> command is not allowed

whext.dll0007.compose.shipping.structure.on.execute.command

Syntax:	<pre>long whext.dll0007.compose.shipping.structure.on.execute.command(const string i.action(), boolean i.before, domain whinh.load i.load, domain whinh.cntr i.shipping.container, domain whinh.shpm i.shipment, domain tcpono i.shipment.line)</pre>
Usage:	Expl: This function allows to implement extension logic within the execution of commands in the compose shipping structure tree. This function is called when one of the following commands is executed by the user: - Freeze - Reopen - Confirm - Select Carrier/LSP - New Load When the command is executed, this function is called twice to allow extension logic to be performed before and after execution of the command in the standard. The i.before argument indicates if the function is called before execution of the standard logic When this method is invoked, the transaction management is to be handled by the extension. The standard command execution will have it's own transaction management. Pre: N.a. Post: N.a. Input: i.action - This argument determines the action which is performed. Possible values are: - FreezeLoad - FreezeShippingContainer - FreezeShipment - FreezeShipmentLine - ReopenLoad - ReopenShippingContainer - ReopenShipment - ReopenShipmentLine - ConfirmLoad - ConfirmShippingContainer - ConfirmShipment - ConfirmShipmentLine - SelectCarrierLoad - NewLoad i.before - Boolean, determines at which moment the function is called, meaning: true: the function is called before the standard functionality of the command is executed. false: the function is called after the standard functionality of the command is executed. i.load - Load that is selected in the compose shipping structure tree. Only filled when either the Load or Shipping Container is selected i.shipping.container - Shipping Container that is selected in the

	<pre> compose shipping structure tree. Only filled when a Shipping Container is selected. i.shipment - Shipment that is selected in the compose shipping structure tree. Only filled when either the Shipment or Shipment Line is selected. i.shipment.line - Shipment Line that is selected in the compose shipping structure tree. Only filled when a Shipment Line is selected Output: Return: 0: Success / <> 0: Error </pre>

Process Extensions for ConfigurableItem

The following process extension(s) is/are available:

[ConfigurableItem.SkipCompileConstraints](#)

ConfigurableItem.SkipCompileConstraints

Skips Configurable Items when compiling constraints.

This process extension is available from [KB2296021](#).

To implement this process extension, you can use the information below:

Usage:	<pre> Process Extension ConfigurableItem.SkipCompileConstraints can be used to skip certain Configurable Items when requesting compilation of constraints. Sessions where this Process Extension can be implemented: - Compile Constraints by Configurable Item (tipcf2201m000) Fields that are available to be used in this Process Extension: - tiipd001 - all fields - tcibd001 - all fields </pre>

Process Extensions for CycleCountingData

The following process extension(s) is/are available:

[CycleCountingData.SkipDelete](#)

CycleCountingData.SkipDelete

Skips deleting unused cycle counting data.

This process extension is available from [KB3539935](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension CycleCountingData.SkipDelete can be used to skip deleting unused cycle counting data. Sessions where this Process Extension can be implemented: - Remove Unused Cycle Counting Data (whinh5240m000) Fields that are available to be used in this Process Extension: - All fields of tables - Cycle Counting Data (whinh540) Note: tables must also be declared in the Process Extension. Pseudocode: Below you can find an example. Hook: Declarations <pre> table twinh540 </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { if <condition on whinh540 = true> then return(true) endif return (false) } </pre>

Process Extensions for DeliveryNote

The following process extension(s) is/are available:

[DeliveryNote.SkipPrint](#)

DeliveryNote.SkipPrint

Skips Printing of Delivery Note.

This process extension is available from [KB2318267](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension DeliveryNote.SkipPrint can be used to skip the printing of a Delivery Note. Sessions where this Process Extension can be implemented: - Print Delivery Note (whinh4477m000) Fields that are available to be used in this Process Extension: - All fields of tables: - Delivery Notes (whinh435)

Process Extensions for EffectivityUnit

The following process extension(s) is/are available:

[EffectivityUnit.CheckInterchangeable](#)

EffectivityUnit.CheckInterchangeable

Check Interchangeability of Effectivity Units.

Technical information for this process extension:

Usage:	With this process extension, the rules for determining if 2 effectivity units are interchangeable can be customized.

To implement this process extension, you need to implement the following method(s):

tcext.uef0001.eff.units.are.interchg

Syntax:	<pre>long tcext.uef0001.eff.units.are.interchg(domain tcitem i.item, domain tcuef.effn i.alternative.eff.unit, domain tcuef.effn i.original.eff.unit,</pre>

		boolean	i.standard.interchangeable,
	ref	boolean	o.custom.rules.applied,
	ref	boolean	o.interchangeable)
Usage:	Expl:	Use this method to add to or replace the standard logic for the interchangeability check of 2 effectivity units. This method is called during the check on interchangeability, just before returning the result of the standard check. Example 1: replace logic under certain conditions <pre> o.custom.rules.applied = false if <some condition> then o.custom.rules.applied = true if <some other condition> then o.interchangeable = true else o.interchangeable = false endif endif return(0) </pre> Example 2: add logic when result standard check = not interchangeable <pre> o.custom.rules.applied = false if i.standard.interchangeable = false then if <some condition> then o.custom.rules.applied = true if <some other condition> then o.interchangeable = true else o.interchangeable = false endif endif endif return(0) </pre> Pre: Inputs are validated by the calling function If input is invalid (e.g. i.original.eff.unit does not exist), then this extension function is not called. Input: i.item The item for which the effectivity units are defined i.alternative.eff.unit Alternative effectivity unit i.original.eff.unit Original effectivity unit i.standard.interchangeable (true / false) Indicates if the 2 effectivity units are interchangeable according the standard logic. This information may or may not be used in the customized logic. Output o.custom.rules.applied (true / false): true: custom logic is applied. The calling function will use the result of the check in this extension (o.interchangeable) false: no custom logic is applied; the the calling function	

	<pre> will return the result of the standard check o.interchangeable (true / false): The result of the check in this extension. Is only relevant when o.custom.rules.applied = true. true: i.alternative.eff.unit is interchangeable with i.original.eff.unit false: i.alternative.eff.unit is NOT interchangeable with i.original.eff.unit. Return: 0 : OK <> 0 : Error in customized logic; use dal.set.error.message() to set a clear error message. The return code of the calling function (= standard check) will indicate 'data not valid'. </pre>

Process Extensions for ExceptionMessagesByItem

The following process extension(s) is/are available:

[ExceptionMessagesByItem.SkipPrint](#)

ExceptionMessagesByItem.SkipPrint

Skips printing Exception Messages By Item.

This process extension is available from [KB2264479](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension ExceptionMessagesByItem.SkipPrint can be used to skip printing. Sessions where this Process Extension can be implemented: - Print Exception Messages by Planner and Item (cproal425m000) Fields that are available to be used in this Process Extension: - cprao020.* (Exception Messages by Item) - tccom001.* (Employees - General) - cprpd400.* (Scenarios) - tcibd001.dsca

Process Extensions for FactoryTrackQuery

The following process extension(s) is/are available:

[FactoryTrackQuery.QueryExtend](#)

FactoryTrackQuery.QueryExtend

Extend Factory Track Queries.

Technical information for this process extension:

Usage:	With this Process Extension, standard Factory Track queries can be extended and custom queries can be created. Extensions on standard queries can be implemented in method <code>brext.qry0001.standard.query.extension</code>. Custom queries can be implemented in method <code>brext.qry0001.custom.query</code>. The following query BDEs can be extended: - IFTStdHUPackingQuery - IFTStdTimeTrack - IWMStdConsignmentWrhQuery - IWMStdExtQuery - IWMStdHoursQuery - IWMStdKanQuery - IWMStdPackingQuery - IWMStdQuery

To implement this process extension, you need to implement the following method(s):

brext.qry0001.custom.query

Syntax:	<pre>long brext.qry0001.custom.query(domain tcmcs.str40 i.query.bde, long i.query.number, domain tcmcs.s999 i.query.input, ref domain tcmcs.str999m o.error.message mb)</pre>
Usage:	Expl: Use this method to define custom Factory Track queries. Only query numbers greater than or equal to 100000 can be used for custom queries. Public interface <code>FactoryTrackQuery.CreateRow</code> can be used to create output rows in the query output. Public interface <code>FactoryTrackQuery.CreateElement</code> can be used to create custom elements in the current query output row. Example implementation:

```

IWMStdQuery 100001 lists the warehouses which have inventory on
hand for some item, ordered by descending quantity.
IWMStdQuery 100002 returns the buy-from business partner for
some lot code.

#pragma used dll obrextqryapi

#include <bic_dam>

function extern long brext.qry0001.custom.query(
    domain    tcmcs.str40    i.query.bde,
    long      i.query.number,
    domain    tcmcs.s999    i.query.input,
    ref       domain    tcmcs.str999m  o.error.message)
{
    long      ret.val
    long      exception.id
    long      dummy.long
    domain    tcitem        item
    domain    tccwar        warehouse
    domain    tcqivl        quantity.on.hand
    domain    tcclot        lot
    domain    tccom.bpid    business.partner
    domain    tcnama        business.partner.name

    on case i.query.bde
    case "IFTStdHUPackingQuery":
    case "IFTStdTimeTrack":
    case "IWMStdConsignmentWrhQuery":
    case "IWMStdExtQuery":
    case "IWMStdHoursQuery":
    case "IWMStdKanQuery":
    case "IWMStdPackingQuery":
        o.error.message = "Invalid query number"
        return(DALHOOKERROR)
    case "IWMStdQuery":
        on case i.query.number
        case 100001:
            item = strip$(i.query.input)

            select  whwmd215.cwar :warehouse,
                    whwmd215.qhnd :quantity.on.hand
            from    whwmd215
            where    whwmd215._index2 = { :item }
            order by whwmd215.qhnd desc
            selectdo
                ret.val = FactoryTrackQuery.CreateRow(
                    o.error.message,
                    exception.id)
                if ret.val <> 0 then
                    return(ret.val)
                endif

                ret.val = FactoryTrackQuery.CreateElement(
                    "Warehouse",
                    domainof(warehouse),
                    o.error.message,
                    exception.id,
                    warehouse)
                if ret.val <> 0 then
                    return(ret.val)

```

```

endif

ret.val = FactoryTrackQuery.CreateElement(
    "QuantityOnHand",
    domainof(quantity.on.hand),
    o.error.message,
    exception.id,
    quantity.on.hand)
if ret.val <> 0 then
    return(ret.val)
endif
selectempty
    o.error.message = "Inventory not found"
    return(DALHOOKERROR)
endselect

break
case 100002:

    item = ""
    lot = ""

    dummy.long = string.scan(
        i.query.input,
        "%s~%s",
        item,
        lot)

    business.partner = ""
    business.partner.name = ""

    select  whltc100.bfbp :business.partner,
            tccom100.nama :business.partner.name
    from    whltc100, tccom100
    where   whltc100._index1 = {      :item,
                                    :lot }
    and     whltc100.bfbp refers to tccom100
            unref clear

    as set with 1 rows
    selectdo
    endselect

    ret.val = FactoryTrackQuery.CreateRow(
        o.error.message,
        exception.id)

    if ret.val <> 0 then
        return(ret.val)
    endif

    ret.val = FactoryTrackQuery.CreateElement(
        "BusinessPartner",
        domainof(business.partner),
        o.error.message,
        exception.id,
        business.partner)

    if ret.val <> 0 then
        return(ret.val)
    endif

    ret.val = FactoryTrackQuery.CreateElement(
        "BusinessPartnerName",
        domainof(business.partner.name),

```

	<pre> o.error.message, exception.id, business.partner.name) if ret.val <> 0 then return(ret.val) endif break default: o.error.message = "Invalid query number" return(DALHOOKERROR) endcase break endcase return(0) } </pre>
Pre:	NA
Post:	NA
Input:	<pre> i.query.bde - The called query bde i.query.number - The called query number i.query.input - The query input </pre>
Output:	<pre> o.error.message - The error message in case of an error </pre>
Return:	<pre> 0 - Success <> 0 - Error </pre>

brext.qry0001.standard.query.extension

Syntax:	<pre> long brext.qry0001.standard.query.extension(domain tcmcs.str40 i.query.bde, long i.query.number, domain tcmcs.s999 i.query.input, domain tcmcs.s999 i.query.ext.input, ref boolean o.skip.row, ref domain tcmcs.str999m o.error.message mb) </pre>
Usage:	Expl: Use this method to extend standard Factory Track queries. This method will be called from the standard process for each query output row, just after it is created. The following scenarios are supported: - Add custom elements to the query output. - Remove output rows from the query output. - Return an error message. Public interface FactoryTrackQuery.CreateElement can be used to create custom elements in the current query output row. Public interface FactoryTrackQuery.GetElement can be used to read custom elements from the current query output row. Note: To prevent conflicts with possible future additions in the standard query, it is strongly advised to use prefix "Ext", "ext" or "ext " in custom element names.

Note: If this process extension removes all output rows for some query, then the query will return an error message.

The following list contains the queries which are extensible. Queries that are not mentioned are not extensible. The tables that are current in the process extension, are listed next to query numbers. Fields of those tables can be used without re-reading the table in this process extension. This holds also for customer defined fields in those tables.

```
IFTStdHUPackingQuery 1: whwmd530
IFTStdHUPackingQuery 2:
IFTStdHUPackingQuery 3: brpac911, whwmd530
IFTStdHUPackingQuery 4: whwmd530
IFTStdHUPackingQuery 5:
IFTStdHUPackingQuery 6: brpac911, whwmd530
IFTStdHUPackingQuery 7: brpac911, whwmd530
IFTStdHUPackingQuery 8: (**)
IFTStdHUPackingQuery 9: (**)
IFTStdHUPackingQuery 10: whwmd530
IFTStdHUPackingQuery 11: brpac911, whwmd530
IFTStdHUPackingQuery 12:
IFTStdHUPackingQuery 13: whwmd530
IFTStdHUPackingQuery 14:
IFTStdHUPackingQuery 15: whinh225
IFTStdHUPackingQuery 16: brpac911, whwmd530
IFTStdHUPackingQuery 17: brpac910 (**)
IFTStdHUPackingQuery 18: (**)
IFTStdHUPackingQuery 19: (**)
IFTStdHUPackingQuery 20: brpac911, whwmd530
IFTStdHUPackingQuery 21:
IFTStdHUPackingQuery 22:
IFTStdHUPackingQuery 23:
IFTStdTimeTrack 9:
IFTStdTimeTrack 10:
IFTStdTimeTrack 19: tisfc010
IFTStdTimeTrack 31: bpmdm050
IFTStdTimeTrack 129:
IFTStdTimeTrack 211: tisfc001
IFTStdTimeTrack 220: tipcs400, tisfc010 (**) (***)
IFTStdTimeTrack 247: tipcs400, tisfc010 (**) (***)
IFTStdTimeTrack 274: tipcs400, tisfc010 (**) (***)
IFTStdTimeTrack 275:
IFTStdTimeTrack 276:
IFTStdTimeTrack 277:
IFTStdTimeTrack 278: (**) (***)
IFTStdTimeTrack 279: tisfc014, tipcs405, tstdm211 (**) (***)
IFTStdTimeTrack 280: tpptc231 (**) (***)
IWMStdConsignmentWrhQuery 12: brcnd100
IWMStdExtQuery 12: brcnd100
IWMStdExtQuery 55: (**)
IWMStdExtQuery 56:
IWMStdExtQuery 57:
IWMStdExtQuery 58:
IWMStdExtQuery 59:
IWMStdExtQuery 60:
IWMStdExtQuery 61:
IWMStdExtQuery 100: brmcs999 (**)
IWMStdHoursQuery 1: brtmm100
IWMStdHoursQuery 2:
IWMStdHoursQuery 3:
```

Process Extensions

	IWMStdHoursQuery 4: brtmm102 IWMStdHoursQuery 5: bpmdm050 IWMStdHoursQuery 6: tcmcs048 IWMStdHoursQuery 7: tcmcs065 IWMStdHoursQuery 8: tcppl030 IWMStdHoursQuery 9: bpmdm050 IWMStdHoursQuery 10: tcmcs048 IWMStdHoursQuery 11: tcmcs048 IWMStdHoursQuery 12: IWMStdHoursQuery 21: IWMStdHoursQuery 22: brtmm101 IWMStdHoursQuery 23: (**) IWMStdHoursQuery 24: (**) IWMStdHoursQuery 25: (**) IWMStdHoursQuery 26: brtmm900 IWMStdHoursQuery 27: brtmm105 IWMStdHoursQuery 28: brtmm106 IWMStdHoursQuery 29: brtmm111 IWMStdHoursQuery 30: brtmm110 IWMStdHoursQuery 31: bpmdm050 IWMStdKanQuery 800: brkan000 (**) IWMStdKanQuery 801: brkan010, brkan030, whwmd211 (*) IWMStdKanQuery 802: IWMStdKanQuery 803: brkan010, brkan030, whwmd211 (*) IWMStdKanQuery 804: IWMStdKanQuery 805: IWMStdKanQuery 806: brkan010, brkan030, whwmd211 (*) IWMStdKanQuery 807: tisfc001 IWMStdPackingQuery 1: whinh225 IWMStdPackingQuery 2: IWMStdPackingQuery 3: IWMStdPackingQuery 4: brpac903 IWMStdPackingQuery 5: IWMStdPackingQuery 6: IWMStdPackingQuery 7: IWMStdPackingQuery 8: IWMStdPackingQuery 9: (**) IWMStdPackingQuery 10: IWMStdPackingQuery 11: brpac902 IWMStdPackingQuery 12: brpac901 IWMStdPackingQuery 13: brpac903 IWMStdPackingQuery 14: brpac902 IWMStdPackingQuery 15: brpac901 IWMStdPackingQuery 16: brpac905 IWMStdPackingQuery 17: whinh225 IWMStdPackingQuery 18: brpac901 IWMStdPackingQuery 21: whinh225 IWMStdPackingQuery 900: IWMStdQuery 1: IWMStdQuery 2: IWMStdQuery 3: whltc100 IWMStdQuery 4: whwmd300 IWMStdQuery 5: whinr140 IWMStdQuery 6: IWMStdQuery 7: IWMStdQuery 8: IWMStdQuery 9: IWMStdQuery 10: IWMStdQuery 11: IWMStdQuery 12: IWMStdQuery 13: IWMStdQuery 16:

	IWMStdQuery 17:
	IWMStdQuery 18: tisfc010
	IWMStdQuery 19: tisfc010
	IWMStdQuery 20: tisfc001
	IWMStdQuery 21:
	IWMStdQuery 22: whinr140
	IWMStdQuery 24:
	IWMStdQuery 26: whinr140
	IWMStdQuery 27: whwmd530
	IWMStdQuery 28:
	IWMStdQuery 29:
	IWMStdQuery 30:
	IWMStdQuery 31: tcibd401, timfc010
	IWMStdQuery 32: tcibd401, timfc010
	IWMStdQuery 34: whwmd300
	IWMStdQuery 35:
	IWMStdQuery 39: whinh225
	IWMStdQuery 40:
	IWMStdQuery 42:
	IWMStdQuery 44: whltc500
	IWMStdQuery 45:
	IWMStdQuery 49: tcppl030
	IWMStdQuery 50:
	IWMStdQuery 51:
	IWMStdQuery 54: timfc011
	IWMStdQuery 57:
	IWMStdQuery 59:
	IWMStdQuery 60: timfc011
	IWMStdQuery 63:
	IWMStdQuery 67: whinr140
	IWMStdQuery 68:
	IWMStdQuery 76: tcibd401, timfc010
	IWMStdQuery 93:
	IWMStdQuery 94:
	IWMStdQuery 95:
	IWMStdQuery 102:
	IWMStdQuery 112: tcppl101
	IWMStdQuery 140: tcmcs001
	IWMStdQuery 141:
	IWMStdQuery 142:
	IWMStdQuery 146:
	IWMStdQuery 148:
	IWMStdQuery 152: tcibd401
	IWMStdQuery 166:
	IWMStdQuery 169:
	IWMStdQuery 172:
	IWMStdQuery 173: (**)
	IWMStdQuery 174:
	IWMStdQuery 199: whinr100
	IWMStdQuery 200:
	IWMStdQuery 235: whltc500
	IWMStdQuery 236:
	IWMStdQuery 237: whinh225
	IWMStdQuery 238:
	IWMStdQuery 239: whinh225
	IWMStdQuery 240:
	IWMStdQuery 241:
	IWMStdQuery 242:
	IWMStdQuery 243:
	IWMStdQuery 244:
	IWMStdQuery 245:
	IWMStdQuery 246:

Process Extensions

	IWMStdQuery 248:
	IWMStdQuery 249:
	IWMStdQuery 250:
	IWMStdQuery 251:
	IWMStdQuery 252:
	IWMStdQuery 253:
	IWMStdQuery 254:
	IWMStdQuery 255: whinr140
	IWMStdQuery 256: whwmd530
	IWMStdQuery 257:
	IWMStdQuery 258:
	IWMStdQuery 259: whwmd410
	IWMStdQuery 260: (**)
	IWMStdQuery 261: (**)
	IWMStdQuery 262: whwmd530
	IWMStdQuery 263: whwmd530
	IWMStdQuery 265: whwmd530
	IWMStdQuery 266: whwmd530
	IWMStdQuery 267: whwmd530
	IWMStdQuery 268: whwmd530
	IWMStdQuery 269:
	IWMStdQuery 270: whwmd530
	IWMStdQuery 271: whwmd530
	IWMStdQuery 272:
	IWMStdQuery 273: whwmd405
	IWMStdQuery 276:
	IWMStdQuery 277:
	IWMStdQuery 278:
	IWMStdQuery 279:
	IWMStdQuery 280: whwmd300
	IWMStdQuery 281: whltc100
	IWMStdQuery 282:
	IWMStdQuery 283:
	IWMStdQuery 284:
	IWMStdQuery 285: whwmd410
	IWMStdQuery 286: whwmd405
	IWMStdQuery 287:
	IWMStdQuery 288:
	IWMStdQuery 289: whwmd530
	IWMStdQuery 290: tisfc010
	IWMStdQuery 291: tcibd401, timfc010
	IWMStdQuery 292:
	IWMStdQuery 293:
	IWMStdQuery 294:
	IWMStdQuery 295: tisfc001
	IWMStdQuery 296: whwmd530
	IWMStdQuery 297:
	IWMStdQuery 298: tcppl101
	IWMStdQuery 299: whwmd300
	IWMStdQuery 300:
	IWMStdQuery 301: (**)
	IWMStdQuery 302:
	IWMStdQuery 303: whwmd530
	IWMStdQuery 304:
	IWMStdQuery 305: whinh225
	IWMStdQuery 306:
	IWMStdQuery 307:
	IWMStdQuery 308: whwmd260
	IWMStdQuery 309:
	IWMStdQuery 310:
	IWMStdQuery 311:
	IWMStdQuery 312: whwmd260

```

IWMStdQuery 313:
IWMStdQuery 314: whwmd530
IWMStdQuery 315:
IWMStdQuery 316:
IWMStdQuery 317:
IWMStdQuery 318:
IWMStdQuery 319:
IWMStdQuery 320:
IWMStdQuery 321:
IWMStdQuery 323:
IWMStdQuery 324:
IWMStdQuery 325: timfc010
IWMStdQuery 326: tcibd401, timfc010
IWMStdQuery 327: tcmcs050
IWMStdQuery 328:
IWMStdQuery 329:
IWMStdQuery 330:
IWMStdQuery 331:
IWMStdQuery 332: (**)
IWMStdQuery 333:
IWMStdQuery 334: whltc100
IWMStdQuery 335: whwmd530
IWMStdQuery 336:
IWMStdQuery 337:
IWMStdQuery 338: tiasl145
IWMStdQuery 339: tiasc210
IWMStdQuery 340: whwmd530
IWMStdQuery 350: (**) (***)
IWMStdQuery 400:
IWMStdQuery 401: whinh225
IWMStdQuery 402:
IWMStdQuery 451: whwmd300 (**) (***)
IWMStdQuery 452: (**) (***)
IWMStdQuery 453: (**) (***)
IWMStdQuery 454: bpmdm050 (**) (***)
IWMStdQuery 455: whwmd530 (**) (***)
IWMStdQuery 456: whinr140 (**) (***)
IWMStdQuery 457: (**) (***)
IWMStdQuery 458: (**)
IWMStdQuery 459: tisfc001, tisfc006
IWMStdQuery 460: tcmcs001
IWMStdQuery 461: tcppl010 (**) (***)
IWMStdQuery 462: tisfc021 (**) (***)
IWMStdQuery 463: tisfc041 (**) (***)
IWMStdQuery 464: tisfc042 (**) (***)
IWMStdQuery 465: tisfc031 (**) (***)
IWMStdQuery 466: tisfc032 (**) (***)
IWMStdQuery 467: titrp011 (**) (***)
IWMStdQuery 470: qmncm001
IWMStdQuery 471: qmncm002
IWMStdQuery 472:
IWMStdQuery 473: tisfc200
IWMStdQuery 474:

```

(*) If kanban is integrated with warehousing, then whwmd211 is current. Otherwise brkan010 and brkan030 are current.

(**) It is not possible to remove output row(s).

(***) It is not possible to read output element "lastrecord" from the query output. When trying to read this output element by means of public interface FactoryTrackQuery.GetElement, the system will return an error.

Example implementation:

- Add custom element "ExtAddress" to the output of IWMStdQuery 1, which holds the address code of the warehouse.
- Return an error message in IWMStdQuery 2 if item signal is filled.
- Skip units in IWMStdQuery 140 which have no description.

```
#pragma used dll obrextqryapi

#include <bic_dam>

table      ttcms001      |* Units

function extern long brext.qry0001.standard.query.extension(
    domain    tcmcs.str40      i.query.bde,
    long      i.query.number,
    domain    tcmcs.s999      i.query.input,
    domain    tcmcs.s999      i.query.ext.input,
    ref       boolean         o.skip.row,
    ref       domain    tcmcs.str999m o.error.message)
{
    long      ret.val
    long      exception.id
    domain    tccwar      warehouse
    domain    tccom.cadr  address
    domain    tcitem      item
    domain    tcdsca      item.signal.description

    o.skip.row = false
    o.error.message = ""

    on case i.query.bde
    case "IFTStdHUPackingQuery":
        break
    case "IFTStdTimeTrack":
        break
    case "IWMStdConsignmentWrhQuery":
        break
    case "IWMStdExtQuery":
        break
    case "IWMStdHoursQuery":
        break
    case "IWMStdKanQuery":
        break
    case "IWMStdPackingQuery":
        break
    case "IWMStdQuery":
        on case i.query.number
        case 1:
            /* Read the warehouse from the query output.
            ret.val = FactoryTrackQuery.GetElement(
                                "whwmd200cwar",
                                o.error.message,
                                exception.id,
                                warehouse)

            if ret.val <> 0 then
                return(ret.val)
            endif

            /* Read the address code.
            select  tcmcs003.cadr :address
```

	<pre> from tcmcs003 where tcmcs003._index1 = { :warehouse } as set with 1 rows selectdo selectempty address = "" endselect /* Create the custom element. ret.val = FactoryTrackQuery.CreateElement("ExtAddress", domainof(address), o.error.message, exception.id, address) if ret.val <> 0 then return(ret.val) endif break case 2: /* Read the item from the query output. ret.val = FactoryTrackQuery.GetElement("itemcode", o.error.message, exception.id, item) if ret.val <> 0 then return(ret.val) endif /* Read the item signal select tcmcs018.dsca :item.signal.description from tcmcs018, tcibd001 where tcibd001._index1 = { :item } and tcibd001.csig <> "" and tcibd001.csig refers to tcmcs018 unref skip as set with 1 rows selectdo o.error.message = "Item Signal: " & item.signal.description return(DALHOOKERROR) endselect break case 140: /* Table tcmcs001 is current. if isspace(tcmcs001.dsca) then o.skip.row = true endif break default: break endcase endcase return(0) } </pre> Pre: NA Post: NA Input:

	<pre> i.query.bde - The called query bde i.query.number - The called query number i.query.input - The standard query input i.query.ext.input - The extended query input Output: o.skip.row - Remove the current query output row o.error.message - The error message in case of an error Return: 0 - Success <> 0 - Error </pre>

Process Extensions for FreightLoadBuilding

The following process extension(s) is/are available:

[FreightLoadBuilding.CustomBuildingConstraints](#)

FreightLoadBuilding.CustomBuildingConstraints

Allows to perform additional load building constraints during generate plan.

Technical information for this process extension:

Usage:	With this Process Extension it is possible to perform additional checks on the load building process. During the load building, the process will search for an existing loads to ship the goods with. There are however load building constraints which must force Infor LN to not combine the freight order line with a found load. This process extension allows a customer to define custom load building criteria to match the business case which is required.

To implement this process extension, you need to implement the following method(s):

fmext.dll0002.custom.load.building.constraints

Syntax:	<pre> long fmext.dll0002.custom.load.building.constraints(domain tcorno i.found.load, domain tcorno i.freight.order, domain tcpono i.freight.order.line, ref boolean o.use.found.load) </pre>
Usage:	Expl: This process extension allows for customer specific load building constraints. This process extension is called from the Infor LN standard, when generating a freight plan.

Note: This process extension is not called when composing a load structure. The constraints that are applicable for the load building should also be added as extension points for composing load, via dedicated table / field extensions on the related field, in line with the load building criteria.

The load building process will try to search for load which are already present to perform the load. Conditions apply on the combinations which are allowed based on order type specific load building criteria. This process extension allows to define custom building criteria to be validated prior to selecting the load for a newly created shipment and shipment line.

Table "Loads" (fmlbd400) is current, for the i.found.load.

Other data that is required to determine if the load can be used can be retrieved by making use of the freight order line key, which is also part of the input of this process extension. Setting the o.use.found.load to false will take care that the i.found.load will not be used as a custom criteria is not matching with the load and the related freight order line.

Within Warehousing there is also a process extension called "LoadBuilding.CustomBuildingConstraints" which will be triggered during the load building in Warehousing. It is advisable to apply the same customer specific building criteria in Warehousing, as otherwise the planning will be overruled by warehousing.

Note: If you execute queries on the standard Infor LN tables, bind the table fields to local variables to prevent disturbing the standard flow.

Pre:	N.a.	
Post:	N.a.	
Input:	i.found.load	- Load that is found during load building
	i.freight.order	- Freight Order
	i.freight.order.line	- Freight Order Line
Output:	o.use.found.load	- Indicator if the found load should be used (true/false)
Return:	0/DALHOOKERROR	

Process Extensions for FreightShipmentBuilding

The following process extension(s) is/are available:

[FreightShipmentBuilding.CustomBuildingConstraints](#)

FreightShipmentBuilding.CustomBuildingConstraints

Allows to perform additional shipment building constraints during generate plan.

Technical information for this process extension:

Usage:	With this Process Extension it is possible to perform additional checks on the shipment building process. During the shipment building, the process will search for an existing shipment to ship the goods with. There are however shipment building constraints which must force Infor LN to not combine the freight order line with a found shipment. This process extension allows a customer to define custom shipment building criteria to match the business case which is required.

To implement this process extension, you need to implement the following method(s):

fmext.dll0001.custom.shipment.building.constraints

Syntax:	<pre>long fmext.dll0001.custom.shipment.building.constraints(domain tcorno i.found.shipment, domain tcorno i.freight.order, domain tcpono i.freight.order.line, ref boolean o.use.found.shipment)</pre>
Usage:	Expl: This process extension allows for customer specific shipment building constraints. This process extension is called from the Infor LN standard, when generating a freight plan. Note: This process extension is not called when composing a shipment structure. The constraints that are applicable for the shipment building should also be added as extension points for composing shipments, via dedicated table / field extensions on the related field, in line with the shipment building criteria. The shipment building process will try to search for shipments which are already present to perform the shipment. Conditions apply on the combinations which are allowed based on order type specific shipment building criteria. This process extension allows to define custom building criteria to be validated prior to selecting the shipment for a newly created shipment line. Table "Shipments" (fmlbd300) is current, for the i.found.shipment. Other data that is required to determine if the shipment can be used can be retrieved by making use of the freight order line key, which is also part of the input of this process extension. Setting the o.use.found.shipment to false will take care that the i.found.shipment will not be used as a custom criteria is not matching with the shipment and the related freight order line. Within Warehousing there is also a process extension called "ShipmentBuilding.CustomBuildingConstraints" which will be triggered during the shipment building in Warehousing.

	It is advisable to apply the same customer specific building criteria in Warehousing, as otherwise the planning will be overruled by warehousing. Note: If you execute queries on the standard Infor LN tables, bind the table fields to local variables to prevent disturbing the standard flow. Pre: N.a. Post: N.a. Input: i.found.shipment - Shipment that is found during shipment building i.freight.order - Freight Order i.freight.order.line - Freight Order Line Output: o.use.found.shipment - Indicator if the found shipment should be used (true/false) Return: 0/DALHOOKERROR

Process Extensions for GenerateFieldChangeOrderLine

The following process extension(s) is/are available:

[GenerateFieldChangeOrderLine.SkipSerializedItem](#)

GenerateFieldChangeOrderLine.SkipSerializedItem

Skips generating Field Change Order Line for Serialized Items.

This process extension is available from [KB3597898](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension GenerateFieldChangeOrderLine.SkipSerializedItem can be used to skip generating field change order lines for specific serialized items. Sessions where this Process Extension can be implemented: - Generate Field Change Order Lines (tssoc5210m000). Fields that are available to be used in this Process Extension: - Key fields of tscfg200 - tscfg200.item (Item) - tscfg200.sern (Serial Number) These fields can be used to read e.g. table tscfg200. Note: table must also be declared in the Process Extension. Pseudocode:

	Below you can find an example. <pre> Hook: Declarations table tscfg200 * Serialized Items Hook: ext.skip function extern boolean ext.skip() { if <condition on tscfg200 = true> then return(true) endif return (false) } </pre>

Process Extensions for GeneratePlan

The following process extension(s) is/are available:

[GeneratePlan.SkipOrderLine](#)

GeneratePlan.SkipOrderLine

Allows to skip Freight Order line from generating a freight plan.

This process extension is available from [KB3565345](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension GeneratePlan.SkipOrderLine can be used to skip Freight Order line from a load planning. For all instances where freight plans are generated this process extension will be invoked. Fields that are available to be used in this Process Extension: - All fields of tables: - Freight Order (fmfoc200) - Freight Order Lines (fmfoc201) In case of replanning the existing plans: - All fields of tables: - Freight Order Lines to Be Replanned (fmlbd020)

Process Extensions for HandlingUnitBuilding

The following process extension(s) is/are available:

[HandlingUnitBuilding.CustomShipmentLine](#)

HandlingUnitBuilding.CustomShipmentLine

Allows to perform additional structure updates after a handling unit is reused or filled up.

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to implement modifications in the handling unit structure when a handling unit is linked to the shipment line as a result of picking inventory or (re)generation of handling units on a shipment line. Note: This process extension will be called after the handling unit has been linked to the shipment line. A handling unit can either be generated, filled-up (anonymous stock is put in an already existing handling unit) or reused (picked handling unit is reused, or part of a picked handling unit is reused).

To implement this process extension, you need to implement the following method(s):

whext.wmd0001.custom.shipment.line

Syntax:	<pre>long whext.wmd0001.custom.shipment.line(domain whinh.shpm i.shipment, domain tcpono i.shipment.line, domain whhuid i.handling.unit, boolean i.handling.unit.reused, ref domain tcmcs.str132m o.error.message.array() fixed mb)</pre>
Usage:	Expl: This function can be implemented to influence the handling unit building process during creation of handling units for the given shipment line. This function will be called from the standard process just after a handling unit is linked to the given shipment line. If a handling unit is re-used from inventory (during picking / release outbound advice when picking is not in the outbound process), the flag <code>i.handling.unit.reused</code> will be set to true. Do not change the record buffers with the additional logic, or if required, store and restore the record buffer prior to changing records, or inserting new records. Error messages can be logged in the error message array. The messages that are to be logged should be started with a @. Implementation note: it is wise to start the message with a

	dedicated hard coded string, so it is clear to the end user that the message logged is set in the Extension. <pre> Pre: NA Post: NA Input: i.shipment - shipment i.shipment.line - shipment line i.handling.unit - handling unit i.handling.unit.reused - picked handling unit is reused Output: o.error.message.array() Return: 0/DALHOOKERROR </pre>

Process Extensions for InboundAdvice

The following process extension(s) is/are available:

[InboundAdvice.SkipPutawayAdvice](#)

InboundAdvice.SkipPutawayAdvice

Skips Inbound Advice during Put Away Inbound Advice.

This process extension is available from [KB3519267](#).

Technical information for this process extension:

Usage:	Process Extension InboundAdvice.SkipPutawayAdvice can be used to skip inbound advice during the put away inbound advice process. Extender may specify a message, to present information about the skip decision. For all instances where inbound advices are put away this process extension will be invoked, this also includes automatic inbound processing. Fields that are available to be used in this Process Extension: - All fields of tables: - Inbound Advice (whinh215) Pseudocode: Below you can find an example. Hook: Declarations <pre> table twinh215 </pre> Hook: ext.skip.with.reason

	<pre> function extern boolean ext.skip.with.reason(ref string o.reason) { if <condition on whinh215 = true> then o.reason = "Putaway skipped because ..." return(true) endif return(false) } </pre>

Process Extensions for IntegrationTransaction

The following process extension(s) is/are available:

[IntegrationTransaction.SkipPrint](#)

IntegrationTransaction.SkipPrint

Skip Print Integration Transactions.

This process extension is available from [KB2327961](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension IntegrationTransaction.SkipPrint can be used to skip Printing Integration Transactions. Sessions where this Process Extension can be implemented: - Print Integration transactions (tfgld4482m000) Fields that are available to be used in this Process Extension: - All fields of table: Integration Transactions (tfgld482)

Process Extensions for IntrastatTransaction

The following process extension(s) is/are available:

[IntrastatTransaction.CheckBlockedForReporting](#)

IntrastatTransaction.CheckBlockedForReporting

Check whether, at adding Intrastat Transaction, the status must be set to Blocked for Reporting.

This process extension is available from [KB3550580](#).

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to have additional checks before an Intrastat Transaction record is written, to set the status 'Blocked for Reporting' at creating an Intrastat Transaction.

To implement this process extension, you need to implement the following method(s):

lpext.ita0001.check.intrastat.transaction.blocked.for.reporting

Syntax:	<code>long lpext.ita0001.check.intrastat.transaction.blocked.for.reporting(ref boolean o.blocked.for.reporting)</code>
Usage:	Expl: Use this method to check whether, at adding Intrastat Transaction record, the status must be set to "Blocked for Reporting". At moment of adding an Intrastat Transaction, LN will do checks first. Only if the Intrastat Transaction status is not set to "Blocked for Reporting" according to the standard logic in LN, this method in the Process Extension is called. In this method, own checks can be implemented and the variable <code>o.blocked.for.reporting</code> can be set. If the variable is set to false, the Intrastat Transaction status will not be changed. Otherwise, the status will be set to "Blocked for Reporting". Fields that can be used in this process extension: Although the Intrastat Transaction is about to be added and not yet committed, the Intrastat Transaction fields from <code>lpita360</code> table are already available. Note that dal messages set in this function will be ignored by the standard. ----- Start of Example of Implementation Pseudocode: <pre>Hook: Declarations table tlpita360 * Intrastat Transactions Hook: lpext.ita0001.get.intrastat.transaction.blocked.for.reporting function extern long lpext.ita0001.get.intrastat.transaction.blocked.for.reporting(ref boolean o.blocked.for.reporting) { domain <domain> field.value</pre>

```

        /* Default value is set to False.
        o.blocked.for.reporting = false

        select  cisli310.<field>:field.value
        from    cisli310
        where   cisli310._index1 = {
                                :lpita360.fcmp,
                                :lpita360.tran,
                                :lpita360.idoc,
                                :lpita360.line}

        as set with 1 rows
        selectdo
            |*****
            |* If customized criteria are met
            |* so the Intrastat transaction status
            |* is set to "Blocked for Reporting".
            |*****
            if field.value = <customized criteria> then
                o.blocked.for.reporting = true
            endif
        endselect

        return(0)
    }
End of Example of Implementation
-----

```

Pre: N.A.

Post: N.A.

Input: N.A.

Output: o.blocked.for.reporting - Indicates whether the Intrastat Transaction record status must be set to "Blocked for Reporting".

Return: 0

<> 0

- Success
- Returning a value <> 0 will abort the creation of Intrastat Transactions.

Process Extensions for InventoryAgingAnalysis

The following process extension(s) is/are available:

[InventoryAgingAnalysis.Custom](#)

[InventoryAgingAnalysis.Skip](#)

InventoryAgingAnalysis.Custom

Cust. Inv. Aging Analysis.

This process extension is available from [KB3514346](#).

Technical information for this process extension:

Usage:	With this Process Extension it is possible to customize the Inventory Aging Analysis report. This process extension allows a customer to add/fill additional report fields or change the content of existing report variables by filling of a TX table.

To implement this process extension, you need to implement the following method(s):

whext.dll0011.customize.inventory.aging.analysis

Syntax:	<pre>long whext.dll0011.customize.inventory.aging.analysis(domain tcmcs.long i.storage.time.in.seconds, domain tcdate i.reference.date, domain tcqivl i.quantity, domain tcamnt i.amount)</pre>
Usage:	Expl: This process extension allows for customer specific inventory analysis constraints. This process extension is called from the Infor LN standard, after function assign.quantity.and.amount.to.period is called. Table "Inventory Receipt Transactions" (whina112) is current. Input fields can for example be used extend the number of Aging Periods on the report or overwrite the quantity/amount for a certain Period by filling of a TX table. Input fields are the same as those who are used for calling function assign.quantity.and.amount.to.period. Note: If you execute queries on the standard Infor LN tables, bind the table fields to local variables to prevent disturbing the standard flow. Pre: N.a. Post: N.a. Input: i.storage.time.in.seconds i.reference.date i.quantity i.amount Output: N.a. Return: 0/DALHOOKERROR

InventoryAgingAnalysis.Skip

Skips Inventory during Inventory Aging Analysis.

This process extension is available from [KB3514346](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension InventoryAgingAnalysis.Skip can be used to skip Inventory during Inventory Aging Analysis. Sessions where this Process Extension can be implemented: - Perform Inventory Aging Analysis (whinal440m000) Fields that are available to be used in this Process Extension: - All fields of tables - Inventory Receipt Transactions (whinal12) Note: tables must also be declared in the Process Extension. Pseudocode: Below you can find an example. Hook: Declarations <pre> table twhinal12 Hook: ext.skip function extern boolean ext.skip() { if <condition on whinal12 = true> then return(true) endif return (false) } </pre>

Process Extensions for InventorySearchEngine

The following process extension(s) is/are available:

[InventorySearchEngine.CheckHandlingUnit](#)

[InventorySearchEngine.CheckStockPoint](#)

InventorySearchEngine.CheckHandlingUnit

Allows to perform additional checks or skips of found Handling Units.

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to perform additional checks on the inventory search engine related to the found handling units, which is user during the generation of outbound advices

	(whinh4201m000) and the Backflushing process.

To implement this process extension, you need to implement the following method(s):

whext.dll0002.check.handling.unit

Syntax:	<pre>long whext.dll0002.check.handling.unit(ref boolean o.skip.handling.unit, ref domain tcmcs.str132m o.error.message.array() fixed mb)</pre>
Usage:	Expl: Use this method to validate/check or skip a handling unit from the inventory search engine. This interface will be called during the inventory search engine after all standard validations and checks are performed. When a handling unit must be skipped from the selection the output variable o.skip.handling.unit should be set to true. When also an error must be given to either the advice log or the end user the output variable o.error.message.array() can be set. When the handling unit is not to be skipped, the error message array will still be used for logging messages in the advice log, allowing the extension to tell the user which checks have been performed, prior to using the inventory. The messages that are to be logged should be started with a @. Implementation note: It is wise to start the message with a dedicated hard coded string, so it is clear to the end user that the message logged is set in the Extension. Following tables are current when this extension is triggered: - tcibd001 - Item - tcmcs003 - Warehouses - whinh220 - Outbound Order Line - whinh200 - Warehouse Order Header - whinh480 - Planned Loads/Shipments (when load planning part of the search engine) - whinh430 - Shipments (when projected shipments selection is enabled for the search engine) - whinh431 - Shipment Lines (when projected shipments selection is enabled for the search engine) - whwmd530 - Handling Units Pre: NA Post: NA Input: NA Output: o.skip.handling.unit o.error.message.array() Return: 0/DALHOOKERROR

InventorySearchEngine.CheckStockPoint

Allows to perform preview of the stock point selection and skips of stock points.

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to perform additional checks on the inventory search engine related to the found stock points, which is used during the generation of outbound advices (whinh4201m000) and the Backflushing process. There are 2 extension points within the process extension, which can be implemented individually, the first one (whext.dll0001.stock.point.selection) will be called from the standard prior to actually selecting the stock points. This allows the extender to get a preview of the stock points which will be selected within the inventory search engine. This can be used to evaluate which stock points will be part of the selection, during the execution of the inventory search engine. The other extension point (whext.dll0001.check.stock.point) will be called when the inventory search engine has selected the actual stock point, in which it will be possible to skip the stock point and additionally provide a message which will be forwarded to the advice log, if necessary. These extension functions share the same declaration and will allow for usage of the same variables (arrays) to determine what should be done. Another option is to use a tx table to store what is the optimal selection, based on the stock points which are considered for allocations during the inventory search engine. An example: Suppose there are a couple of locations in the system which require higher priority when advising / picking inventory as opposed to the location type (bulk / pick). So the following locations: BULK1 - Bulk - Prio 10 - Available inventory: ITEM X 20 pcs BULK2 - Bulk - Prio 20 - Available inventory: ITEM X 40 pcs BULK3 - Bulk - Prio 30 - Available inventory: ITEM X 60 pcs PICK1 - Pick - Prio 10 - Available inventory: ITEM X 80 pcs The standard will first always consider the locations of type Pick, but suppose the customer wants to use the bulk locations first, which has the highest priority. An order is created for 80 pcs of the item ITEM X. When selecting the inventory in the standard flow, the location PICK1 will be selected, as that has the exact quantity 80 pcs and it is a pick location, suppose the BULK3 location and BULK2 location are to be used first, because the inventory located there needs to be consumed prior to the inventory on the picking location, the extension can be implemented as such: Implement whext.dll0001.stock.point.selection to see which stock points are about to be selected by the standard. Based on extension logic, the customer can write logic to determine which locations should be used and which should be skipped. When the inventory search engine will select the inventory, the function whext.dll0001.check.stock.point will be executed, in which the extension can skip the selected stock point. So suppose PICK1 should be skipped, BULK1 should be skipped, and the inventory on BULK2 and BULK 3 can be used. The standard will first come with PICK1 which will be skipped by the extension, then BULK1 which will also be skipped. Then BULK2 will be selected, only 40 pcs are available

	on that location, so an outbound advice of 40 pcs will be created, next the BULK3 will be selected from which also 40 pcs will be available, as a result also an outbound advice will be created, for the remaining 40 pcs.

To implement this process extension, you need to implement the following method(s):

whext.dll0001.check.stock.point

Syntax:	<pre>long whext.dll0001.check.stock.point(ref boolean o.skip.stock.point, ref domain tcmcs.str132m o.error.message.array() fixed mb)</pre>
Usage:	Expl: Use this method to validate/check or skip a stock point from the inventory search engine. This interface will be called during the inventory search engine after all standard validations and checks are performed. When a stock point must be skipped from the selection the output variable o.skip.stock.point should be set to true. When also an error must be given to either the advice log or the end user the output variable o.error.message.array() can be set. When the stock point is not to be skipped, the error message array will still be used for logging messages in the advice log, allowing the extension to tell the user which checks have been performed, prior to using the inventory. The messages that are to be logged should be started with a @. Implementation note: It is wise to start the message with a dedicated hard coded string, so it is clear to the end user that the message logged is set in the Extension. Following tables are current when this extension is triggered: - tcibd001 - Item - tcmcs003 - Warehouses - whinh220 - Outbound Order Line - whinh200 - Warehouse Order Header - whinh480 - Planned Loads/Shipments (when load planning part of the search engine) - whinh430 - Shipments (when projected shipments selection is enabled for the search engine) - whinh431 - Shipment Lines (when projected shipments selection is enabled for the search engine) - whinr150 - Inventory Structure - whinr140 - Inventory - whwmd300 - Locations (for location controlled warehouses) - whltc500 - Serials by Warehouse (for serialized items) - whltc105 - Lots by Warehouse (for lot controlled items) Pre: NA Post: NA Input: NA Output: o.skip.stock.point o.error.message.array() Return: 0/DALHOOKERROR

Syntax:	long whext.dll0001.stock.point.selection(long i.query.id)
Usage:	Expl: Use this method to fetch the stock points which are going to be selected by the Inventory Search Engine. The i.query.id is the parsed query by the standard which can be fetched using this process extension. The flow is as follows: - Inventory Search Engine is started - Functional loop is executed to determine the best stock points, based on the setup applicable for the order line that is to be advised. In this loop a query is created. - When the process extension is implemented: - The query id (i.query.id) is the query that can be fetched in this extension point - Standard fetches the stock points and will generate an outbound advice / backflush if this is allowed. Technically the Infor LN standard will perform the sql.exec and sql.break, within the process extension only the sql.fetch is needed to be executed on the i.query.id. Fetching a record can of course result in a fetched record or not. Upon fetching the query.id, the following tables will be current, and can be used in this process extension: - whinr150 - Inventory Structure - whinr140 - Inventory - whwmd300 - Locations (for location controlled warehouses) Pre: This process extension will run within a logical transaction, so it is not allowed to use abort.transaction() or commit.transaction(), db.retry.point() should also not be set. Post: NA Input: i.query.id Output: NA Return: 0/DALHOOKERROR

The following process extension(s) is/are available:

Invoice.CustomComposingCriteriaMet

[Invoice.SkipCreditNoteLinking](#)

[Invoice.SkipPrintInvoice](#)

Invoice.CombineAdvancesIsAllowed

Combine advance billable lines into one invoice when billable lines are being composed together.

This process extension is available from [KB3552629](#).

Technical information for this process extension:

Usage:	Currently LN standard doesn't allow to compose advance billable lines into one invoice and creates separate invoice for each advances. With this Process Extension, it is possible to combine advance billable lines when the billable lines are being composed into one invoice.

To implement this process extension, you need to implement the following method(s):

ciext.sli0004.combine.advances.is.allowed

Syntax:	<pre>long ciext.sli0004.combine.advances.is.allowed(ref boolean o.combine.advances.is.allowed)</pre>
Usage:	Expl: Note:- LN standard does not allow to compose advance billable lines into one single invoice. Composing two or more advance billable lines into one single invoice can have consequences depending on Implementation and Parameter settings for which LN standard will not be able to provide a solution. Use this method to check whether, at the time of composing, the advance billable lines can be composed together in one invoice. At moment of composing Billable Line, LN will do own checks first. According to the standard logic in LN, the Advance invoices cannot be combined in one invoice and creates separate invoice for each advance invoice. But with this process extension, it will be possible to combine two or more advance billable lines into one invoice. In this method, own checks can be implemented and the variable o.combine.advances.is.allowed can be set. If the variable is set to true, then the advance billable lines will be composed together in one single invoice subject to composing criteria. Fields that are available to be used in this Process Extension: - All fields of table: Billable lines (cisli810) - All fields of table: Although the Invoice is not yet

	committed, the Invoice Header (cisli305) fields are already available. The Invoice header key fields can be used to read other tables like Invoice Lines (cisli310) or Invoice Lines - Additional Fields (cisli311). The dal messages set in this function will be ignored by the standard. Pre: N.A. Post: N.A. Input: N.A. Output: o.combine.advance.invoice.allowed - Indicates whether the advance billable lines can be combined. Return: 0 <> 0 - Success - Error

Invoice.CustomComposingCriteriaMet

Determines whether billable line should be composed in the current invoice number or not.

This process extension is available from [KB2305642](#).

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to have custom composing criteria defined by customer/extender to check whether the current billable line should be composed in the current invoice or not. If this Process Extension returns false then program will look for the next available invoice number and if available then this process extension is called again. If no invoice number is available then a new invoice number will be created and this process extension will not be called again.

To implement this process extension, you need to implement the following method(s):

ciext.sli0003.invoice.custom.composing.criteria.met

Syntax:	<pre>long ciext.sli0003.invoice.custom.composing.criteria.met(ref boolean o.custom.composing.criteria.met)</pre>
Usage:	Expl: Use this method to check your own defined composing criteria, whether the current billable line should be composed in the current invoice number or not. At the time of composing of billable lines together, LN performs its own check to determine whether the lines can be composed together based on the standard composing criteria logic. Only when standard logic in LN allows the current billable line

	to be composed together in the current invoice, this method in the Process Extension is called. Default value of "o.custom.composing.criteria.met" is True. In this method, own checks can be implemented and the variable o.custom.composing.criteria.met can be set. If the variable is set to false, then the current billable line will not be composed in the current invoice but then it will look for the next available invoice number and if available then this process extension is called again. If no invoice number is available then a new invoice number will be created and this process extension will not be called again. If the variable is set to true then the current billable line will be added in the current invoice number. Fields that are available to be used in this Process Extension: - All fields of table: Billable lines (cisli810) - All fields of table: Although the Invoice is not yet committed, the Invoice Header (cisli305) fields are already available. The Invoice header key fields can be used to read other tables like Invoice Lines (cisli310) or Invoice Lines - Additional Fields (cisli311). Note: 1) Dal messages set in this function will be ignored by the standard. 2) User must use bind variables when they read data, otherwise the standard flow may be affected. 3) External variables that are available to be used in this Process. ----- Start of Example of Implementation Requirement - Each Packing slip should be composed separately in a new invoice number. Solution - Packing slip is not available in the invoicing method to make it one of the standard composing criteria. So Customer can use this process extension to make it one of the custom composing criteria. Packing slip is available on the Billable line and is available in the Invoice line Additional Fields (cisli311). Pseudocode - Hook: Declarations <pre> table tcisli305 * Invoice Header table tcisli810 * Billable Lines </pre> Hook: ciext.sli0003.invoice.custom.composing.criteria.met <pre> function extern long ciext.sli0003.invoice.custom.composing.criteria.met(ref boolean o.custom.composing.criteria.met) { domain cisli.fldv2 field.value fixed domain cisli.fldc field.code fixed </pre>

	<pre> field.code = "PKSP" /* Default value is set to True. o.custom.composing.criteria.met = true select cisli311.fldv:field.value from cisli311 where cisli311._index1 = {:cisli305.sfcpl, :cisli305.tran, :cisli305.idoc} and cisli311.fldc = :field.code and cisli311.fldv <> :cisli810.pksp as set with 1 rows selectdo ***** * If different Packing Slip is * available in the invoice then the * current billable line should not be * inserted into given invoice number * so the composing criteria does not * meet here. ***** o.custom.composing.criteria.met = false endselect return (0) } End of Example of Implementation ----- Pre: N.A. Post: N.A. Input: N.A. Output: o.custom.composing.criteria.met - Determines whether current billable line should be composed in the current invoice or not. Return: 0 DALHOOKERROR - Success - When an error occurs during checking; the invoice will not be composed together. </pre>

Invoice.CustomXMLInvoicing

Define custom elements to be used in XML invoicing.

Technical information for this process extension:

Usage:	With this Process Extension it is possible to define extra custom data elements to be used in the XML invoice (ciext.sli0001.define.custom.xml.data.elements)and give a value to those custom defined data elements by means of ciext.sli0001.get.value.for.custom.xml.data.element.

To implement this process extension, you need to implement the following method(s):

ciext.sli0001.define.custom.xml.data.elements

Syntax:	<pre> long ciext.sli0001.define.custom.xml.data.elements(ref long o.number.of.elements, ref domain tcfdnm.c o.custom.element.field.names() fixed, ref domain tcdscg o.custom.element.descriptions() fixed mb, ref domain tcxml.tgtp o.custom.element.data.types(), ref domain cisli.xils o.custom.element.sections()) </pre>
Usage:	Expl: Use this method to define extra custom XML data elements to be used in the XML data elements. The custom XML data elements can then be used for mapping to an XML attribute in session XML Invoice Layouts (cisli1151m000). If a custom XML data element is mapped, during building the XML invoice, a value must be retrieved for the element. For that, the following method must be used: <pre>ciext.sli0001.get.value.for.custom.xml.data.element</pre> Notes: Keep in mind that when a custom XML data element is used in a XML layout and that XML layout is used in an invoice you should not stop supporting that custom XML data element. Because reprinting of an old invoice can result in a different invoice when the custom XML data element is not available or properly filled anymore. Do not use the prefixes "h.", "f.", "l." or "c." for a "custom element field name" because those prefixes are used in the standard implementation, this excludes the risk that when in the standard a new element field name is introduced it gets the same name as a custom element field name. ----- Start of Example of Implementation <pre> long domain.length.tcfdnm.c long domain.length.tcdscg long dummy.convert rdi.domain.string("tcfdnm.c", domain.length.tcfdnm.c, * ref dummy.convert) * ref rdi.domain.string("tcdscg", domain.length.tcdscg, * ref dummy.convert) * ref o.number.of.elements = 4 * 4 data elements will be defined and * returned if (alloc.mem(o.custom.element.field.names, domain.length.tcfdnm.c, o.number.of.elements) <> 0) or (alloc.mem(o.custom.element.descriptions, domain.length.tcdscg, o.number.of.elements) <> 0) or </pre>

```

        (alloc.mem( o.custom.element.data.types,
                    o.number.of.elements) <> 0 ) or
        (alloc.mem( o.custom.element.sections,
                    o.number.of.elements) <> 0 ) then
            dal.set.error.message( "cisli0266")
            |* Allocation of memory failure, process stopped.
            return (DALHOOKERROR)
        endif

    |* Data elements for Invoice Header
    o.custom.element.field.names(1,1) = "ext_fld_h001"
    o.custom.element.descriptions(1,1) = "Extensibility h001 description"
    o.custom.element.data.types(1) = tcxml.tgtp.string
    o.custom.element.sections(1) = cisli.xils.header

    o.custom.element.field.names(1,2) = "ext_fld_h002"
    o.custom.element.descriptions(1,2) = "Extensibility h002 description"
    o.custom.element.data.types(2) = tcxml.tgtp.dec.number
    o.custom.element.sections(2) = cisli.xils.header

    |* Data elements for Invoice Line
    o.custom.element.field.names(1,3) = "ext_fld_l001"
    o.custom.element.descriptions(1,3) = "Extensibility l001 description"
    o.custom.element.data.types(3) = tcxml.tgtp.string
    o.custom.element.sections(3) = cisli.xils.line

    |* Data elements for Tax Summary
    o.custom.element.field.names(1,4) = "ext_fld_s001"
    o.custom.element.descriptions(1,4) = "Extensibility s001 description"
    o.custom.element.data.types(4) = tcxml.tgtp.string
    o.custom.element.sections(4) = cisli.xils.tax.summary

    return (0)

    End of Example of Implementation
    -----

Pre:    N.A.
Post:   N.A.
Input:  N.A.
Output: o.number.of.elements      - Number of custom XML data elements
        o.custom.element.field.names - Array with field names
        o.custom.element.descriptions - Array with descriptions.
                                         Maximum of 70 characters.
        o.custom.element.data.types  - Array data types
                                         Data type should be one of the
                                         following values:
                                         tcxml.tgtp.string
                                         tcxml.tgtp.dec.number
                                         tcxml.tgtp.bool
                                         tcxml.tgtp.date
                                         tcxml.tgtp.utc.date
                                         tcxml.tgtp.enum
        o.custom.element.sections    - Array with section
                                         Section should be one of the
                                         following values:
                                         cisli.xils.header
                                         cisli.xils.line
                                         cisli.xils.tax.summary

Return: 0          - Success
        <> 0       - When an error occurs in setting

```

	custom xml data elements.

ciext.sli0001.get.value.for.custom.xml.data.element

Syntax:	<pre> long ciext.sli0001.get.value.for.custom.xml.data.element(domain tcxml.layd i.xml.layout, domain tcxml.tgtp i.xml.tag.type, domain tcxml.tgid i.xml.tag.id, domain tcxml.tag i.xml.tag mb, domain cisli.xils i.xml.section, long i.section.key.object, domain tcfdnm.c i.custom.element.field.name, ref string o.custom.element.value(), ref long o.custom.element.type) </pre>
Usage:	Expl: Use this method to get a value and type for the given custom XML data element field. ----- Start of Example of Implementation <pre> table tcisli315 #define NO_FIELD_TYPE 0 #define AMOUNT_FIELD 10 #define UTC_DATE_FIELD 20 #define BOOLEAN_FIELD 70 #define QUANTITY_FIELD 80 #define PRICE_PERCENTAGE_FIELD 90 #include <bic_dam> #include <bic_ext> /* For Process Extensions o.custom.element.type = NO_FIELD_TYPE on case strip\$(i.custom.element.field.name) case "ext_fld_h001": o.custom.element.value = "This is the string value of field ext_fld_h001" o.custom.element.type = NO_FIELD_TYPE break case "ext_fld_h002": o.custom.element.value = "8.8" o.custom.element.type = AMOUNT_FIELD break case "ext_fld_h003": o.custom.element.value = "1" /* = true o.custom.element.type = BOOLEAN_FIELD break case "ext_fld_h004": o.custom.element.value = enum.descr\$("tcinv", tcinv.travel, "2") o.custom.element.type = NO_FIELD_TYPE break case "ext_fld_h005": o.custom.element.value = str\$(utc.num()) </pre>

```

        o.custom.element.type = UTC_DATE_FIELD
        break
    case "ext_fld_h007":
        o.custom.element.value = "12.43"
        o.custom.element.type = QUANTITY_FIELD
        break
    case "ext_fld_h008":
        o.custom.element.value = "1.1"
        o.custom.element.type = PRICE_PERCENTAGE_FIELD
        break
    case "ext_fld_l001":
        o.custom.element.value = "This is the string value of ext_fld_l001"
        o.custom.element.type = NO_FIELD_TYPE
        break
    case "ext_fld_s001":
        object.to.keyfields(i.section.key.object)
        o.custom.element.value =
            concat$("/",
                "Custom TaxSummary Field: ",
                cisli315.sfcf,
                cisli315.tran,
                cisli315.idoc,
                cisli315.ccty,
                cisli315.cvat)

        o.custom.element.type = NO_FIELD_TYPE    |* other field
        break
    endcase

return(0)

```

End of Example of Implementation

```

Pre:      N.A.
Post:     N.A.
Input:    i.xml.layout      - The xml invoice layout code
          i.xml.tag.type    - type of the xml.tag
          i.xml.tag.id      - xml tag id
          i.xml.tag         - xml tag name
          i.xml.section     - section where the tag is used for:
                           cixils.header - invoice header
                           cixils.line   - invoice line
                           cixils.tax.summary - invoice tax summary
          - The above xml fields are indicating in which (part of)
            XML layout "i.custom.element.field.name" is used.

          i.section.key.object
          - This field contains the current value of the key fields
            Those key fields can be used when the value of
            the i.custom.element.field.name is stored in
            one of the tables mentioned below.
            Those keyfields can be restored by using
            "object.to.keyfields".
            In case i.xml.section reads: cixils.header the
            keyvalues of cisli305 are restored
            In case i.xml.section reads: cixils.line the
            keyvalues of cisli310 are restored
            In case i.xml.section reads: cixils.tax.summary the
            keyvalues of cisli315 are restored
            If other then the keyfields are needed a query
            must be done.

```

	<code>i.custom.element.field.name</code> - Custom XML data element for which a value must be retrieved. Output: <code>o.custom.element.value</code> - The value of the custom element field <code>o.custom.element.type</code> - The element type of the custom element field, the "<code>o.custom.element.value</code>" will be formatted or converted according the element type value. - The next element types are supported: <pre>#define AMOUNT_FIELD 10 - o.custom.element.value is assumed to be in Invoice header currency, and formatted according (manual) amount format of the XML #define UTC_DATE_FIELD 20 - o.custom.element.value will be formatted according format settings of the XML layout #define BOOLEAN_FIELD 70 - o.custom.element.value contains a long and will be converted as follows 0 = "false", >0 = "true" #define QUANTITY_FIELD 80 - o.custom.element.value will be reformatted according decimal and group signs defined on the XML layout and domain "tcqsl1" (Example 10000.00 --> 10.000,00) #define PRICE_PERCENTAGE_FIELD 90 - o.custom.element.value will be reformatted according decimal and group signs defined on the XML layout and domain "tcpric" (Example 10.00 --> 10,00) #define NO_FIELD_TYPE 0 - o.custom.element.value will be used without formatting or conversion. Use this type for strings or when you want to deviate from the element types above.</pre> Return: 0 - Success <> 0 - When an error occurs in getting a value for custom xml data element.

Invoice.SkipCreditNoteLinking

Skip Credit Note Linking to the original invoice at the time of posting of invoice.

This process extension is available from [KB3551578](#).

To implement this process extension, you can use the information below:

Usage:	When an Invoice is being posted, as per LN Standard logic, if Credit Note is allowed to be linked to the original invoice, then it will be automatically linked. Process Extension Invoice.SkipCreditNoteLinking can be used to skip the automatic linking of credit note to the original invoice at the time of posting of invoice.

	Fields that are available to be used in this Process Extension: - Invoice Header (cisli305) Pseudocode: Hook: Declarations <pre> table tcisli305 </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { if cisli305.itbp = "BP0000001" then /* In case of Invoice-to BP is equal to above /* Skip automatic linking of Credit Note at the /* time of posting of Invoice. return(true) endif /* Automatically link the Credit Note if applicable at /* the time of posting of Invoice. return (false) } </pre>

Invoice.SkipPrintInvoice

Skips Printing Invoice.

To implement this process extension, you can use the information below:

Usage:	Process Extension Invoice.SkipPrintInvoice can be used to skip invoices when printing the invoices in Central Invoicing. Sessions where this Process Extension can be implemented: - Print Invoices (cisli2400m000, cisli2405m000) Fields that are available to be used in this Process Extension: - Key fields of cisli305 - cisli305.sfcf (Financial Company) - cisli305.tran (Transaction Type) - cisli305.idoc (Invoice Number) Those fields can be used to read table cisli305 (binded) - proc_ext_print_option - Indicates if a draft or original invoice is printed or a reprint is done. - possible values are: - cisli.prno.draft - cisli.prno.original - cisli.prno.reprint Note: tables must also be declared in the Process Extension. Pseudocode: In de code below reprinting of an invoice with InvoiceTo Business Partner

	<pre> "TDCUS0014" is skipped. Hook: Declarations table tcisli305 extern domain cisli.prno proc_ext_print_option domain tccom.bpid invoice.to.bp Hook: ext.skip function extern boolean ext.skip() { invoice.to.bp = "" on case proc_ext_print_option case cisli.prno.draft: case cisli.prno.original: break case cisli.prno.reprint: select cisli305.itbp:invoice.to.bp from cisli305 where cisli305._index1 = { :cisli305.sfcpr, :cisli305.tran, :cisli305.idoc} as set with 1 rows selectdo if invoice.to.bp = "TDCUS0014" then return(true) endif endselect break default: break endcase return (false) } </pre>

Process Extensions for InvoiceFromBP

The following process extension(s) is/are available:

[InvoiceFromBP.SkipPrintAgingAnalysis](#)

[InvoiceFromBP.SkipPrintOpenEntries](#)

[InvoiceFromBP.SkipWriteOffPaymentDifference](#)

InvoiceFromBP.SkipPrintAgingAnalysis

Skips Invoice-from Business Partner when Printing Aging Analysis.

To implement this process extension, you can use the information below:

Usage:	Process Extension InvoiceFromBP.SkipPrintAgingAnalysis can be used to skip certain Invoice-from Business Partner when Printing Aging Analysis from Print Invoice-from Business Partner Aging Summary. Sessions where this Process Extension can be implemented: - Print Invoice-from Business Partner Aging Summary. (tfacp3425m000) - Print Invoice-from Business Partner Aging Analysis. (tfacp2420m000) Fields that are available to be used in this Process Extension: - Primary key fields of tccoml22: tccoml22.ifbp (Invoice-from Business Partner) tccoml22.cofc (Department) Pseudocode: Below you can find an example: Hook: Declarations <pre> table ttccoml22 Hook: ext.skip function extern boolean ext.skip() { /* read data using tccoml22.ifbp, tccoml22.cofc if <condition on the data read = true> then return(true) endif return (false) } </pre>

InvoiceFromBP.SkipPrintOpenEntries

Skips Invoice-from Business Partner when Printing Open Entries.

To implement this process extension, you can use the information below:

Usage:	Process Extension InvoiceFromBP.SkipPrintOpenEntries can be used to skip certain Invoice-from Business Partner when Printing Open Entries from Print Invoice-From Business Partner Open Entries. Sessions where this Process Extension can be implemented: - Print Invoice-From Business Partner Open Entries. (tfacp2421m000) Fields that are available to be used in this Process Extension: - All fields of table tccoml22 are available. Pseudocode: Below you can find an example: Hook: Declarations

	<pre> table ttccom122 Hook: ext.skip function extern boolean ext.skip() { if <condition on the data read = true> then return(true) endif return (false) } </pre>

InvoiceFromBP.SkipWriteOffPaymentDifference

Skips Invoice-from Business Partner when Writing Off Payment Difference.

To implement this process extension, you can use the information below:

Usage:	InvoiceFromBP.SkipWriteOffPaymentDifference can be used to skip Invoice-from Business Partner when Writing Off Payment Difference from Write Off Payment Difference (tfacp2230m000). Fields that are available to be used in this Process Extension: - All fields of table tfacp500 are available. Pseudocode: Below you can find an example: Hook: Declarations <pre> table ttfacp500 Hook: ext.skip function extern boolean ext.skip() { if <condition on the data read = true> then return(true) endif return (false) } </pre>

Process Extensions for InvoiceToBP

The following process extension(s) is/are available:

To implement this process extension, you can use the information below:

To implement this process extension, you can use the information below:

Infor LN Public Interfaces & Process Extensions Reference Guide (On-premises) | 1247

	- Print Invoice-to Business Partner Open Entries. (tfacr2421m000) Fields that are available to be used in this Process Extension: - All fields of table tccom112 are available. Pseudocode: Below you can find an example: Hook: Declarations <pre> table ttccom112 </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { if <condition on the data read = true> then return(true) endif return (false) } </pre>

Process Extensions for ItemMasterPlan

The following process extension(s) is/are available:

[ItemMasterPlan.SkipCopyToScenario](#)

[ItemMasterPlan.SkipCopyToScenarioForPeriod](#)

ItemMasterPlan.SkipCopyToScenario

Skips copying Item Master Plan to Scenario.

To implement this process extension, you can use the information below:

Usage:	Process Extension ItemMasterPlan.SkipCopyToScenario can be used to skip certain plan items when copying Plan Item Master Plan data from one scenario to another. Sessions where this Process Extension can be implemented: - Copy Master Plan to Scenario (cprmp2203m000) Fields that are available to be used in this Process Extension: - cprpd100.plni (Plan Item), - cprpd100.plvl (Plan Level) External variables that are available to be used in this Process Extension:

	<pre>- proc_ext_skip_item_masterplan_cpy_scen_from [type: domain cpcom.plnc] - proc_ext_skip_item_masterplan_cpy_scen_to [type: domain cpcom.plnc]</pre>

ItemMasterPlan.SkipCopyToScenarioForPeriod

Skips copying Item Master Plan to Scenario for specific Period.

This process extension is available from [KB2274036](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension ItemMasterPlan.SkipCopyToScenarioForPeriod can be used to skip copying Master Plan to Scenario per plan Period in the from Scenario. Sessions where this Process Extension can be implemented: - Copy Master Plan to Scenario (cprmp2203m000) Fields that are available to be used in this Process Extension: - cprpd100.plni (Plan Item), - cprpd100.plvl (Plan Level) - cprmp300.chan (Channel) - cprmp300.plnc (Scenario From), - cprmp300.pern (Period) - cprmp300.sdat (Period Start Date), - cprmp300.pdat (Period Finish Date) External variables that are available to be used in this Process Extension: <pre>- proc_ext_skip_item_masterplan_cpy_period_scen_to [type: domain cpcom.plnc]</pre>

Process Extensions for LoadBuilding

The following process extension(s) is/are available:

[LoadBuilding.CustomBuildingConstraints](#)

LoadBuilding.CustomBuildingConstraints

Allows to perform additional load building constraints during load building.

Technical information for this process extension:

Usage:	With this Process Extension it is possible to perform additional checks on the load building process. During the load building, the process will search for existing loads to add newly picked inventory to be shipped with. There are however load building constraints which must force Infor LN to not combine the picked inventory with a found load. This process extension allows a customer to define custom load building criteria to match the business case which is required.

To implement this process extension, you need to implement the following method(s):

whext.dll0004.custom.load.building.constraints

Syntax:	<pre>long whext.dll0004.custom.load.building.constraints(domain whinh.load i.found.load, domain whinh.shpm i.found.shipment, domain whinh.oorg i.order.origin, domain tcorno i.order.number, domain tcpono i.order.line, domain tcpono i.order.sequence, ref boolean o.use.selected.load)</pre>
Usage:	Expl: This process extension allows for customer specific load building constraints. This process extension is called from the Infor LN standard, when one of the following events are triggered: - Create Projected Shipment - Confirm Pick (when Outbound Inspections are not in use) - Approve (when Outbound Inspections are in use) - Cross Docking (when Confirmation of Receipt or Putaway of Inbound Advice is done and the goods are to be Cross-Docked or when Approve is done for the Inbound Inspection where Cross-Docking should be done in non location controlled warehouses.) - Create Outbound Order Line for Cost/Service items which are released to warehousing. Note: This process extension is not called when composing a load structure. The constraints that are applicable for the load building should also be added as extension points for composing loads, via dedicated table / field extensions on the related field, in line with the load building criteria. The load building process will try to search for loads which are already present to perform the shipment of the newly staged inventory to be shipped. Conditions apply on the combinations which are allowed based on order type specific load building criteria. This process extension allows to define custom building criteria to be validated prior to selecting the load for a newly created shipment and shipment line. When a shipment is found, which is suitable to perform the shipping of the outbound order line, it can be that the load will not allow to add a new shipment line, due to customer defined constraint. The i.found.shipment is filled only when there is also a shipment found in the shipment building process, otherwise the i.found.shipment will be empty.

	Table "Loads" (whinh440) is current, for the i.found.load. Other data that is required to determine if the load can be used can be retrieved by making use of the outbound order line key, which is also part of the input of this process extension. Setting the o.use.selected.load to false will take care that the i.found.load will not be used because a custom criteria is not matching with the load and the new inventory moved to the staging location. Within Freight Management there is also a process extension called "FreightLoadBuilding.CustomBuildingConstraints" which will be triggered during the generation of a plan. If freight is implemented and used for planning purposes, it is advisable to apply the same customer specific building criteria in freight management, as otherwise the planning will be overruled by warehousing. Note: If you execute queries on the standard Infor LN tables, bind the table fields to local variables to prevent disturbing the standard flow. Pre: N.a. Post: N.a. Input: i.found.load - Load that is found during load building i.found.shipment - Shipment that is found during shipment building Outbound Order Line Key i.order.origin i.order.number i.order.line i.order.sequence Output: o.use.selected.load - Indicator if the found load should be used (true/false) Please also set an error message regarding this shipment building constraint, so the users will get feedback when required. Return: 0/DALHOOKERROR

Process Extensions for LocationSearchEngine

The following process extension(s) is/are available:

[LocationSearchEngine.CheckLocation](#)

LocationSearchEngine.CheckLocation

Allows to skip or replace a location advised by standard LN search engine.

This process extension is available from [KB3563599](#).

Technical information for this process extension:

Usage:	With this Process Extension it is possible to perform additional checks on the Location Search Engine, which is used when getting available location to store the goods. There are 2 extension points within the process extension, which can be implemented individually: The first one (whext.dll0015.override.location) will be called from the standard prior to location constraints checks (multi-item (lot), capacity and other properties). This allows the extender to replace the suggested warehouse location with any other suitable location of their choice during the execution of the Location Search Engine. If the location has failed the standard storage condition check, the selection process will continue until a suitable location is found or will stop if no available locations are detected. The other extension point (whext.dll0015.skip.location) will be called when the Standard LN Location Search Engine has selected the suitable location and performed standard location availability checks. Here the user will be able to skip the current location.

To implement this process extension, you need to implement the following method(s):

whext.dll0015.override.location

Syntax:	<pre>long whext.dll0015.override.location(domain tccwar i.warehouse, domain whloca i.location, domain tcitem i.item, domain tcqstl i.required.quantity, domain tccuni i.required.unit, ref domain tcitem i.multi.item.array() fixed, ref domain tcqstl i.multi.item.required.quantity.array(), ref domain tccuni i.multi.item.required.unit.array() fixed, domain tcmcs.long i.number.of.units, domain tcclot i.lot, domain tccom.bpid i.owner, domain tcwght i.required.weight, domain tcleng i.required.volume, domain tcleng i.required.floor, domain tcleng i.required.depth, domain tcleng i.required.width, domain tcleng i.required.height, domain whwmd.pkdf i.package.definition, domain tcqivl i.quantity.package.definition.received, domain tcqivl i.quantity.package.definition.advised, domain whinh.ittp i.transaction.type, boolean i.assign.location.to.business.partner, boolean i.all.on.one.location, boolean i.must.be.multi.item, boolean i.must.be.multi.lot, domain whwmd.loct i.location.type, boolean i.minimal.put.away,</pre>

	ref	domain	whloca	o.new.location)
Usage:	Expl:	This function allows a user to specify a new location to be used instead of the location suggested by Standard LN Search Engine. This is particularly useful in situations where it is necessary to change the storage location for goods or materials. The function provides flexibility and adaptability, enabling the user to easily manage changes in planning and logistics. After executing the function, warehouse location constraints will be checked according to LN standards. If the replaced location does not match the conditions for storing the goods, the function will proceed to search for the next location. If at each iteration, as a result of executing the extension, the same location (or a location that does not fit the placement conditions) is obtained, then a situation may arise that the LN won't be able to successfully complete the location search. In case of generating Inbound Advice with a selected option 'Detailed Inbound Advice Log', a message about the location replacement will be added to this Log. IMPORTANT: it is not possible to use the Public Interface 'InboundAdvice.GetAvailableLocation' inside this process extension, as this would cause a recursion error. As an example of the extension implementation, can be considered creation of a manual Inbound Advice where the user can specify a location of their own choice. However, after verification by the LN standard, the specified location may not meet the storage conditions and the advice cannot be completed.		
	Pre:	NA		
	Post:	Location found by LN Standard may be replaced by a user-defined location if it meets the storage conditions		
	Input:	- i.warehouse - Warehouse - i.location - Found Location - i.item - Item - i.required.quantity - Quantity that must be advised - i.required.unit - Unit of i.required.qty - i.multi.item.array() - The items in the handling unit structure - i.multi.item.required.quantity.array() - The quantities in the handling unit structure - i.multi.item.required.unit.array() - The units in the handling unit structure - i.number.of.units - The number of handling units in the handling unit structure (excluding dummy handling units) - i.lot - Lot code - i.owner - The business partner allocated to the location - i.required.weight - Required weight - i.required.volume - Required volume - i.required.floor - Required floor in location units. Where floor = length unit*length unit - i.required.depth - Required depth - i.required.width - Required width - i.required.height - Required height - i.package.definition - Package definition - i.quantity.package.definition.received - Only applicable in case of package		

	<pre> definitions. i.quantity.package.definition.advised - Only applicable in case of package definitions. i.transaction.type - Transaction type (receipt, issue, etc) i.assign.location.to.business.partner - true: if no location can be found for the given owner then an empty, unassigned location is searched for. If found, then this location is assigned to the owner and advise is done to this location. false: no location will be assigned to an owner. i.all.on.one.location - true: advise must be done to one location. If not possible, then no advise will be generated. false: advise can be done to several locations. i.must.be.multi.item - true: location must be multi item false:location may also be single item Note: a multi item HU can not be placed on a single item location. i.must.be.multi.lot - true: location must be a multi lot location false: location may be single lot as well i.location.type - Location type to search for i.minimal.put.away - Minimal put away Output: o.new.location - Location that might replace Found Location. Initially initialized by i.location Return: 0/DALHOOKERROR </pre>

whext.dll0015.skip.location

Syntax:	<pre> long whext.dll0015.skip.location(domain tccwar i.warehouse, domain whloca i.location, domain tcitem i.item, domain tcqstl i.required.quantity, domain tccuni i.required.unit, ref boolean o.skip.location) </pre>
Usage:	Expl: This function allows skipping the warehouse location suggested by the Standard LN Search Engine based on user-defined criteria. The user can set specific conditions or parameters that will be used to filter the suggested locations. If the proposed location does not match these criteria, the function will automatically skip it and move to the next possible location. This ensures a more accurate and personalized location selection process that meets the user's specific requirements. In case of generating Inbound Advice with a selected option 'Detailed Inbound Advice Log', a message about the skipped location will be added to this Log. IMPORTANT: it is not possible to use the Public Interface

	'InboundAdvice.GetAvailableLocation' inside this process extension, as this would cause a recursion error. Following tables are current when this extension is triggered: whwmd300 - Locations Pre: NA Post: NA Input: i.warehouse - Warehouse i.location - Found Location i.item - Item i.required.quantity - Required Quantity i.required.unit - Required Unit Output: o.skip.location - location must be skipped (true/false) Return: 0/DALHOOKERROR

Process Extensions for OutboundAdvice

The following process extension(s) is/are available:

[OutboundAdvice.SkipGeneratePickingList](#)

[OutboundAdvice.SkipPrint](#)

[OutboundAdvice.SkipReleaseAdvice](#)

OutboundAdvice.SkipGeneratePickingList

Skips Outbound Advice Line during Generate Picking List.

To implement this process extension, you can use the information below:

Usage:	Process Extension OutboundAdvice.SkipGeneratePickingList can be used to skip outbound advice lines during the generate picking list process. For all instances where outbound advices are put on a picking list this process extension will be invoked, this also includes automatic outbound processing. Fields that are available to be used in this Process Extension: - All fields of tables: - Outbound Advice (whinh225) - Outbound Order Lines (whinh220)

OutboundAdvice.SkipPrint

Skips Printing of Outbound Advice.

To implement this process extension, you can use the information below:

Usage:	Process Extension OutboundAdvice.SkipPrint can be used to skip the printing of Outbound Advice. Sessions where this Process Extension can be implemented: - Print Outbound Advice (whinh4460m000) Fields that are available to be used in this Process Extension: - All fields of tables: - Outbound Advice (whinh225) - Outbound Order Lines (whinh220)

OutboundAdvice.SkipReleaseAdvice

Skips Outbound Advice Line during Release Outbound Advice.

To implement this process extension, you can use the information below:

Usage:	Process Extension OutboundAdvice.SkipReleaseAdvice can be used to skip outbound advice lines during the release outbound advice process. For all instances where outbound advices are released this process extension will be invoked, this also includes automatic outbound processing. Fields that are available to be used in this Process Extension: - All fields of tables: - Outbound Advice (whinh225) - Outbound Order Lines (whinh220)

Process Extensions for OutboundOrderLine

The following process extension(s) is/are available:

[OutboundOrderLine.SkipGenerateAdvice](#)

OutboundOrderLine.SkipGenerateAdvice

Skips Outbound Order Lines during Generate Outbound Advice.

To implement this process extension, you can use the information below:

Usage:	Process Extension OutboundOrderLine.SkipGenerateAdvice can be used to skip outbound order lines during the generate outbound advice process. For all instances where outbound advices are generated this process extension will be invoked, this also includes automatic outbound processing. Fields that are available to be used in this Process Extension: - All fields of tables: - Outbound Order Lines (whinh220)

Process Extensions for PackingList

The following process extension(s) is/are available:

[PackingList.SkipPrint](#)

PackingList.SkipPrint

Skips Printing of Packing List.

This process extension is available from [KB2307865](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension PackingList.SkipPrint can be used to skip the printing of a Packing List. Sessions where this Process Extension can be implemented: - Print Packing List (whinh4476m000) Fields that are available to be used in this Process Extension: - All fields of tables: - Shipments (whinh430) - Shipment Lines (whinh431)

Process Extensions for PackingSlip

The following process extension(s) is/are available:

[PackingSlip.SkipPrint](#)

PackingSlip.SkipPrint

Skips printing of Packing Slips.

This process extension is available from [KB3569968](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension PackingSlip.SkipPrintShipmentLine can be used to skip the printing of Shipment Lines on a Packing Slip. This process extension will be invoked when printing a Packing Slip, either via session Print Packing Slips (whinh4475m000) or via the automatic printing of the shipping documents. Fields that are available to be used in this Process Extension: - All fields of tables: - Shipments (whinh430) - Shipment Lines (whinh431)

Process Extensions for PaymentAdvice

The following process extension(s) is/are available:

[PaymentAdvice.CustomAuditPayments](#)

PaymentAdvice.CustomAuditPayments

Define Custom Auditing and set error messages if present.

This process extension is available from [KB2308085](#).

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to define Custom Auditing by customer/extender and set an error messages if required.

To implement this process extension, you need to implement the following method(s):

tfext.cmg0002.custom.audit.composed.payment

Syntax:	<pre>long tfext.cmg0002.custom.audit.composed.payment(ref boolean o.custom.audit.errors.present)</pre>
Usage:	Expl: Use this method to define Custom specific auditing and return errors if present. Fields that are available to be used in this Process Extension: - Below fields present in the select of tfcmg103, are current. - tfcmg103._index1 - Fields of Index 1 - tfcmg103.amth - Amount in Home Currency - tfcmg103.amnt - Transaction Amount - tfcmg103.bank - Bank - tfcmg103.ifbp - Invoice-from Business Partner - tfcmg103.icmp - Invoice Company Default value of 'o.custom.audit.errors.present' should be set to False. If audit errors are found, these must be set via <code>dal.set.error.message()</code> and the 'o.custom.audit.errors.present' must be set to true. Pre: N.A. Post: N.A. Input: N.A. Output: o.custom.audit.errors.present - Indicates whether errors are present in the custom logic of auditing the composed payment. Return: 0 - Success - DALHOOKERROR - When an error occurs during custom auditing.

tfext.cmg0002.custom.audit.payment.advice.line

Syntax:	<pre>long tfext.cmg0002.custom.audit.payment.advice.line(ref boolean o.custom.audit.errors.present)</pre>
Usage:	Expl: Use this method to define Custom specific auditing and return errors if present. Fields that are available to be used in this Process Extension: - All fields from tfcmg101 are current (we are processing a particular tfcmg101 line related to the current composed payment line). Default value of 'o.custom.audit.errors.present' should be set to False. If audit errors are found, these must be set via

	<pre> dal.set.error.message() and the 'o.custom.audit.errors.present' must be set to true. Pre: preconditions Post: postconditions Pre: N.A. Post: N.A. Input: N.A. Output: o.custom.audit.errors.present - Indicates whether errors are present in the custom logic of auditing the payment advice line. Return: 0 - Success DALHOOKERROR - When an error occurs during custom auditing. </pre>

Process Extensions for PaymentReceipt

The following process extension(s) is/are available:

[PaymentReceipt.CustomXMLHandling](#)

PaymentReceipt.CustomXMLHandling

Define custom elements for XML Payments/Receipts used at creating the XML file.

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to define extra custom elements to be used in the XML Payment/Receipt Layout.

To implement this process extension, you need to implement the following method(s):

tfext.cmg0001.define.custom.payment.elements

Syntax:	<pre> long tfext.cmg0001.define.custom.payment.elements(ref long o.number.of.elements, ref domain tcmcs.st12 o.custom.element.codes() fixed, ref domain tcmcs.str132m o.custom.element.descriptions() fixed mb, ref domain tfcmg.xttp o.custom.element.data.types()) </pre>
Usage:	Expl: Use this method to define extra custom defined elements to be used in the XML Payment layout and file. These custom element codes must start with "8" or "9"; other defined element codes

will be ignored.

Mapping a custom element to a Remittance Related XML Tag, only the custom elements starting with "8" can be used. The custom elements starting with "9" can only be used for mapping to non-remittance related XML Tags.

The custom defined elements can then be used for mapping to an XML attribute in session XML Payment/Receipt Layout Lines (tfcmg0125m000). If a custom defined element is mapped, during building the XML file, a value must be retrieved for the element. For that, the following method must be used:

```
tftext.cmg0001.get.value.for.custom.payment.element
```

Start of Example of Implementation

```
long    length.element.code
long    length.element.description
long    elem.nr
```

```
long    dummy.long
```

```
dummy.long = rdi.domain.string(
                                domainof(o.custom.element.codes),
                                length.element.code,
                                dummy.long)
dummy.long = rdi.domain.string(
                                domainof(o.custom.element.descriptions),
                                length.element.description,
                                dummy.long)
```

```
elem.nr = 0
```

```
|*****
|* We will add 2 custom elements.
|*****
o.number.of.elements = 2
```

```
if alloc.mem( o.custom.element.codes,
              length.element.code,
              o.number.of.elements) +
  alloc.mem( o.custom.element.descriptions,
              length.element.description,
              o.number.of.elements) +
  alloc.mem( o.custom.element.data.types,
              o.number.of.elements) <> 0 then
  return (DALHOOKERROR)
endif
```

```
|*****
|* Add the element Name for the Pay-to BP
|*****
elem.nr = elem.nr + 1
o.custom.element.codes(1, elem.nr) = "910210000000"
o.custom.element.descriptions(1, elem.nr) =
  "Composed Payments/Pay-to Business Partner/Name"
o.custom.element.data.types(elem.nr) = tfcmg.xttp.string

|*****
|* Add the element Transaction Amount Positive
```

	<pre> ***** elem.nr = elem.nr + 1 o.custom.element.codes(1, elem.nr) = "911200000000" o.custom.element.descriptions(1, elem.nr) = "Composed Payments/Transaction Amount Positive" o.custom.element.data.types(elem.nr) = tfcmg.xttp.decimal return(0) End of Example of Implementation ----- In above example, a custom element is added to influence the transaction amount in the file (i.e. making the transaction amount positive); note that this impacts the control amount in the file and thus a custom element is needed as well to determine the control amount in the extension. Pre: N.A. Post: N.A. Input: N.A. Output: o.number.of.elements - Number of additional custom payment elements o.custom.element.codes - Array with element codes. The element code must start with a "8" or "9". Maximum of 12 characters. E.g. "910100000000". o.custom.element.descriptions - Array with descriptions of the element codes. Maximum of 132 characters. o.custom.element.data.types - Array with data types of the element code. Data type should be one of the following values: tfcmg.xttp.string (String) tfcmg.xttp.date (Date) tfcmg.xttp.utc (UTC Date) tfcmg.xttp.decimal (Decimal Number) tfcmg.xttp.bool (Boolean) Return: 0 - Success DALHOOKERROR - When an error occurs in defining the custom elements. </pre>

tfext.cm0001.define.custom.receipt.elements

Syntax:	<pre> long tfext.cm0001.define.custom.receipt.elements(ref long o.number.of.elements, ref domain tcmcs.st12 o.custom.element.codes() fixed, ref domain tcmcs.str132m o.custom.element.descriptions() fixed mb, ref domain tfcmg.xttp o.custom.element.data.types()) </pre>
Usage:	Expl: Use this method to define extra custom defined elements to be used in the XML Receipt layout and file. These custom element codes must start with "8" or "9"; other defined element codes

	will be ignored. Mapping a custom element to a Remittance Related XML Tag, only the custom elements starting with "8" can be used. The custom elements starting with "9" can only be used for mapping to non-remittance related XML Tags. The custom defined elements can then be used for mapping to an XML attribute in session XML Payment/Receipt Layout Lines (tfcmg0125m000). If a custom defined element is mapped, during building the XML file, a value must be retrieved for the element. For that, the following method must be used: <pre>tftext.cmg0001.get.value.for.custom.receipt.element</pre> An example implementation is given in the method <code>tftext.cmg0001.define.custom.payment.elements</code>, which is likely same. Pre: N.A. Post: N.A. Input: N.A. Output: o.number.of.elements - Number of additional custom receipt elements o.custom.element.codes - Array with element codes. The element code must start with a "8" or "9". Maximum of 12 characters. E.g. "910900000000". o.custom.element.descriptions - Array with descriptions of the element codes. Maximum of 132 characters. o.custom.element.data.types - Array with data types of the element code. Data type should be one of the following values: tfcmg.xttp.string (String) tfcmg.xttp.date (Date) tfcmg.xttp.utc (UTC Date) tfcmg.xttp.decimal (Decimal Number) tfcmg.xttp.bool (Boolean) Return: 0 - Success DALHOOKERROR - When an error occurs in defining the custom elements.

tftext.cmg0001.get.value.for.custom.payment.element

Syntax:	<pre>domain tcmcs.s999m tftext.cmg0001.get.value.for.custom.payment.element (domain tcncmp i.payment.company, domain tcmcs.st12 i.custom.element.code, domain tccwoc i.accounting.office)</pre>
Usage:	Expl: Use this method to get a value for the defined custom payment element code during the creation of the Payment XML File.

	The custom elements must be defined with the following method: <pre>tftext.cmg0001.define.custom.payment.elements</pre> Following tables are current when this extension is triggered (during creation of the XML Bank File): <table> <tr><td>tccom000</td><td>Implemented Software Components (Companies)</td></tr> <tr><td>tccom100</td><td>Business Partners</td></tr> <tr><td>tccom115</td><td>Bank Accounts by Pay-by Business Partner</td></tr> <tr><td>tccom125</td><td>Bank Accounts by Pay-to Business Partner</td></tr> <tr><td>tccom130</td><td>Addresses</td></tr> <tr><td>tccom139</td><td>Cities by Country</td></tr> <tr><td>tcms002</td><td>Currencies</td></tr> <tr><td>tcms010</td><td>Countries</td></tr> <tr><td>tcms029</td><td>Business Partner Types</td></tr> <tr><td>tcms046</td><td>Languages</td></tr> <tr><td>tcms143</td><td>States/Provinces</td></tr> </table> <table> <tr><td>tfcmg001</td><td>Bank Relations</td></tr> <tr><td>tfcmg003</td><td>Payment/Receipt Methods</td></tr> <tr><td>tfcmg009</td><td>Financial Institutions</td></tr> <tr><td>tfcmg011</td><td>Bank Branches</td></tr> </table> <table> <tr><td>tfcmg103</td><td>Composed Payments</td></tr> <tr><td>tfcmg002</td><td>Reasons for Payment</td></tr> <tr><td>tfcmg025</td><td>Payment/Receipt XML Layout Details</td></tr> </table> The below table is current only for Remittance Related elements: <table> <tr><td>tfcmg101</td><td>Payment/Trade Notes Payable Advice</td></tr> </table> ----- Start of Example of Implementation <pre> domain tcnama name.of.bp domain tfgld.amnt composed.amount name.of.bp = "" composed.amount = 0.0 on case i.custom.element.code case "910210000000": /* Name of Composed Pay-to BP get.var(pid, "tccom100.nama", name.of.bp) return(name.of.bp) case "911200000000": /* Transaction Amount Composed Payment (Positive) /* Return always a POSITIVE value of the Amount field get.var(pid, "tfcmg103.amnt", composed.amount) return(trim\$(sprintf\$("%@ZZZZZZZZZZZZZZ9V.99@" , abs(composed.amount)))) default: break endcase return("") End of Example of Implementation ----- </pre> Pre: N.A.	tccom000	Implemented Software Components (Companies)	tccom100	Business Partners	tccom115	Bank Accounts by Pay-by Business Partner	tccom125	Bank Accounts by Pay-to Business Partner	tccom130	Addresses	tccom139	Cities by Country	tcms002	Currencies	tcms010	Countries	tcms029	Business Partner Types	tcms046	Languages	tcms143	States/Provinces	tfcmg001	Bank Relations	tfcmg003	Payment/Receipt Methods	tfcmg009	Financial Institutions	tfcmg011	Bank Branches	tfcmg103	Composed Payments	tfcmg002	Reasons for Payment	tfcmg025	Payment/Receipt XML Layout Details	tfcmg101	Payment/Trade Notes Payable Advice
tccom000	Implemented Software Components (Companies)																																						
tccom100	Business Partners																																						
tccom115	Bank Accounts by Pay-by Business Partner																																						
tccom125	Bank Accounts by Pay-to Business Partner																																						
tccom130	Addresses																																						
tccom139	Cities by Country																																						
tcms002	Currencies																																						
tcms010	Countries																																						
tcms029	Business Partner Types																																						
tcms046	Languages																																						
tcms143	States/Provinces																																						
tfcmg001	Bank Relations																																						
tfcmg003	Payment/Receipt Methods																																						
tfcmg009	Financial Institutions																																						
tfcmg011	Bank Branches																																						
tfcmg103	Composed Payments																																						
tfcmg002	Reasons for Payment																																						
tfcmg025	Payment/Receipt XML Layout Details																																						
tfcmg101	Payment/Trade Notes Payable Advice																																						

	Post: N.A. Input: i.payment.company - Payment Company i.custom.element.code - The custom element code mapped to an XML attribute. i.accounting.office - Accounting Office. Output: N.A. Return: String with the value of the custom payment element.

tfext.cmg0001.get.value.for.custom.receipt.element

Syntax:	<pre>domain tcmcs.s999m tfext.cmg0001.get.value.for.custom.receipt.element(domain tcncmp i.receipt.company, domain tcmcs.st12 i.custom.element.code, domain tccwoc i.accounting.office)</pre>																																						
Usage:	Expl: Use this method to get a value for the defined custom receipt element code during the creation of the Direct Debit XML File. The custom elements must be defined with the following method: <pre>tfext.cmg0001.define.custom.receipt.elements</pre> Following tables are current when this extension is triggered (during creation of the XML Bank File): <table> <tr><td>tccom000</td><td>Implemented Software Components (Companies)</td></tr> <tr><td>tccom100</td><td>Business Partners</td></tr> <tr><td>tccom115</td><td>Bank Accounts by Pay-by Business Partner</td></tr> <tr><td>tccom125</td><td>Bank Accounts by Pay-to Business Partner</td></tr> <tr><td>tccom130</td><td>Addresses</td></tr> <tr><td>tccom139</td><td>Cities by Country</td></tr> <tr><td>tcmcs002</td><td>Currencies</td></tr> <tr><td>tcmcs010</td><td>Countries</td></tr> <tr><td>tcmcs029</td><td>Business Partner Types</td></tr> <tr><td>tcmcs046</td><td>Languages</td></tr> <tr><td>tcmcs143</td><td>States/Provinces</td></tr> <tr><td>tfcmg001</td><td>Bank Relations</td></tr> <tr><td>tfcmg003</td><td>Payment/Receipt Methods</td></tr> <tr><td>tfcmg009</td><td>Financial Institutions</td></tr> <tr><td>tfcmg011</td><td>Bank Branches</td></tr> <tr><td>tfcmg027</td><td>Direct Debit Mandates</td></tr> <tr><td>tfcmg403</td><td>Composed Direct Debits</td></tr> <tr><td>tfcmg025</td><td>Payment/Receipt XML Layout Details</td></tr> </table> The below table is current only for Remittance Related elements: <table> <tr><td>tfcmg401</td><td>Direct Debit Advice</td></tr> </table> An example implementation is given in the method <code>tfext.cmg0001.get.value.for.custom.payment.element</code>, which is likely same. Pre: N.A. Post: N.A. Input: i.receipt.company - Receipt Company i.custom.element.code - The custom element code mapped to an XML attribute.	tccom000	Implemented Software Components (Companies)	tccom100	Business Partners	tccom115	Bank Accounts by Pay-by Business Partner	tccom125	Bank Accounts by Pay-to Business Partner	tccom130	Addresses	tccom139	Cities by Country	tcmcs002	Currencies	tcmcs010	Countries	tcmcs029	Business Partner Types	tcmcs046	Languages	tcmcs143	States/Provinces	tfcmg001	Bank Relations	tfcmg003	Payment/Receipt Methods	tfcmg009	Financial Institutions	tfcmg011	Bank Branches	tfcmg027	Direct Debit Mandates	tfcmg403	Composed Direct Debits	tfcmg025	Payment/Receipt XML Layout Details	tfcmg401	Direct Debit Advice
tccom000	Implemented Software Components (Companies)																																						
tccom100	Business Partners																																						
tccom115	Bank Accounts by Pay-by Business Partner																																						
tccom125	Bank Accounts by Pay-to Business Partner																																						
tccom130	Addresses																																						
tccom139	Cities by Country																																						
tcmcs002	Currencies																																						
tcmcs010	Countries																																						
tcmcs029	Business Partner Types																																						
tcmcs046	Languages																																						
tcmcs143	States/Provinces																																						
tfcmg001	Bank Relations																																						
tfcmg003	Payment/Receipt Methods																																						
tfcmg009	Financial Institutions																																						
tfcmg011	Bank Branches																																						
tfcmg027	Direct Debit Mandates																																						
tfcmg403	Composed Direct Debits																																						
tfcmg025	Payment/Receipt XML Layout Details																																						
tfcmg401	Direct Debit Advice																																						

	<pre>i.accounting.office - Accounting Office. Output: N.A. Return: String with the value of the custom receipt element.</pre>

Process Extensions for PerformInventoryValuation

The following process extension(s) is/are available:

[PerformInventoryValuation.SkipItem](#)

PerformInventoryValuation.SkipItem

Skips Items during Perform Inventory Valuation.

This process extension is available from [KB2298172](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension PerformInventoryValuation.SkipItem can be used to skip Items during Perform Inventory Valuation. Sessions where this Process Extension can be implemented: - Perform Inventory Valuation (whinal210m000) Fields that are available to be used in this Process Extension: - All fields of tables - Items (tcibd001) - Warehouses (tcmcs003) Note: tables must also be declared in the Process Extension. Pseudocode: Below you can find an example. Hook: Declarations <pre>table tcibd001</pre> Hook: ext.skip <pre>function extern boolean ext.skip() { if <condition on tcibd001 = true> then return(true) endif return (false) }</pre>

Process Extensions for PickingMission

The following process extension(s) is/are available:

[PickingMission.HandleAfterGenerate](#)

PickingMission.HandleAfterGenerate

Allows to add functionality after the generation of a picking mission.

Technical information for this process extension:

Usage:	With this Process Extension it is possible execute actions after the creation of a picking mission. This process extension will only be triggered when the picking mission is generated, not when a reprint or preview is done for the picking mission.

To implement this process extension, you need to implement the following method(s):

whext.dll0005.picking.mission.handle.after.generate

Syntax:	<pre>long whext.dll0005.picking.mission.handle.after.generate(domain whinh.btno i.runnumber, domain tccwar i.warehouse, domain whinh.picm i.picking.mission, domain tcncmp i.company, ref boolean o.remove.picking.mission)</pre>
Usage:	Expl: This process extension allows to include customer specific functionality after the process of generating picking missions. This process extension will be called from Infor LN standard when the picking mission is generated via session "Generate Picking List" (whinh4415m000), or when the generation of picking lists is an automatic order step. For each picking mission the process extension is executed after generation. No table fields are current, based on the input arguments the records must be fetched. When the process extension is executed, the picking missions are committed to the database. Any transactions which are required to the database (inserts, updates, deletes) must be accompanied with a new transaction (db.retry.point / abort/commit).

	Picking missions are removed when o.remove.picking.mission is set to True or when this function returns a value <> 0. Pre: N.a. Post: N.a. Input: i.runnumber - The run number for which the picking mission is created i.warehouse - Warehouse i.picking.mission - Picking Mission which is generated i.company - Logistic Company of the outbound advice Output: o.remove.picking.mission - Indicator if the standard should remove the picking mission. Return: 0: Success / <> 0: Error

Process Extensions for PlannedInventoryTransaction

The following process extension(s) is/are available:

[PlannedInventoryTransaction.SkipPrint](#)

PlannedInventoryTransaction.SkipPrint

Skips printing of Planned Inventory Transaction.

This process extension is available from [KB3557494](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension PlannedInventoryTransaction.SkipPrint can be used to skip the printing of Planned Inventory Transactions by Item. Sessions where this Process Extension can be implemented: - Print Planned Inventory Transactions by Item (whinpl400m000) Fields that are available to be used in this Process Extension: - All fields of tables: - Planned Inventory Transactions by Item (whinpl100)

Process Extensions for PlannedOrder

The following process extension(s) is/are available:

[PlannedOrder.SkipConfirmOrder](#)

[PlannedOrder.SkipPrint](#)

[PlannedOrder.SkipTransfer](#)

PlannedOrder.SkipConfirmOrder

Skips planned orders to be confirmed.

This process extension is available from [KB3602370](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension PlannedOrder.SkipConfirmOrder can be used to skip considering the planned orders to be confirmed. Sessions where this Process Extension can be implemented: - Confirm Order Planning (cprrp1200m000) Fields that are available to be used in this Process Extension: - All fields of table: Planned Orders (cprrp100)

PlannedOrder.SkipPrint

Skips printing Planned Orders.

This process extension is available from [KB3553172](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension PlannedOrder.SkipPrint can be used to skip certain Planned Orders when printing a range of Planned Orders. Sessions where this Process Extension can be implemented: - Print Planned Orders (cprrp1410m000) Fields that are available to be used in this Process Extension: - All fields of table: Planned Orders (cprrp100) - All fields of table: Items - Planning (cprpd100) - All fields of table: Scenarios (cprpd400) - All fields of table: Carriers/LSP (tcmcs080) - All fields of table: Projects (tcmcs052)

PlannedOrder.SkipTransfer

Skips transfer of Planned Orders.

This process extension is available from [KB2236258](#).

Technical information for this process extension:

Usage:	Process Extension PlannedOrder.SkipTransfer can be used to skip the transfer of certain Planned Orders when transferring Planned Orders from Enterprise Planning to the execution level. Extender may specify a message, to present information about the skip decision on the report. Sessions where this Process Extension can be implemented: - Transfer Order Planning (cppat1210m000). Fields that are available to be used in this Process Extension: - cprrp100.plnc (Scenario) - cprrp100.type (Planned Order Type) - cprrp100.orno (Planned Order) Note: Tables & messages must also be declared in the Process Extension. Pseudocode: Below you can find an example. Hook: Declarations <pre>table tcprrp100</pre> Hook: ext.skip.with.reason <pre>function extern boolean ext.skip.with.reason(ref string o.reason) { if <condition on cprrp100 = true> then o.reason = "Transfer skipped because ..." return(true) endif return(false) }</pre>

Process Extensions for PlannedServiceActivity

The following process extension(s) is/are available:

[PlannedServiceActivity.SkipSwitchStatus](#)

PlannedServiceActivity.SkipSwitchStatus

Skips Planned Service Activity when Switching its Status.

To implement this process extension, you can use the information below:

Usage:	Process Extension PlannedServiceActivity.SkipSwitchStatus can be used to skip Planned Service Activities when switching the status. Sessions where this Process Extension can be implemented: - Switch Status Maintenance Plan (tsspc2201m000). Fields that are available to be used in this Process Extension: - All fields of table Planned Activities (tsspc200). - If in the selection range of session 'Switch Status Maintenance Plan' (tsspc2201m000) the field which controls the selection of serialized items is either set to 'Selection of Serialized Items', or set to 'Selection of Top Items', then also all fields of table 'Serialized Items' (tscfg200) can be used. - If in the selection range of session 'Switch Status Maintenance Plan' (tsspc2201m000) the field which controls the selection of maintenance scenarios is set to 'Selection of Scenarios', then also all fields of table 'Preventive Maintenance Scenario Lines' (tsspc131) can be used. Note: tables must also be declared in the Process Extension. Pseudocode: Below you can find an example. Hook: Declarations <pre> table ttsspc200 ***** * Fields of tscfg200 can be used if Selection of Serialized * Items is set to 'Selection of Serialized Items' or set to * 'Selection of Top Items'. ***** table ttscfg200 ***** * Fields of tsspc131 can be used if Selection of Maintenance * Scenarios is set to 'Selection of Scenarios'. ***** table ttsspc131 </pre> Hook: ext.skip

	<pre> function extern boolean ext.skip() { if <condition on tsspc200 = true> then return(true) endif if Selection of Serialized Items is set to 'Selection of Serialized Items' or set to 'Selection of Top Items' then if <condition on tscfg200 = true> then return(true) endif endif if Selection of Maintenance Scenarios is set to 'Selection of Scenarios' then if <condition on tsspc131 = true> then return(true) endif endif return (false) } </pre>

Process Extensions for Pricing

The following process extension(s) is/are available:

[Pricing.DefaultPriceDateType](#)

[Pricing.RetrievePriceAndDiscounts](#)

Pricing.DefaultPriceDateType

Determine customer specific default Price Date Type.

This process extension is available from [KB3572014](#).

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to set a customer determined price date type from an extension to a value other than the defaulted value based on LN logic. This price date type (if valid) is then used in all flows where a price date type is defaulted.

To implement this process extension, you need to implement the following method(s):

tdext.pcg0002.get.customer.determined.purchase.price.date.type.parameter

Syntax:	<pre>long tdext.pcg0002.get.customer.determined.purchase.price.date.type.parameter(domain tdpur.pric i.purchase.price.date.type.from.standard.logic, domain tcncmp i.logistic.company, ref domain tdpur.pric o.purchase.price.date.type.from.extension)</pre>
Usage:	Expl: Use this method to get a customer determined purchase price date type from an extension for purchase parameter. This price date type will be used in Procurement. Using this process extension, the purchase price date type from standard logic can be changed to another value. The purchase price date type from extension is only retrieved and used if: - Process Extension Pricing.DefaultPriceDateType (tdpcg.pric.def.pric.date.type) is implemented; - The purchase price date type from extension is valid (and not empty); - No errors are found during executing the extension. Pre: Not applicable Post: Not applicable Input: i.purchase.price.date.type.from.standard.logic - As defaulted under the standard logic i.logistic.company - Logistic company Output: o.purchase.price.date.type.from.extension - The price date type determined by the extension Return: 0 - Success DALHOOKERROR - When an error occurs in the determination of the price date type from extension

tdext.pcg0002.get.customer.determined.purchase.price.date.type.setting

Syntax:	<pre>long tdext.pcg0002.get.customer.determined.purchase.price.date.type.setting(domain tdpur.pric i.purchase.price.date.type.from.standard.logic, domain tcncmp i.logistic.company, domain tccwoc i.purchase.office, ref domain tdpur.pric o.purchase.price.date.type.from.extension)</pre>
Usage:	Expl: Use this method to get a customer determined purchase price date type from an extension for purchase office setting. This price date type will be used in Procurement. Using this process extension, the purchase price date type from standard logic can be changed to another value.

	Note: The purchase office can be used to determine the price date type. The purchase price date type from extension is only retrieved and used if: - Process Extension Pricing.DefaultPriceDateType (tdpcg.pric.def.pric.date.type) is implemented; - The purchase price date type from extension is valid (and not empty); - No errors are found during executing the extension. Pre: Not applicable Post: Not applicable Input: i.purchase.price.date.type.from.standard.logic - As defaulted under the standard logic i.logistic.company - Logistic company i.purchase.office - Purchase office Output: o.purchase.price.date.type.from.extension - The price date type determined by the extension Return: 0 - Success DALHOOKERROR - When an error occurs in the determination of the price date type from extension

tdext.pcg0002.get.customer.determined.sales.price.date.type.parameter

Syntax:	<pre>long tdext.pcg0002.get.customer.determined.sales.price.date.type.parameter(domain tdpur.pric i.sales.price.date.type.from.standard.logic, domain tcncmp i.logistic.company, ref domain tdpur.pric o.sales.price.date.type.from.extension)</pre>
Usage:	Expl: Use this method to get a customer determined sales price date type from an extension for sales parameter. This price date type will be used in Sales. Using this process extension, the sales price date type from standard logic can be changed to another value. The sales price date type from extension is only retrieved and used if: - Process Extension Pricing.DefaultPriceDateType (tdpcg.pric.def.pric.date.type) is implemented; - The sales price date type from extension is valid (and not empty); - No errors are found during executing the extension. Pre: Not applicable Post: Not applicable Input: i.sales.price.date.type.from.standard.logic - As defaulted under the standard logic i.logistic.company - Logistic company Output: o.sales.price.date.type.from.extension - The price date type determined by the extension Return: 0 - Success DALHOOKERROR - When an error occurs in the determination of the price date type from extension

Syntax:	<pre>long tdxtext.pcg0002.get.customer.determined.sales.price.date.type.setting(domain tdpur.pric i.sales.price.date.type.from.standard.logic, domain tcncmp i.logistic.company, domain tccwoc i.sales.office, ref domain tdpur.pric o.sales.price.date.type.from.extension)</pre>
Usage:	Expl: Use this method to get a customer determined sales price date type from an extension for sales office setting. This price date type will be used in Sales. Using this process extension, the sales price date type from standard logic can be changed to another value. Note: The sales office can be used to determine the price date type. The sales price date type from extension is only retrieved and used if: - Process Extension Pricing.DefaultPriceDateType (tdpcg.pric.def.pric.date.type) is implemented; - The sales price date type from extension is valid (and not empty); - No errors are found during executing the extension. Pre: Not applicable Post: Not applicable Input: i.sales.price.date.type.from.standard.logic - As defaulted under the standard logic i.logistic.company - Logistic company i.sales.office - Sales office Output: o.sales.price.date.type.from.extension - The price date type determined by the extension the extension Return: 0 - Success DALHOOKERROR - When an error occurs in determination of the price date type from extension

Syntax:	<pre> long tdxtext.pcg0002.get.customer.determined.service.price.date.type.parameter(domain tdpur.pric i.service.price.date.type.from.standard.logic, domain tcncmp i.logistic.company, ref domain tdpur.pric o.service.price.date.type.from.extension) </pre>			

Usage:	Expl: Use this method to get a customer determined service price date type from an extension for service parameter. This price date type will be used in Service. Using this process extension, the service price date type from standard logic can be changed to another value. The service price date type from extension is only retrieved and used if: - Process Extension Pricing.DefaultPriceDateType (tdpcg.pric.def.pric.date.type) is implemented; - The service price date type from extension is valid (and not empty); - No errors are found during executing the extension. Pre: Not applicable Post: Not applicable Input: i.service.price.date.type.from.standard.logic - As defaulted under the standard logic i.logistic.company - Logistic company Output: o.service.price.date.type.from.extension - The price date type determined by the extension Return: 0 - Success DALHOOKERROR - When an error occurs in the determination of the price date type from extension

tdext.pcg0002.get.customer.determined.service.price.date.type.setting

Syntax:	<pre>long tdext.pcg0002.get.customer.determined.service.price.date.type.setting(domain tdpur.pric i.service.price.date.type.from.standard.logic, domain tcncmp i.logistic.company, domain tccwoc i.service.office, ref domain tdpur.pric o.service.price.date.type.from.extension)</pre>
Usage:	Expl: Use this method to get a customer determined service price date type from an extension for service office setting. This price date type will be used in Service. Using this process extension, the service price date type from standard logic can be changed to another value. Note: The service office can be used to determine the price date type. The service price date type from extension is only retrieved and used if: - Process Extension Pricing.DefaultPriceDateType (tdpcg.pric.def.pric.date.type) is implemented; - The service price date type from extension is valid (and not empty); - No errors are found during executing the extension. Pre: Not applicable Post: Not applicable Input: i.service.price.date.type.from.standard.logic - As defaulted under the standard logic i.logistic.company - Logistic company i.service.office - Service office

	Output: o.service.price.date.type.from.extension determined by Return: 0 DALHOOKERROR the deter- date type	- The price date type the extension - Success - When an error occurs in mination of the price from extension

Pricing.RetrievePriceAndDiscounts

Determine customer specific Price and Discounts.

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to set customer determined price and discounts from an extension to a value other than the defaulted values based on LN logic . This price and discounts are then used in all flows where the price and discounts are retrieved.

To implement this process extension, you need to implement the following method(s):

tdext.pcg0001.get.customer.determined.price.and.discounts

Syntax:	<pre>long tdext.pcg0001.get.customer.determined.price.and.discounts(domain tdpcg.tyor i.type.of.order, domain tcorno i.number, domain tccono i.contract, domain tcpono i.contract.line, domain tccwoc i.contract.office, domain tcpono i.contract.sequence, domain tcitem i.item, domain tccwoc i.office, domain tcsite i.site, domain tcncmp i.logistic.company, domain tcyesno i.subcontracted, domain tcuef.effn i.effectivity.unit, domain tccpva i.product.variant, domain tcolid i.option.list.id, domain tccom.bpid i.buy.sell.bp, domain tccom.bpid i.ship.bp, domain tccom.bpid i.pricing.bp, domain tcmcs.cpcl i.product.class, domain tcmcs.cpln i.product.line, domain tcmpnr i.manufacturer.part.number, domain tcmcs.cmnf i.manufacturer,</pre>

Process Extensions

fixed,		domain	tcdate	i.order.date,	
		domain	tcdate	i.planned.delivery.date,	
		domain	tcqsl1	i.order.quantity,	
		domain	tcqsl1	i.original.order.quantity,	
		domain	tcqsl1	i.price.group.quantity,	
		domain	tccuni	i.quantity.unit,	
		domain	tcconv	i.quantity.unit.conversion.factor,	
			boolean	i.delete.manual.discount,	
			boolean	i.delete.pricebook.discount,	
		domain	tdpcg.prds	i.apply.price.discount.both,	
		domain	tdpcg.prpc	i.price.percentage,	
			boolean	i.update.uef,	
			boolean	i.update.pcf,	
		domain	tcncmp	i.financial.company,	
		domain	tcccur	i.transaction.currency,	
	ref	domain	tcratc	i.rate(),	
	ref	domain	tcratf	i.rate.factor(),	
		domain	tcdate	i.rate.date,	
		domain	tcrtyp	i.rate.type,	
		domain	tdpcg.prit	i.price.type,	
		domain	tcorno	i.production.order,	
		domain	tcorno	i.maintenance.work.order,	
		domain	tcorno	i.service.order,	
		domain	tcmscs.chan	i.channel,	
		domain	tccdec	i.delivery.terms,	
		domain	tdpcg.pror	i.default.price.book,	
		domain	tccuni	i.price.unit.from.standard.logic,	
		domain	tcconv	i.price.unit.conv.factor.from.standard.logic,	
		domain	tcpric	i.price.from.standard.logic,	
		domain	tcperc	i.percentage.from.standard.logic,	
			boolean	i.derived.item.used.from.standard.logic,	
		domain	tdpcg.prbk	i.price.book.from.standard.logic,	
		domain	tcprsg	i.price.stage.from.standard.logic,	
		domain	tdgen.porg	i.price.origin.from.standard.logic,	
		domain	tdpcg.made	i.price.matrix.definition.from.standard.logic,	
		domain	tdpcg.prse	i.price.matrix.sequence.from.standard.logic,	
	const	domain	tdpcg.maty	i.disc.matrix.type.from.standard.logic(),	
	const	domain	tdgen.dorg	i.discount.origin.from.standard.logic(),	
	const	domain	tdpcg.made	i.disc.matrix.definition.from.standard.logic()	
		const	domain	tdpcg.prse	i.disc.matrix.sequence.from.standard.logic(),
		const	domain	tcdisc	i.discount.percentage.from.standard.logic(),
		const	domain	tddiam	i.discount.amount.from.standard.logic(),
		const	domain	tccdsc	i.discount.code.from.standard.logic() fixed,
		const	domain	tddmth	i.discount.method.from.standard.logic(),
		const	domain	tdpcg.dssc	i.discount.schedule.from.standard.logic() fixed,
		domain	tcyesno	i.determining.from.standard.logic,	
		domain	tcyesno	i.eligible.from.standard.logic,	
	ref	domain	tccuni	o.price.unit,	
	ref	domain	tcconv	o.price.unit.conv.factor,	
	ref	domain	tcpric	o.price,	
	ref	domain	tcperc	o.percentage,	
	ref		boolean	o.derived.item.used,	
	ref	domain	tdpcg.prbk	o.price.book,	
	ref	domain	tcprsg	o.price.stage,	
	ref	domain	tdgen.porg	o.price.origin,	
	ref	domain	tdpcg.made	o.price.matrix.definition,	
	ref	domain	tdpcg.prse	o.price.matrix.sequence,	
	ref	domain	tdpcg.maty	o.disc.matrix.type(),	
	ref	domain	tdgen.dorg	o.discount.origin(),	
	ref	domain	tdpcg.made	o.disc.matrix.definition() fixed,	
	ref	domain	tdpcg.prse	o.disc.matrix.sequence(),	

	<pre> ref domain tcdisc o.discount.percentage(), ref domain tddiam o.discount.amount(), ref domain tccdsc o.discount.code() fixed, ref domain tddmth o.discount.method(), ref domain tdpcg.dssc o.discount.schedule() fixed, ref domain tcyesno o.determining, ref domain tcyesno o.eligible, ref boolean o.fallback.to.standard.logic.in.case.of.errors) </pre>
Usage:	Expl: This function gets the customer determined price and discounts from an extension, if applicable. The price and discounts are only retrieved and used if: - Process Extension Pricing.RetrievePriceAndDiscounts (tdpcg.retrieve.price.and.disc) is implemented; - No errors are found during executing the extension. If the extension returns a zero price and/or zero discounts, these will be treated as valid prices and discounts. If that's not desired, then the extension should return an error in those cases. Pre: NA Post: NA Input: <pre> i.type.of.order - The type of order being processed. Possible values: tdpcg.tyor.so: Sales Orders tdpcg.tyor.sq: Quotations tdpcg.tyor.po: Purchase Orders tdpcg.tyor.pq: RFQ's tdpcg.tyor.gs: General Sales tdpcg.tyor.gp: General Purchase tdpcg.tyor.sr: General Service tdpcg.tyor.sr.quote: Service Quote tdpcg.tyor.sr.msc: Maintenance Sales Order tdpcg.tyor.sr.soc: Service Order i.number - The "Order Number". This depends on the Type of Order (i.type.of.order): tdpcg.tyor.so: Sales Order tdpcg.tyor.sq: Quotation Number tdpcg.tyor.po: Purchase Order tdpcg.tyor.pq: RFQ tdpcg.tyor.gs: "" (empty string) tdpcg.tyor.gp: "" (empty string) tdpcg.tyor.sr: "" (empty string) tdpcg.tyor.sr.quote: Service Quote tdpcg.tyor.sr.msc: Maintenance Sales Order tdpcg.tyor.sr.soc: Service Order i.contract The contract that is linked to the purchase or sales order line i.contract.line The contract line that is linked to the purchase or sales order line i.contract.office The contract office i.contract.sequence The contract sequence to which the purchase order line is linked i.item Item i.office The office used for e.g. reading Office specific settings i.site Site i.logistic company Logistic Company i.subcontracted Subcontracted order line (Yes/No). </pre>

		Only for Purchase Order Lines
	i.effectivity.unit	Effectivity unit
	i.product.variant	Product Variant from the order line
	i.option.list.id	Option list id from the order line
	i.buy.sell.bp	Purchase: buy-from business partner Sales: sold-to business partner
	i.ship.bp	Purchase: ship-from business partner Sales: ship-to business partner
	i.pricing.bp	Parent Business Partner for Prices and Discounts.
	i.product.class	Product Class
	i.product.line	Product Line
	i.manufacturer.part.number	Manufacturer Part Number
	i.manufacturer	Manufacturer
	i.order.date	Order date from the line.
	i.planned.delivery.date	Planned delivery date from line
	i.order.quantity	Total order quantity from line.
	i.original.order.quantity	Original order quantity of line. In add mode, this is zero. In edit mode this is set to the quantity before the line was edited.
	i.price.group.quantity	Quantity used to apply discounts for 'like' price groups.
	i.quantity.unit	Unit in which the quantity is expressed
	i.quantity.unit.conversion.factor	Conversion factor from quantity units to inventory unit
	i.delete.manual.discount	Delete manual discounts (True/ False).
	i.delete.pricebook.discount	Delete discount applied from the price book (True/False).
	i.apply.price.discount.both	Apply a price, apply the discounts or both.
	i.price.percentage	Retrieve a price or a percentage
	i.update.uef	Update Total Upgrade Price in UEF or retrieve UEF Total Upgrade Price.
	i.update.pcf	Update Price Structure in PCF or retrieve PCF price.
	i.financial.company	Financial company (Belongs to sales / purchase office.
	i.transaction.currency	Transaction currency.
	i.rate	Exchange rate array.
	i.rate.factor	Rate factor array.
	i.rate.date	Rate date
	i.rate.type	Rate type
	i.price.type	Price Type; possible values: Not Applicable Buying Item Subcontracting Operation Subcontracting Service Subcontracting
	i.production.order	Linked production order
	i.maintenance.work.order	Linked work order
	i.service.order	Linked service order
	i.channel	Channel (Sales order only)
	i.delivery.terms	Delivery Terms (Sales order only)
	i.default.price.book	Default Price Book; possible values:

		Sales Service	
	i.price.unit.from.standard.logic	As determined under	
standard logic			
	i.price.unit.conv.factor.from.standard.logic	As determined under	
standard logic			
	i.price.from.standard.logic	As determined under	
standard logic			
	i.percentage.from.standard.logic	As determined under	
standard logic			
	i.derived.item.used.from.standard.logic	As determined under	
standard logic			
	i.price.book.from.standard.logic	As determined under	
standard logic			
	i.price.stage.from.standard.logic	As determined under	
standard logic			
	i.price.origin.from.standard.logic	As determined under	
standard logic			
	i.price.matrix.definition.from.standard.logic	As determined under	
standard logic			
	i.price.matrix.sequence.from.standard.logic	As determined under	
standard logic			
		Arrays below have been	
allocated			
		with 11 elements.	
	i.disc.matrix.type.from.standard.logic(11)	As determined under	
standard logic			
	i.discount.origin.from.standard.logic(11)	As determined under	
standard logic			
	i.disc.matrix.definition.from.standard.logic(11)	As determined under	
standard logic			
	i.disc.matrix.sequence.from.standard.logic(11)	As determined under	
standard logic			
	i.discount.percentage.from.standard.logic(11)	As determined under	
standard logic			
	i.discount.amount.from.standard.logic(11)	As determined under	
standard logic			
	i.discount.code.from.standard.logic(11)	As determined under	
standard logic			
	i.discount.method.from.standard.logic(11)	As determined under	
standard logic			
	i.discount.schedule.from.standard.logic(11)	As determined under	
standard logic			
	i.determining.from.standard.logic	As determined under	
standard logic			
	i.eligible.from.standard.logic	As determined under	
standard logic			
	Output:		
	o.price.unit	Price Unit	
	o.price.unit.conv.factor	Conversion factor from Price Unit to inventory unit	
	o.price	Unit Price	
	o.percentage	Percentage	
	o.derived.item.used	The derived from item was used to retrieve the price (True/False).	
	o.price.book	Price Book	
	o.price.stage	Price Stage	
	o.price.origin	Price Origin	
	o.price.matrix.definition	Price Matrix Definition	
	o.price.matrix.sequence	Price Matrix Sequence	
		Arrays below have been allocated	

	<pre> o.disc.matrix.type(11) o.discount.origin(11) o.disc.matrix.definition(11) o.disc.matrix.sequence(11) o.discount.percentage(11) o.discount.amount(11) o.discount.code(11) o.discount.method(11) o.discount.schedule(11) o.determining o.eligible o.fallback.to.standard.logic.in.case.of.errors Return: 0 <> 0 </pre>	<pre> with 11 elements. Discount Matrix Type array Discount Origin array Discount Matrix Definition array Discount Matrix Sequence array Discount Percentage array Discount Amount array Discount Code array Discount Method array Discount Schedule array Determining (Yes/No) Eligible (Yes/No) True: When an error is returned by the extension, then LN will use the outcome of the LN standard logic. False: Errors in the extension are returned as-is to LN. Output variables are initialized to 0/empty/"" Success An error occurred </pre>

Process Extensions for Procurement

The following process extension(s) is/are available:

[Procurement.DefaultPurchaseOffice](#)

[Procurement.HandleGeneralLedgerCodeDefaulting](#)

Procurement.DefaultPurchaseOffice

Determine customer specific default Purchase Office for Procurement.

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to set a customer determined purchase office from an extension to a value other than the defaulted value based on LN logic . This purchase office (if valid) is then used in all flows where a purchase office is defaulted.

To implement this process extension, you need to implement the following method(s):

tdext.pur0001.get.customer.determined.default.purchase.office

Syntax:	<pre> long tdext.pur0001.get.customer.determined.default.purchase.office(domain tcmcs.str8 i.defaulting.origin, domain tccwoc i.purchase.office.from.standard.logic, domain tcomm.grid i.enterprise.unit, domain tccom.bpid i.buy.from.business.partner, domain tccom.bpid i.ship.from.business.partner, domain tcitem i.item, domain tcncmp i.logistic.company, domain tcsite i.site, domain tccwar i.warehouse, ref domain tccwoc o.purchase.office) </pre>
Usage:	Expl: Use this method to get a customer determined default purchase office from an extension. This office will be used in Procurement. The standard logic in LN determines the purchase office from master data like enterprise unit, item data or user profile. Using this process extension, the defaulted value can be changed to another value. Note: 1. The input arguments can be used to determine the option. 2. LN will check if the provided default is valid. 3. The purchase office defaulted based on standard LN logic will overrule the selected office from the extension, otherwise standard logic may fail. Pre: NA Post: NA Input: i.defaulting.origin - Defaulting Origin, Possible Values: - tdpur100 - RFQ - tdpur200 - Purchase Requisition - tdpur300 - Purchase Contract - tdpur310 - Purchase Schedule - tdpur400 - Purchase Order - tdpur600 - Services Procurement Order - Empty - Generation of a purchase order / schedule via 'generate purchase order/schedule flows' when e.g., calling from other packages (Project, Service, Warehousing, etc.). i.purchase.office.from.standard.logic - As defaulted under the standard logic i.enterprise.unit - Enterprise Unit i.buy.from.business.partner - Buy-from Business Partner i.ship.from.business.partner - Ship-from Business Partner i.item - Item i.logistic.company - Logistic Company i.site - Site i.warehouse - Warehouse Output: o.purchase.office - The purchase office determined by the extension. Return: 0 - Success

	DALHOOKERROR	-	When an error occurs in the determination of the purchase office.

Procurement.HandleGeneralLedgerCodeDefaulting

Customize the logic for defaulting the general ledger code in LN Procurement processes.

This process extension is available from [KB3613480](#).

Technical information for this process extension:

Usage:	This Process Extension allows customization of the logic used for defaulting the general ledger code in Procurement processes. Specifically, it enables you to override or skip the defaulting behavior of the general ledger code. The extension is invoked whenever the standard LN logic attempts to determine a default ledger code for Procurement-related objects.

To implement this process extension, you need to implement the following method(s):

tdext.pur0005.skip.general.ledger.code.defaulting

Syntax:	<pre>boolean tdext.pur0005.skip.general.ledger.code.defaulting(boolean i.user.interaction, domain tcmcs.tabl i.from.table, domain tcsess i.from.session, domain tcfieldname i.table.field)</pre>
Usage:	Expl: Use this method to skip the defaulting of the general ledger (GL) code in Procurement processes. The input arguments provided to this method depend on the context from which it is invoked. There are two possible origins: * DAL (Data Access Layer): In this context, no user interaction is allowed. You can use any table field from the given table to determine the output. However, note that the session that triggered the defaulting logic is not available in this context. * UI (User Interface): In this context, LN identifies the specific table field that was modified and triggered the defaulting logic. It also knows which session caused the trigger. You can use any table field from the given table to determine the output. Pre: NA Post: NA Input: i.user.interaction - Indicates whether user interaction is allowed in the current context.

	i.from.table - The table-code for which LN is determining a default ledger code. (Mandatory). Supported Values: - tdpur201 Requisition Line - tdpur401 Purchase Order Line i.from.session - The session that triggered the defaulting. This field is empty when called from the DAL or from a non-UI context. Possible Values (among others): - tdpur2502m000 - tdpur2502m100 - tdpur4101m000 - tdpur4101m100 i.table.field - The table field that was modified and triggered the defaulting logic. This field is empty when called from the DAL or from a non-UI context. Possible Values: - Any table field from <i.from.table> that can trigger the general ledger code defaulting logic. Output: Not Applicable. Return: true - The extension has determined that general ledger code defaulting must be skipped for the given input. - The standard logic for defaulting applies. false

Process Extensions for ProductionOrder

The following process extension(s) is/are available:

[ProductionOrder.PrintCustomReports](#)

[ProductionOrder.SkipCalculateIntermediateResults](#)

[ProductionOrder.SkipClose](#)

[ProductionOrder.SkipCreateOrderGroup](#)

[ProductionOrder.SkipGenerateSubcDocuments](#)

[ProductionOrder.SkipPrintMaterialShortage](#)

[ProductionOrder.SkipReportComplete](#)

[ProductionOrder.SkipWarehouseForMaterialShortage](#)

ProductionOrder.PrintCustomReports

Production Order Custom Reports.

Technical information for this process extension:

Usage:	With this Process Extension it is possible to integrate Custom Linked Reports to the session Print Production Order Documents (tisfc0408m000) as additions to the Production Order's standard document set. The extension provides hooks for: - Registration of the Custom Linked Reports (method tiext.sfc0001.register.custom.reports). - Decision logic to decide whether or not for a specific order the document should be printed (method tiext.sfc0001.check.print.condition). - The retrieval of required data and sending it to the report driver (method tiext.sfc0001.print.custom.report). - The update of the print status for the report (method tiext.sfc0001.update.print.status). Note that for this Process Extension to work, in session extension Print Production Order Documents (tisfc0408m000), each Custom Linked Report must be added as the single report in a new report group.

To implement this process extension, you need to implement the following method(s):

tiext.sfc0001.check.print.condition

Syntax:	<pre>long tiext.sfc0001.check.print.condition(domain tcmcs.long i.report.group, domain tcyesno i.print.originals, domain tcyesno i.print.duplicates, domain tcyesno i.print.modified, ref boolean o.print.report, ref string o.message())</pre>
Usage:	Expl: This function is called in session tisfc0408m000 when determining which custom reports to print. The function is called once for every Production Order and every report to be printed. The tisfc001 record for the Production Order to be printed has been made current. Implementation example: Create a custom tx-table (example txtext001) with an index of the production order and the report group. Add a field (ex. txtext001.stat) in the created tx-table for the print status with domain tisfc.pdst. Write the function as follows: <pre>function extern long tiext.sfc0001.check.print.condition(domain tcmcs.long i.report.group, domain tcyesno i.print.originals, domain tcyesno i.print.duplicates,</pre>

	<pre> domain tcyesno i.print.modified, ref boolean o.print.report, ref string o.message) { domain tisfc.pdst print.status print.status = get.print.status.(tisfc001.pdno, i.report.group) if print.status = tisfc.pdst.original.doc and i.print.originals = tcyesno.yes then return(true) endif if print.status = tisfc.pdst.modified.doc and i.print.modified = tcyesno.yes then return(true) endif if print.status = tisfc.pdst.doc.printed and i.print.duplicates = tcyesno.yes then return(true) endif return(false) } function domain tisfc.pdst get.print.status(domain tcpdno i.production.order, domain tcms.long i.report.group) { domain tisfc.pdst print.status select txext001.stat:print.status from txext001 where txext001._index1 = { :i.production.order, :i.report.group} selectempty print.status = tisfc.pdst.original.doc endselect return(print.status) } </pre> Pre: The custom reports must be registered using tiext.sfc0001.registrate.custom.reports(). Post: Print the custom reports using tiext.sfc0001.print.custom.report() Input: - i.report.group - The report group of the custom report - i.print.originals - Print Option: original reports must be printed. - i.print.duplicates - Print Option: duplicate reports must be printed. - i.print.modified - Print Option: modified reports must be printed. Output: o.print.report - If true, the custom report should be printed. If false, the custom report should not be printed. o.message - message, multibyte - max 300 characters

	Return: 0	- Success
	DALHOOKERROR	- When an error occurs in the added logic

tiext.sfc0001.print.custom.report

Syntax:	<pre> long tiext.sfc0001.print.custom.report(domain tcmcs.long i.report.group, domain tcmcs.st14 i.spooler.device, long i.lfn.custom.spool, ref o.lfn.custom.brpfile, ref o.message()) </pre>
Usage:	Expl: This function is called to print the custom reports, once for every production order. The spooler device has already been opened and the spool.id has been provided in i.lfn.custom.spool. The function should open the report using the spooler device, read all required data and send it to the custom report. This function is called once for every custom report and every Production Order to be printed. The tisfc001 record for the Production Order to be printed has been made current. Implementation Example: A custom report that prints the production order has been created in report group 30 using spooler device "D". <pre> extern long lfn.custom.spool * External logical Spool File extern long lfn.custom.brpfile * External logical File extern boolean open.custom.brp * Ext brp File Opening table ttisfc001 * Production Order tiext.sfc0001.print.custom.report(domain tcmcs.long i.report.group, domain tcmcs.st14 i.spooler.device, long i.lfn.custom.spool, ref o.lfn.custom.brpfile, ref o.message) { reportgrp = i.report.group choice.report(report\$) if not open.custom.brp then open.custom.brpfile(true, i.spooler.device, i.lfn.custom.spool, o.lfn.custom.brpfile) endif brp.ready(o.lfn.custom.brpfile) return(0) } </pre>

	<pre> function open.custom.brpfiler(boolean i.do.brpfiler.open, domain tcmcs.st14 i.spooler.device, long i.lfn.custom.spool, ref long o.lfn.custom.brpfiler) { if i.lfn.custom.spool = 0 then return endif spool.id = i.lfn.custom.spool if i.do.brpfiler.open then o.lfn.custom.brpfiler = brpfiler.open(report\$, i.spooler.device, 0) if o.lfn.custom.brpfiler <= 0 then job.process.error = true return endif open.custom.brpfiler = true endif } </pre> Pre: Determine if the custom report must be printed using tiext.sfc0001.check.print.condition(). Post: Update the print status using tiext.sfc0001.update.print.status(). Input: i.report.group - The report group that will be printed. i.spooler.device - The spooler device that the report is to be printed to. i.lfn.custom.spool - The logical Spool File for the spooler device opened for the custom report. Output: o.lfn.custom.brpfiler - The logical Baan report File opened for the custom report. o.message - message, multibyte - max 300 characters Return: 0 - Success DALHOOKERROR - When an error occurs in the added logic

tiext.sfc0001.register.custom.reports

Syntax:	<pre> long tiext.sfc0001.register.custom.reports(ref long o.number.of.custom.reports, ref long o.custom.report.groups(), ref domain tcmcs.st14 o.custom.print.devices() fixed, ref string o.message()) </pre>
Usage:	Expl: This function is called when session Print Production Order Documents (tisfc0408m000), is started. With this method, one or more custom reports can be registered and spooler devices can be defined. All custom report groups must 30 or higher. Implementation example:

	Implementing a custom report linked to tisfc0408m000 in reportgroup 30 using spooler device "D". <pre> function extern long tiext.sfc0001.register.custom.reports(ref long o.number.of.custom.reports, ref long o.custom.report.groups(), ref domain tcmcs.st14 o.custom.print.devices(), ref string o.message()) { string dummy.string(1) long dummy.long long domain.length rdi.domain("tcmcs.st14", dummy.string, dummy.string, dummy.string, dummy.long, dummy.string, dummy.long, domain.length) alloc.mem(o.custom.report.groups, 1) alloc.mem(o.custom.report.spools, domain.length, 1) o.custom.report.groups(1) = 30 o.custom.report.spools(1, 1) = "D" o.number.of.custom.reports = 1 return (0) } </pre> Pre: Use the extension modeler to link the custom report to session tisfc0408m000. Post: Determine if the custom report must be printed using tiext.sfc0001.check.print.condition(). Input: - Output: o.number.of.custom.reports - The number of custom reports registered in this method. o.custom.report.groups - The array with the group numbers of all custom reports. o.custom.report.spools - The array with the spooler devices for the custom reports. o.message - message, multibyte - max 300 characters. Return: 0 - Success DALHOOKERROR - When an error occurs in the added logic

tiext.sfc0001.update.print.status

Syntax:	<pre> long tiext.sfc0001.update.print.status(ref boolean i.custom.report.printed(), </pre>

	ref	string	o.message())
Usage:	Expl:	This function is called the reports have been printed for a given Production Order. The function is called once for every Production Order. Use this function to update the print status for the custom reports. The tisfc001 record for the Production Order to be printed has been made current. The external variable proc_ext_print_cust_rep_num_of_cust_reports is available and contains the length of the array. The external array proc_ext_print_cust_rep_report_groups is available and contains the array of report groups. Implementation example: Create a custom tx-table (example txtext001) with an index of the production order and the reportgroup. Add a field (ex. txtext001.stat) in the created tx-table for the print status with domain tisfc.pdst. Write the function as follows: <pre> extern long proc_ext_print_cust_rep_num_of_cust_reports extern domain tcmcs.long proc_ext_print_cust_rep_report_groups(1) based function extern long tiext.sfc0001.update.print.status(ref boolean i.custom.report.printed(), ref string o.message) { long report long ret for report = 1 to proc_ext_print_cust_rep_num_of_cust_reports if i.custom.report.printed(report) = true then ret = set.print.status.printed(tisfc001.pdno, proc_ext_print_cust_rep_report_groups(report)) if ret <> 0 then return(ret) endif endif endfor return(0) } function long set.print.status.printed(domain tcpdno i.production.order, domain tcmcs.long i.report.group) { domain tisfc.pdst status long ret select txtext001.stat:status from txtext001 for update where txtext001._index1 = { :i.production.order, :i.report.group} selectdo ret = dal.change.object(txtext001) if ret <> 0 then </pre>	

	<pre> return(ret) endif dal.set.field("txext001.stat", tisfc.pdst.doc.printed) ret = dal.save.object(txext001) if ret <> 0 then return(ret) endif endselect return(0) } Pre: Print the custom reports using tiext.sfc0001.print.custom.report(). Post: - Input: i.custom.report.printed - The array indicating if the report has been printed. The same sequence is used as the array with the custom report groups. Output: o.message - message, multibyte - max 300 characters Return: 0 - Success DALHOOKERROR - When an error occurs in the added logic </pre>

ProductionOrder.SkipCalculateIntermediateResults

Skips Production Orders when executing Calculate Intermediate Results.

This process extension is available from [KB2294561](#).

Technical information for this process extension:

Usage:	Process Extension ProductionOrder.SkipCalculateIntermediateResults can be used to skip intermediate result calculation for specific Production Orders. A message may be filled, to present information about the skip decision on the report. This message can have max 300 characters. Sessions where this Process Extension can be implemented: - Calculate Intermediate Results (ticst0201m100) Fields that are available to be used in this Process Extension: - All fields of table: Production Orders (tisfc001) Pseudocode: Below you can find an example. Hook: Declarations <pre> table tisfc001 </pre> Hook: ext.skip.with.reason <pre> function extern boolean ext.skip.with.reason(ref string o.reason) { </pre>

	<pre> if <condition on tisfc001 = true> then o.reason = "Production Order skipped because ..." return(true) endif return(false) } </pre>

ProductionOrder.SkipClose

Skips Production Orders when executing Close Orders.

This process extension is available from [KB2284251](#).

Technical information for this process extension:

Usage:	Process Extension ProductionOrder.SkipClose can be used to skip closure of specific Production Orders. A message may be filled, to present information about the skip decision on the report. This message can have max 300 characters. Sessions where this Process Extension can be implemented: - Close Production Orders (ticst0201m000) Fields that are available to be used in this Process Extension: - All fields of table: Production Orders (tisfc001)

ProductionOrder.SkipCreateOrderGroup

Skips Creating a Production Order Group.

This process extension is available from [KB3548856](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension ProductionOrder.SkipCreateOrderGroup can be used to skip production order grouping. Sessions where this Process Extension can be implemented: - Create Production Order Group (tisfc3250m000) Field that is available to be used in this Process Extension: - All fields of table Production Orders (tisfc001) Note: tables must also be declared in the Process Extension. Pseudocode: Below you can find an example:

	<pre> Hook: Declarations table ttisfc001 Hook: ext.skip function extern boolean ext.skip() { if <condition on tisfc001 = true> then return(true) endif return (false) } </pre>

ProductionOrder.SkipGenerateSubcDocuments

Skips generation of Subcontracting Purchase Documents for Production Order Operations.

This process extension is available from [KB2322308](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension ProductionOrder.SkipGenerateSubcDocuments can be used to skip certain Production Order Operations when generating subcontracting purchase documents. Sessions where this Process Extension can be implemented: - Generate Subcontracting Purchase Documents (tisfc2250m000) Fields that are available to be used in this Process Extension: - All fields of table: Production Order Operations (tisfc010) - From table Production Orders, fields: - tisfc001.site - tisfc001.clco

ProductionOrder.SkipPrintMaterialShortage

Skips shortages when printing.

This process extension is available from [KB3543720](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension ProductionOrder.SkipPrintMaterialShortage can be used to skip printing specific Material Shortages. Sessions where this Process Extension can be implemented: - Print Material Shortages by Production Order (tisfc0419m000)

	Field that is available to be used in this Process Extension: - timfc001.oorg (Order Origin) - timfc001.pdno (Production Order) - timfc001.pono (Position) - timfc001.seqn (Sequence Number) - timfc001.ittp (Transaction Type) - timfc001.cwar (Warehouse)

ProductionOrder.SkipReportComplete

Skips Production Orders when Reporting Orders as Complete.

To implement this process extension, you can use the information below:

Usage:	Process Extension ProductionOrder.SkipReportComplete can be used to skip certain Production Orders when reporting a range of Production Orders as complete. Sessions where this Process Extension can be implemented: - Report Orders Completed Globally (tisfc0206m000) Fields that are available to be used in this Process Extension: - All fields of table: Production Orders (tisfc001)

ProductionOrder.SkipWarehouseForMaterialShortage

Skips Warehouses when Calculating Shortage.

This process extension is available from [KB3628424](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension ProductionOrder.SkipWarehouseForMaterialShortage can be used to skip considering certain warehouses for available material inventory during the shortage calculation. Sessions where this Process Extension can be implemented: - Print Material Shortages by Production Order (tisfc0419m000) - Release Production Orders (tisfc0204m000) External variables made available to be used in this Process Extension: - proc_ext_skip_warehouse_for_mat_shortage_warehouse (Warehouse) (domain tccwar) - proc_ext_skip_warehouse_for_mat_shortage_cluster (Cluster) (domain tcemm.clus) - proc_ext_skip_warehouse_for_mat_shortage_material (Material) (domain tcitem) - proc_ext_skip_warehouse_for_mat_shortage_production_order (Production Order) (domain tcpdno)

	<pre> Pseudocode: Below you can find an example. Hook: Declarations Hook: ext.skip function extern boolean ext.skip() { if <condition on proc_ext_skip_warehouse_for_mat_shortage_warehouse > then return(true) endif return (false) } </pre>

Process Extensions for ProductVariant

The following process extension(s) is/are available:

[ProductVariant.AddCustomCPQIntegrationParameters](#)

[ProductVariant.AddToCPQConfiguration](#)

[ProductVariant.AdjustPrice](#)

[ProductVariant.SkipDeleteOrExpireJSBOM](#)

ProductVariant.AddCustomCPQIntegrationParameters

Add Custom CPQ Integration Parameters to configuration request.

This process extension is available from [KB2291878](#).

Technical information for this process extension:

Usage:	<pre> This process extension is called during construction of the configuration request for CPQ. In addition to the standard CPQ Integration Parameters, custom integration parameters may be added to the request structure. The extension provides hooks for: - Registration of the additional CPQ integration parameters (method tiext.pcf0003.register.cpq.integration.parameters). - Retrieval of a value of a registered additional CPQ integration parameter (method tiext.pcf0003.get.cpq.integration.parameter.value). </pre>

To implement this process extension, you need to implement the following method(s):

tiext.pcf0003.get.cpq.integration.parameter.value

Syntax:	<pre> long tiext.pcf0003.get.cpq.integration.parameter.value(domain tcmcs.str50 i.parameter.name, ref long o.type, ref boolean o.is.null, ref string o.string.value(), ref long o.long.value, ref double o.double.value) </pre>
Usage:	Expl: Return the proper value for the additional CPQ integration parameter. Variables made current in support of decision logic: - CPQ configurator settings (tipcf011) table fields: - tipcf011.ccfg: Configuration Group - tipcf011.item: Configurable Item - tipcf011.name: Namespace - tipcf011.prid: Profile - Product Variant IDs (tipcf500): all table fields. Supported types in o.type and mapping to CPQ base type: <pre> 10 - string --> o.string.value -> CPQ type String 20 - long --> o.long.value -> CPQ type Number 30 - double --> o.double.value -> CPQ type Number 40 - date-time --> o.string.value -> CPQ type DateTime date-time format "YYYYMMDD HHMMSS" 50 - boolean --> o.long.value -> CPQ type Boolean </pre> Implementation example: <pre> o.type = 0 o.string.value = "" o.long.value = 0 o.double.value = 0.0 o.is.null = false on case strip\$(i.parameter.name) case "ParameterString": o.type = 10 o.string.value = "CustomIntegrationParameterValue" break case "ParameterLong": o.type = 20 /* This example assumes a long value is kept in a /* customer defined field, retrieval requires generic /* item and business partner. o.long.value = read.cdf.field.for.item.business.partner(tipcf500.item, tipcf500.cuno) break case "ParameterDouble": o.type = 30 o.double.value = 3.1415927 break case "ParameterDate": o.type = 40 </pre>

	<pre> o.string.value = "20240124 141632" break case "ParameterBoolean": o.type = 50 o.long.value = true break case "ParameterNull": o.is.null = true o.type = 10 break default: /* Unknown parameter name, will be encoded, but with /* NULL value, and type String o.is.null = true o.type = 10 break endcase return(0) Pre: Parameter was registered via method: tiext.pcf0003.register.cpq.integration.parameters. Post: Input: i.parameter.name - name of a registered custom CPQ integration parameter. Output: o.type - type, see supported types in description above o.is.null - encode parameter using NULL value o.string.value - string value if needed for type, max 512 characters. o.long.value - long value if needed for type o.double.value - long value if needed for type Return: 0 successfully set output variables <> 0 to indicate failure </pre>

tiext.pcf0003.register.cpq.integration.parameters

Syntax:	<pre> long tiext.pcf0003.register.cpq.integration.parameters(ref long o.number.of.parameters, ref domain tcmcs.str50 o.parameter.names() fixed) </pre>
Usage:	Expl: This function is called when for a product variant a configuration request towards CPQ is prepared. A set of (custom) CPQ Integration parameters can be registered. Implementation example: <pre> /* allocate memory for integration parameters o.number.of.parameters = 6 alloc.mem(o.parameter.names, 50, o.number.of.parameters) o.parameter.names(1,1) = "ParameterString" o.parameter.names(1,2) = "ParameterLong" o.parameter.names(1,3) = "ParameterDouble" o.parameter.names(1,4) = "ParameterDate" o.parameter.names(1,5) = "ParameterBoolean" o.parameter.names(1,6) = "ParameterNull" </pre>

	<pre> return(0) Pre: Post: Input: - Output: o.number.of.parameters - The number of custom CPQ integration parameters registered. o.parameter.names - The array with custom CPQ integration parameter names. Allocation of this output array is required. Return: 0 - Success DALHOOKERROR - When an error occurs in the added logic </pre>

ProductVariant.AddToCPQConfiguration

Execute additional logic directly after finishing the configuration process of a CPQ product variant.

Technical information for this process extension:

Usage:	This extension method is called after finishing the configuration process for a product variant configuration by CPQ. This is after the product variant is updated with engineering options, price structures and status. It applies for the interactive configuration (raised from the Configure option in various sessions) as well as for the background configuration of sales options (either directly via SalesOrderBOD processing, or via tipcf5205m000). It allows the implementor to add logic for creating or updating related order line data and/or other related data in LN. This can be based on additional data elements or rules in the related CPQ ruleset.

To implement this process extension, you need to implement the following method(s):

tiext.pcf0002.add.to.cpq.configuration

Syntax:	<pre> long tiext.pcf0002.add.to.cpq.configuration(long i.service.node, long i.request.node, long i.response.node, domain tipcf.acfs i.cpq.result.status, domain tccpva i.product.variant, domain tccpva i.reused.product.variant, domain tcqsll i.config.qty, domain tcolid i.option.list.id) </pre>
Usage:	Expl: This extension method is called after finishing the configuration process for a product variant configuration by CPQ. This is after the product variant is updated with

	engineering options, price structures and status. It applies for the interactive onfiguration (raised from the Configure option in various sessions) as well as for the background configuration of sales options (either directly via SalesOrderBOD processing, or via tipcf5205m000). It allows the implementor to add logic for creating or updating related order line data and/or other related data in LN. This can be based on additional data elements or rules in the related CPQ ruleset. The implementation of the method must contain its own transaction management (db.retry.point, commit or abort transaction). Pre: Post: Input: - i.service.node - Service Node (XML Structure) - i.request.node - Request Node (XML Structure) - i.response.node - CPQ configuration (XML Structure) - i.cpq.result.status - CPQ Configurator Status - i.product.variant - Product Variant - i.reused.product.variant - Reused Product Variant: - an existing product variant (<> i.product.variant) - or zero when no product variant is reused - i.config.qty - Configured quantity - i.option.list.id - Option List ID Output: Return: 0 - Success - DALHOOKERROR - When an error occurs in the added logic

ProductVariant.AdjustPrice

Adjust the price of a product variant at the end of the price calculation.

Technical information for this process extension:

Usage:	Adjust the Product Variant Sales Price (Surcharge) or Purchase Price (Surcharge) at the end of the price calculation.

To implement this process extension, you need to implement the following method(s):

tiext.pcf0001.adjust.variant.price

Syntax:	long tiext.pcf0001.adjust.variant.price(ref domain tipcf.pric io.price)
Usage:	Expl: This method is called at the end of the sales or purchase price calculation for a certain line of the generic price

	list, just before rounding the final result. So when a generic price list has multiple lines, this method is called during processing of each of these lines. With this method, the value of calculated sales or purchase price can be adjusted. When this method is called, the Generic Price List record (tipcf400) is read and all tipfc400 fields are current (including the Customer Defined Fields) and can be used in this method. EXAMPLE: Assume that tipcf400.acst (Administration Cost) is a CDF on table tipcf400, and must be applied during the sales price calculation. The following logic could be added to adjust the calculated price <pre> if tipcf400.kopl = tckopl.sales then io.price = io.price + tipfc400.acst endif </pre> Pre: when tipcf400 fields are used: include table ttipcf400 in the global variable section of the extension. Post: N.A. Input: io.price Output: io.price Return: 0 - Success DALHOOKERROR - When an error occurs in the added logic

ProductVariant.SkipDeleteOrExpireJSBOM

Skips Delete or Expire Job Shop Bill of Materials Revisions.

This process extension is available from [KB3528322](#).

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to skip delete/expire of the job shop bill of materials when generating the project structure for a product variant, provided the 'regenerate existing product variant' option is selected. This public interface only works when the Sites concept is active.

To implement this process extension, you need to implement the following method(s):

tiext.pcs0002.product.variant.skip.delete.or.expire.jsbom

Syntax:	<pre>long tiext.pcs0002.product.variant.skip.delete.or.expire.jsbom(domain tccpva i.product.variant, domain tcitem i.parent.item, domain tcsite i.site, domain tibmdl i.bill.of.material, domain tibmrsv i.revision, domain tibmst i.status, ref boolean o.skip.delete.or.expire.jsbom)</pre>												
Usage:	Expl: This function is used to skip delete/expire of the job shop bill of materials when generating the project structure for a product variant, provided the 'regenerate existing product variant' option is selected. This public interface only works when the Sites concept is active. Implementation Example: Intention: For a specific Parent Item (SLO), when Regenerate Product Variant Structure is requested, Delete or Expire Job Shop Bill Materials is not allowed. Hook Declarations: Hook tiext.pcs0002.product.variant.skip.delete.or.expire.jsbom: <pre>function extern long tiext.pcs0002... ... { if str.startswith(i.parent.item, "SLO") then o.skip.delete.or.expire.jsbom = true endif return(0) }</pre> Pre: NA Post: NA Input: <table><tr><td>i.product.variant</td><td>- Product Variant</td></tr><tr><td>i.parent.item</td><td>- Parent Item</td></tr><tr><td>i.site</td><td>- Site</td></tr><tr><td>i.bill.of.material</td><td>- Bill of Material</td></tr><tr><td>i.revision</td><td>- Revision</td></tr><tr><td>i.status</td><td>- Status</td></tr></table> Output: o.skip.delete.or.expire.jsbom - Skip Delete or Expire Job Shop Bill of materials. Return: 0 - Success. DALHOOKERROR - When an error occurs in determination of the skip job shop bill of materials.	i.product.variant	- Product Variant	i.parent.item	- Parent Item	i.site	- Site	i.bill.of.material	- Bill of Material	i.revision	- Revision	i.status	- Status
i.product.variant	- Product Variant												
i.parent.item	- Parent Item												
i.site	- Site												
i.bill.of.material	- Bill of Material												
i.revision	- Revision												
i.status	- Status												

Process Extensions for ProjectBudget

The following process extension(s) is/are available:

[ProjectBudget.SkipGenerateServiceOrders](#)

ProjectBudget.SkipGenerateServiceOrders

Skips Projects when Generating Service Orders from a Project Budget.

To implement this process extension, you can use the information below:

Usage:	Process Extension <code>ProjectBudget.SkipGenerateServiceOrders</code> can be used to skip certain Projects when Generating Service Orders from a Project Budget. Sessions where this Process Extension can be implemented: - Generate Service Orders (tppss6250m000) Fields that are available to be used in this Process Extension: - All fields of table: Projects (tppdm600)

Process Extensions for ProjectControl

The following process extension(s) is/are available:

[ProjectControl.SkipPrint](#)

ProjectControl.SkipPrint

Skips Projects when Printing Project Control Data.

To implement this process extension, you can use the information below:

Usage:	Process Extension <code>ProjectControl.SkipPrint</code> can be used to skip certain Projects when Printing Project Control data. Sessions where this Process Extension can be implemented: - Print Cost Control (tpppc4411m000) - Print Principals Project Control Report (tpppc4412m000) - Print Project Managers Control Report (tpppc4413m000) - Print Project Control (tpppc4420m000) - Print Management Report (tpppc4421m000) Fields that are available to be used in this Process Extension: - All fields of table: Projects (tppdm600)

Process Extensions for ProjectNetworkPlanning

The following process extension(s) is/are available:

[ProjectNetworkPlanning.SkipGenerate](#)

ProjectNetworkPlanning.SkipGenerate

Skips Projects when Generating Network Planning in Project.

To implement this process extension, you can use the information below:

Usage:	Process Extension ProjectNetworkPlanning.SkipGenerate can be used to skip certain Projects when Generating Network Planning in Project. Sessions where this Process Extension can be implemented: - Generate Network Planning (tpss2240m000) Fields that are available to be used in this Process Extension: - All fields of table: Projects (tpdm600)

Process Extensions for ProjectPCS

The following process extension(s) is/are available:

[ProjectPCS.SkipClose](#)

[ProjectPCS.SkipGlobalStatusUpdate](#)

ProjectPCS.SkipClose

Project PCS Skip Close of Project.

Technical information for this process extension:

Usage:	With this process extension, the Project close functionality can be controlled for specific Main Projects, Sub Projects & Single Projects.

To implement this process extension, you need to implement the following method(s):

tiext.pcs0001.project.pcs.skip.close

Syntax:	<pre>long tiext.pcs0001.project.pcs.skip.close(ref boolean o.skip.close, ref string o.message())</pre>
Usage:	Expl: This function is called in the process of Closing Projects: - Main Projects and Structures - Sub Projects - Single Projects To decide if the process must be skipped for a specific Closing Project Type. A message may be returned, to present information about the decision to the user. When this function is called all fields of tables: - General Project Data (tipcs020) - Project Details (tipcs030) are current. Pre: NA Post: NA Input: NA Output: o.skip.close - decision result wherein o.message - message, multibyte - max 300 characters Return: 0 - success DALHOOKERROR - error

tiext.pcs0001.project.pcs.skip.close.implemented

Syntax:	<pre>boolean tiext.pcs0001.project.pcs.skip.close.implemented()</pre>
Usage:	Expl: This function checks if the process extension ProjectPCS.SkipClose is implemented. pre: N.a. Post: N.a. Input: N.a. Output: N.a. Return: true/false

ProjectPCS.SkipGlobalStatusUpdate

Skips status update when executing Global Status Update of PCS projects.

This process extension is available from [KB3518588](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension ProjectPCS.SkipGlobalStatusUpdate can be used to skip certain PCS Project when changing the status for a range of Projects. Sessions where this Process Extension can be implemented: - Global Update of Project Status (tipcs2280m000) Fields that are available to be used in this Process Extension: - All fields of table: Project Details (tipcs030) - From table Description and Project Type, fields: tipcs020.kopr

Process Extensions for ProjectPlannedPurchaseOrder

The following process extension(s) is/are available:

[ProjectPlannedPurchaseOrder.SkipApprove](#)

ProjectPlannedPurchaseOrder.SkipApprove

Skips Planned Purchase Orders in Project for Material, Equipment and Subcontracting when Approving.

To implement this process extension, you can use the information below:

Usage:	Process Extension ProjectPlannedPurchaseOrder.SkipApprove can be used to skip Planned Purchase Orders in Project for cost types Material, Equipment and Subcontracting when Approving. Sessions where this Process Extension can be implemented: - Approve Planned PRP Purchase Orders (tppss6220m000) Fields that are available to be used in this Process Extension: - All fields of table: Planned PRP Purchase Orders - Material (tppss610) - All fields of table: Planned PRP Purchase Orders - Equipment (tppss611) - All fields of table: Planned PRP Purchase Orders - Subcontracting (tppss612) External variables that are available to be used in this Process Extension: - proc_ext_cost_type [type: string(20)] Note: tables and external variables must also be declared in the Process Extension.

Pseudocode

The session Approve Planned PRP Purchase Orders can approve planned orders for 3 cost types: Material, Equipment and Subcontracting. Each cost type has its own table, so the process extension must be applied on these 3 tables. Available table fields per cost type:

- Material - tppss610 (variable proc_ext_cost_type = "material")
- Equipment - tppss611 (variable proc_ext_cost_type = "equipment")
- Subcontracting - tppss612 (variable proc_ext_cost_type = "subcontracting")

Below you can find an example how to handle the conditions per cost type.

Hook: Declarations

```

table ttpss610
table ttpss611
table ttpss612

extern          string          proc_ext_cost_type(20)

Hook: ext.skip

function extern boolean ext.skip()
{
    on case proc_ext_cost_type
    case "material":
        if <condition on ttpss610 = true> then
            return(true)
        endif
        break
    case "equipment":
        if <condition on ttpss611 = true> then
            return(true)
        endif
        break
    case "subcontracting":
        if <condition on ttpss612 = true> then
            return(true)
        endif
        break
    default:
        break
    endcase

    return (false)
}

```

Process Extensions for ProjectPlannedWarehouseOrder

The following process extension(s) is/are available:

[ProjectPlannedWarehouseOrder.SkipApprove](#)

ProjectPlannedWarehouseOrder.SkipApprove

Skips Planned Warehouse Orders in Project when Approving.

To implement this process extension, you can use the information below:

Usage:	Process Extension ProjectPlannedWarehouseOrder.SkipApprove can be used to skip Planned Warehouse Orders in Project when Approving. Sessions where this Process Extension can be implemented: - Approve Planned PRP Warehouse Orders (tppss6225m000) Fields that are available to be used in this Process Extension: - All fields of table: Planned PRP Warehouse Orders (tppss615)

Process Extensions for ProjectStructuralElement

The following process extension(s) is/are available:

[ProjectStructuralElement.SkipGenerate](#)

ProjectStructuralElement.SkipGenerate

Skips Estimate Structures when Generating Structural Elements for an Estimate Version in Project.

To implement this process extension, you can use the information below:

Usage:	Process Extension ProjectStructuralElement.SkipGenerate can be used to skip certain Estimate Structures when Generating Structural Elements for an Estimate Version in Project. Sessions where this Process Extension can be implemented: - Generate Structural Elements (tpest1220m000) Fields that are available to be used in this Process Extension: - All fields of table: Estimate Structures (tpest110)

Process Extensions for ProjectTransaction

The following process extension(s) is/are available:

[ProjectTransaction.SkipProcess](#)

ProjectTransaction.SkipProcess

Skips Projects when Processing Transactions in Project.

To implement this process extension, you can use the information below:

Usage:	Process Extension ProjectTransaction.SkipProcess can be used to skip certain Projects when Processing Transactions in Project. This applies to the following types of transactions: - Costs - Revenues - Interim Results - Project Completion Sessions where this Process Extension can be implemented: - Process Transactions (tpppc4802m000) Fields that are available to be used in this Process Extension: - All fields of table: Projects (tppdm600)

Process Extensions for PurchaseOrder

The following process extension(s) is/are available:

[PurchaseOrder.SkipApprove](#)

[PurchaseOrder.SkipPrintPurchaseInvoice](#)

[PurchaseOrder.SkipPrintPurchaseOrder](#)

[PurchaseOrder.SkipPrintReceivableInvoice](#)

[PurchaseOrder.SkipReleaseToWarehousing](#)

PurchaseOrder.SkipApprove

Skips Purchase Order when Approving.

To implement this process extension, you can use the information below:

Usage:	Process Extension PurchaseOrder.SkipApprove can be used to skip Purchase Orders when Approving the Purchase Order. Sessions where this Process Extension can be implemented: - Approve Purchase Orders (tdpur4210m100) - All sessions and processes that trigger the Approve. Fields that are available to be used in this Process Extension: - All fields of table Purchase Order (tdpur400). Note: tables must also be declared in the Process Extension. Pseudocode: Below you can find an example. Hook: Declarations <pre> table ttdpur400 </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { if <condition on tdpur400 = true> then return(true) endif return (false) } </pre>

PurchaseOrder.SkipPrintPurchaseInvoice

Skips Purchase Order when Printing Purchase Invoices.

To implement this process extension, you can use the information below:

Usage:	Process Extension PurchaseOrder.SkipPrintPurchaseInvoice can be used to skip specific Purchase Orders when printing the Purchase Order Invoice. Sessions where this Process Extension can be implemented: - Print Purchase Invoices (tdpur4404m000) - All sessions and processes that trigger the printing of the Purchase Invoices (like automatic processing logic). Fields that are available to be used in this Process Extension: - All fields of table Purchase Orders (tdpur400) Pseudocode:

	Below you can find an example: Hook: Declarations <pre> table ttdpur400 Hook: ext.skip function extern boolean ext.skip() { if <condition on tdpur400 = true> then return(true) endif return (false) } </pre>

PurchaseOrder.SkipPrintPurchaseOrder

Skips Purchase Order when Printing Purchase Order.

To implement this process extension, you can use the information below:

Usage:	Process Extension PurchaseOrder.SkipPrintPurchaseOrder can be used to skip Purchase Orders when printing the Purchase Order. Sessions where this Process Extension can be implemented: - Print Purchase Orders (tdpur4401m000) - All sessions and processes that trigger the printing of the Purchase Order (like automatic processing logic). Fields that are available to be used in this Process Extension: - All fields of table Purchase Order (tdpur400) Pseudocode: Below you can find an example: Hook: Declarations <pre> table ttdpur400 Hook: ext.skip function extern boolean ext.skip() { if <condition on tdpur400 = true> then return(true) endif return (false) } </pre>

PurchaseOrder.SkipPrintReceivableInvoice

Skips Purchase Order when Printing Receivable Invoices.

This process extension is available from [KB2327957](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension PurchaseOrder.SkipPrintReceivableInvoice can be used to skip Purchase Orders when printing receivable invoices. Sessions where this Process Extension can be implemented: - Print Receivable Invoices Up to Period (tfacpl439m000) - Checklist Reconciliation Goods Received Not Invoiced (tfgld4495m200) Note: For the same sessions, ReceivableInvoice.SkipPrintSpecification() is available to skip printing based on "Order Data for Approval" (tfacp240) Fields that are available to be used in this Process Extension: - All fields of table Purchase Order History (tdpur450) Pseudocode: Below you can find an example: Hook: Declarations <pre> table ttdpur450 </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { if <condition on tdpur450 = true> then return(true) endif return (false) } </pre>

PurchaseOrder.SkipReleaseToWarehousing

Skips Purchase Order Lines when Releasing to Warehousing.

To implement this process extension, you can use the information below:

Usage:	Process Extension PurchaseOrder.SkipReleaseToWarehousing can be used to skip Purchase Order Lines when Releasing to Warehousing. Sessions where this Process Extension can be implemented: - Release Purchase Orders to Warehousing (tdpur4246m000) - All sessions and processes that trigger the release of lines to Warehousing (like automatic processing logic). Fields that are available to be used in this Process Extension: - All fields of table Purchase Order Lines (tdpur401) when releasing

	<pre> order lines. Note: tables must also be declared in the Process Extension. The Purchase order release to warehousing process releases Purchase order lines. Pseudocode: Below you can find an example. Hook: Declarations table ttdpur401 Hook: ext.skip function extern boolean ext.skip() { if <condition on tdpur401 = true> then return(true) endif return (false) } </pre>

Process Extensions for PurchaseOrderGenerate

The following process extension(s) is/are available:

[PurchaseOrderGenerate.HandleCombining](#)

[PurchaseOrderGenerate.SuppressCommingleLine](#)

PurchaseOrderGenerate.HandleCombining

Customize the logic with regards to combining new purchase order lines to existing purchase orders.

This process extension is available from [KB2326972](#).

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to customize the logic with regards to combining new purchase order lines to existing purchase orders. The extension provides hooks for: - Get a customer determined purchase order that will be used when

	generating purchase order lines. - Suppress combining purchase order lines on an existing purchase order during generation. These hooks are called for generate processes that can create a new purchase order header.

To implement this process extension, you need to implement the following method(s):

tdext.pur0003.determine.purchase.order.for.combining.order.lines.during.generation

Syntax:	<pre>long tdext.pur0003.determine.purchase.order.for.combining.order.lines.during.generation (domain tcorno i.purchase.order, domain tcpono i.purchase.order.line, domain tdpur.corg i.purchase.order.origin, long i.source.generating.process, ref domain tcorno o.purchase.order)</pre>
Usage:	Expl: Use this method to get a customer determined purchase order that will be used when generating purchase order lines. If this extension returns a purchase order, then that purchase order will be used to add the new purchase order line to. Note that after calling this method, LN will also consider the method tdext.pur0003.suppress.combining.of.purchase.order.lines.to.existing.order.during.generation If implemented, that extension could prevent combining the order line. If the extension returns a purchase order that is closed or cancelled, then LN will create a new order. The same happens if attributes on the returned purchase order do not match with the attributes that are actually required. If the extension returns a purchase order that is not updatable, then an error is returned during generation. This extension is implemented for multiple purchase order generation processes. See below. Input: - i.purchase.order - Purchase Order If filled, this refers to the purchase order that would be used by the standard logic. If empty, LN standard logic would generate a new purchase order. - i.purchase.order.line - Purchase Order Line (Optional) If filled, this refers to the purchase order line that would be used by standard logic; LN would create a new detail-line. - i.purchase.order.origin - The Purchase Order Origin from / for which the purchase order is generated.

	<code>i.source.generating.process</code> - Indicates the source process from / for which the purchase order is generated and where an order header can be created. Possible Values: <table> <tr> <td>0</td><td>Undefined - Use <code>i.purchase.order.origin</code> to determine the source process</td></tr> <tr> <td>1</td><td>Sales Order Components (Kitting)</td></tr> <tr> <td>2</td><td>Return Rejected</td></tr> </table> The following generation sources can support creation of multiple order headers and can be affected by the process extension. These processes are either identified by the origin (<code>i.purchase.order.origin</code>) or by the value of <code>i.source.generating.process</code>. <table> <tr> <th>Source Process</th><th>Value of <code>i.purchase.order.origin</code></th></tr> <tr> <td>- Unexpected Warehouse Receipts</td><td><code>tdpur.corg.wh.receipt</code></td></tr> <tr> <td>- RFQ</td><td><code>tdpur.corg.inquiries</code></td></tr> <tr> <td>- Subcontracting Purchase Order</td><td><code>tdpur.corg.subc.pur.order</code></td></tr> <tr> <td>- Subcontracting Purchase Schedule</td><td><code>tdpur.corg.subc.pur.sched</code></td></tr> <tr> <td>- Production</td><td><code>tdpur.corg.sfc</code></td></tr> <tr> <td>- Planning</td><td><code>tdpur.corg.mrp</code></td></tr> <tr> <td>- Purchase Payment (Pay on Use)</td><td><code>tdpur.corg.payment</code></td></tr> <tr> <td>- Price Calculation</td><td><code>tdpur.corg.price.calc</code></td></tr> <tr> <td>- Sales</td><td><code>tdpur.corg.sls</code></td></tr> <tr> <td>- Warehousing</td><td><code>tdpur.corg.inv</code></td></tr> <tr> <td>- Contract Delivery Scheme</td><td><code>tdpur.corg.contracts</code></td></tr> <tr> <td>- Service Material Cost Lines</td><td><code>tdpur.corg.sma</code></td></tr> <tr> <td>- Maintenance Work Order</td><td><code>tdpur.corg.wcs</code></td></tr> </table> <table> <tr> <th>Source Process</th><th>Value of <code>i.source.generating.process</code></th></tr> <tr> <td>- Sales Order Components (Kitting)</td><td>1</td></tr> <tr> <td>- Return Rejected</td><td>2</td></tr> </table> Other generation sources can support creation of order headers but cannot be affected by the process extension and will not invoke the process extension because: - LN standard creates one order and line (hard coded) - Multiple orders may cause conflicts with current standard behavior or maybe unwanted. These are: - Price Calculator - Assembly - List items - Net Prices per Buy-from Business Partner - Copy BOM - Project (PRP Orders) - Service Customer Claim - Requisition - Orders generated through external integrations: BOD/BDE/... Output: <code>o.purchase.order</code> - The purchase Order as determined by the extension. * non-empty:	0	Undefined - Use <code>i.purchase.order.origin</code> to determine the source process	1	Sales Order Components (Kitting)	2	Return Rejected	Source Process	Value of <code>i.purchase.order.origin</code>	- Unexpected Warehouse Receipts	<code>tdpur.corg.wh.receipt</code>	- RFQ	<code>tdpur.corg.inquiries</code>	- Subcontracting Purchase Order	<code>tdpur.corg.subc.pur.order</code>	- Subcontracting Purchase Schedule	<code>tdpur.corg.subc.pur.sched</code>	- Production	<code>tdpur.corg.sfc</code>	- Planning	<code>tdpur.corg.mrp</code>	- Purchase Payment (Pay on Use)	<code>tdpur.corg.payment</code>	- Price Calculation	<code>tdpur.corg.price.calc</code>	- Sales	<code>tdpur.corg.sls</code>	- Warehousing	<code>tdpur.corg.inv</code>	- Contract Delivery Scheme	<code>tdpur.corg.contracts</code>	- Service Material Cost Lines	<code>tdpur.corg.sma</code>	- Maintenance Work Order	<code>tdpur.corg.wcs</code>	Source Process	Value of <code>i.source.generating.process</code>	- Sales Order Components (Kitting)	1	- Return Rejected	2
0	Undefined - Use <code>i.purchase.order.origin</code> to determine the source process																																								
1	Sales Order Components (Kitting)																																								
2	Return Rejected																																								
Source Process	Value of <code>i.purchase.order.origin</code>																																								
- Unexpected Warehouse Receipts	<code>tdpur.corg.wh.receipt</code>																																								
- RFQ	<code>tdpur.corg.inquiries</code>																																								
- Subcontracting Purchase Order	<code>tdpur.corg.subc.pur.order</code>																																								
- Subcontracting Purchase Schedule	<code>tdpur.corg.subc.pur.sched</code>																																								
- Production	<code>tdpur.corg.sfc</code>																																								
- Planning	<code>tdpur.corg.mrp</code>																																								
- Purchase Payment (Pay on Use)	<code>tdpur.corg.payment</code>																																								
- Price Calculation	<code>tdpur.corg.price.calc</code>																																								
- Sales	<code>tdpur.corg.sls</code>																																								
- Warehousing	<code>tdpur.corg.inv</code>																																								
- Contract Delivery Scheme	<code>tdpur.corg.contracts</code>																																								
- Service Material Cost Lines	<code>tdpur.corg.sma</code>																																								
- Maintenance Work Order	<code>tdpur.corg.wcs</code>																																								
Source Process	Value of <code>i.source.generating.process</code>																																								
- Sales Order Components (Kitting)	1																																								
- Return Rejected	2																																								

	The purchase order line will be added to the given purchase order. if the given purchase order does not exist, then LN will create a new purchase order. * empty: Standard logic from LN will be used to decide which existing purchase order (if any) will be used to add the purchase order line to. Return: 0 <> 0 - Success - When an error occurs in the determination of the purchase order. The standard logic for combining will be applied.

tdext.pur0003.suppress.combining.of.purchase.order.lines.to.existing.order.during.generation

Syntax:	<pre>boolean tdext.pur0003.suppress.combining.of.purchase.order.lines.to.existing.order.during.generation(domain tcorno i.purchase.order, domain tcpono i.purchase.order.line, domain tdpur.corg i.purchase.order.origin, long i.source.generating.process)</pre>
Usage:	Expl.: Use this method to suppress combining purchase order lines on the same purchase order during generation. The standard logic in LN will first determine whether combining is allowed at all. If so, this method will then be called to decide whether combining must be suppressed. When this method is called, the following applies: - i.purchase.order is always filled; - i.purchase.order.line can be zero or > zero - A record of tdpur400 is current. Input: i.purchase.order - Purchase Order This refers to the purchase order that would be used by the standard logic. i.purchase.order.line - Purchase Order Line Note that the sequence cannot be input for the order generation process. i.purchase.order.origin - The Purchase Order Origin from / for which the purchase order is generated. i.source.generating.process - Indicates the source process from / for which the purchase order is generated and where an order header can be created. Possible Values: 0 Undefined - Use

		i.purchase.order.origin to determine the source process
1		Sales Order Components (Kitting)
2		Return Rejected
The following generation sources can support creation of multiple order headers and can be affected by the process extension. These processes are either identified by the origin (i.purchase.order.origin) or by the value of i.source.generating.process.		
Source Process		Value of i.purchase.order.origin
- Unexpected Warehouse Receipts		tdpur.corg.wh.receipt
- RFQ		tdpur.corg.inquiries
- Subcontracting Purchase Order		tdpur.corg.subc.pur.order
- Subcontracting Purchase Schedule		tdpur.corg.subc.pur.sched
- Production		tdpur.corg.sfc
- Planning		tdpur.corg.mrp
- Purchase Payment (Pay on Use)		tdpur.corg.payment
- Price Calculation		tdpur.corg.price.calc
- Sales		tdpur.corg.sls
- Warehousing		tdpur.corg.inv
- Contract Delivery Scheme		tdpur.corg.contracts
- Service Material Cost Lines		tdpur.corg.sma
- Maintenance Work Order		tdpur.corg.wcs
Source Process		Value of i.source.generating.process
- Sales Order Components (Kitting)		1
- Return Rejected		2
Other generation sources can support creation of order headers but cannot be affected by the process extension and will not invoke the process extension because: - LN standard creates one order and line (hard coded) - Multiple orders may cause conflicts with current standard behavior or maybe unwanted.		
These are: - Price Calculator - Assembly - List items - Net Prices per Buy-from Business Partner - Copy BOM - Project (PRP Orders) - Service Customer Claim - Requisition - Orders generated through external integrations: BOD/BDE/...		
Output: Not Applicable. Return: True - The extension has determined that combining must be suppressed for the given input.		
False - The standard logic for combining applies.		

PurchaseOrderGenerate.SuppressCommingleLine

Suppress commingling of purchase order lines when generating them.

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to suppress commingling to an existing purchase order line during generation of a new purchase order line. This Process Extension is used in all flows where a purchase order line is generated.

To implement this process extension, you need to implement the following method(s):

tdext.pur0002.suppress.commingling.of.purchase.order.line.during.generation

Syntax:	<pre>boolean tdext.pur0002.suppress.commingling.of.purchase.order.line.during.generation(domain tcorno i.previous.order, domain tcpono i.previous.line, domain tcpono i.previous.sequence, domain tdpur.corg i.origin, domain tcorno i.originating.order, domain tcpono i.originating.line, domain tcpono i.originating.sequence, domain tccom.bpid i.buy.from.business.partner, domain tccom.bpid i.ship.from.business.partner, domain tcitem i.item, domain tcuef.effn i.effectivity.unit, domain tdobid i.product.variant, domain tcyesno i.subcontracted, domain tccwar i.warehouse)</pre>								
Usage:	Expl.: This method can be implemented to suppress commingling during purchase order line generation. The standard logic in LN will first determine whether commingling is allowed at all. If so, this process extension will then be called to decide whether commingling must be suppressed. Input: <table border="0"> <tr> <td>i.previous.order</td><td>The previous purchase order that was generated in this process. If commingling is allowed then LN will commingle the new purchase order line to this order.</td></tr> <tr> <td>i.previous.line</td><td>The previous purchase order line that was generated in this process. If commingling is allowed and it is not suppressed, then LN will commingle the new purchase order line to this order line.</td></tr> <tr> <td>i.previous.sequence</td><td>The previous purchase order sequence that was generated in this process.</td></tr> <tr> <td>i.origin</td><td>The origin of the new order line. Possible values: tdpur.corg.mrp (EP) tdpur.corg.wcs (Maintenance)</td></tr> </table>	i.previous.order	The previous purchase order that was generated in this process. If commingling is allowed then LN will commingle the new purchase order line to this order.	i.previous.line	The previous purchase order line that was generated in this process. If commingling is allowed and it is not suppressed, then LN will commingle the new purchase order line to this order line.	i.previous.sequence	The previous purchase order sequence that was generated in this process.	i.origin	The origin of the new order line. Possible values: tdpur.corg.mrp (EP) tdpur.corg.wcs (Maintenance)
i.previous.order	The previous purchase order that was generated in this process. If commingling is allowed then LN will commingle the new purchase order line to this order.								
i.previous.line	The previous purchase order line that was generated in this process. If commingling is allowed and it is not suppressed, then LN will commingle the new purchase order line to this order line.								
i.previous.sequence	The previous purchase order sequence that was generated in this process.								
i.origin	The origin of the new order line. Possible values: tdpur.corg.mrp (EP) tdpur.corg.wcs (Maintenance)								

		tdpur.corg.inv (Warehousing) tdpur.corg.contracts (Contract) tdpur.corg.inquiries (RFQ) tdpur.corg.eop (EDI) tdpur.corg.manual (Manual) tdpur.corg.sfc (Job Shop Control) tdpur.corg.project (Project) tdpur.corg.sls (Sales) tdpur.corg.sma (Service) tdpur.corg.pmg (Process) tdpur.corg.requisition (Requisition) tdpur.corg.asc (Assembly) tdpur.corg.extern (External) tdpur.corg.wh.receipt (Warehousing Receipt) tdpur.corg.payment (Purchase Payment) tdpur.corg.subc.pur.order (Subcontracting
Purchase Order)		
Claim)		tdpur.corg.serv.cust.claim (Service Customer
	i.originating.order	tdpur.corg.routing (Routing) tdpur.corg.price.calc (Price Calculation) The originating order. This field is guaranteed to be filled in case of the following values for i.origin: * tdpur.corg.sls (Sales) * tdpur.corg.inquiries (RFQ's) * tdpur.corg.requisition (Requisition) * tdpur.corg.subc.pur.order (Subcontracting Purchase Order) * tdpur.corg.price.calc (Price Calculation) For other origins, the originating order may be filled or not, depending on the scenario.
	i.originating.line	Originating line. May be filled or not. See 'i.originating.order'.
	i.originating.sequence	Originating sequence May be filled or not. See 'i.originating.order'.
	i.buy.from.business.partner	Buy-from business partner.
	i.ship.from.business.partner	Ship-from business partner.
	i.item	Item
	i.effectivity.unit	Effectivity Unit
	i.product.variant	Product Variant
	i.subcontracted	Subcontracted (Yes/No)
	i.warehouse	Warehouse
	Output: N.A.	
	Return: true	The extension has determined that commingling must be suppressed for the given input.
	false	The standard logic for commingling applies.

Process Extensions for PurchaseOrderLine

The following process extension(s) is/are available:

[PurchaseOrderLine.SkipPrintPurchaseInvoice](#)[PurchaseOrderLine.SkipPrintPurchaseOrder](#)[PurchaseOrderLine.SkipPrintPurchaseOrderReminder](#)[PurchaseOrderLine.SkipPrintReceivableInvoice](#)

PurchaseOrderLine.SkipPrintPurchaseInvoice

Skips Purchase Order Line when Printing Purchase Invoices.

To implement this process extension, you can use the information below:

Usage:	Process Extension <code>PurchaseOrderLine.SkipPrintPurchaseInvoice</code> can be used to skip specific Purchase Order Lines when printing Purchase Invoices. Sessions where this Process Extension can be implemented: - Print Purchase Invoices (tdpur4404m000) - All sessions and processes that trigger the printing of the Purchase Invoices (like automatic processing logic). Fields that are available to be used in this Process Extension: - All fields of table Purchase Orders (tdpur400) - All fields of table Purchase Order Lines (tdpur401) Pseudocode: Below you can find an example: Hook: Declarations <pre>table ttdpur401</pre> Hook: ext.skip <pre>function extern boolean ext.skip() { if <condition on tdpur401 = true> then return(true) endif return (false) }</pre>

PurchaseOrderLine.SkipPrintPurchaseOrder

Skips Purchase Order Line when Printing Purchase Order.

To implement this process extension, you can use the information below:

Usage:	Process Extension <code>PurchaseOrderLine.SkipPrintPurchaseOrder</code> can be used to skip specific Purchase Order Lines when printing the Purchase Order. Sessions where this Process Extension can be implemented: - Print Purchase Orders (tdpur4401m000) - All sessions and processes that trigger the printing of the Purchase Order (like automatic processing logic). Fields that are available to be used in this Process Extension: - All fields of table Purchase Order (tdpur400) - All fields of table Purchase Order Lines (tdpur401) Pseudocode: Below you can find an example: Hook: Declarations <pre> table ttdpur401 </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { if <condition on tdpur401 = true> then return(true) endif return (false) } </pre>

PurchaseOrderLine.SkipPrintPurchaseOrderReminder

Skips Purchase Order Line when Printing Purchase Order Reminders.

To implement this process extension, you can use the information below:

Usage:	Process Extension <code>PurchaseOrderLine.SkipPrintPurchaseOrderReminder</code> can be used to skip specific Purchase Order Lines when printing Purchase Order Reminders. Sessions where this Process Extension can be implemented: - Print Purchase Order Reminders (tdpur4403m000) Fields that are available to be used in this Process Extension: - All fields of table Purchase Order Lines (tdpur401) Pseudocode: Below you can find an example: Hook: Declarations <pre> table ttdpur401 </pre> Hook: ext.skip <pre> function extern boolean ext.skip() </pre>

	<pre> { if <condition on tdpur401 = true> then return(true) endif return (false) } </pre>

PurchaseOrderLine.SkipPrintReceivableInvoice

Skips Purchase Order Line when Printing Receivable Invoices.

This process extension is available from [KB2327957](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension PurchaseOrderLine.SkipPrintReceivableInvoice can be used to skip Purchase Order Lines when printing receivable invoices. Sessions where this Process Extension can be implemented: - Print Receivable Invoices Up to Period (tfacp1439m000) - Checklist Reconciliation Goods Received Not Invoiced (tfgld4495m200) Note: For the same sessions, ReceivableInvoice.SkipPrintSpecification() is available to skip printing based on "Order Data for Approval" (tfacp240) Fields that are available to be used in this Process Extension: - All fields of table Purchase Orders History (tdpur450) - All fields of table Purchase Order Lines History (tdpur451) Pseudocode: Below you can find an example: Hook: Declarations <pre> table ttdpur451 </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { if <condition on tdpur451 = true> then return(true) endif return (false) } </pre>

Process Extensions for PurchaseSelfBilledInvoice

The following process extension(s) is/are available:

[PurchaseSelfBilledInvoice.CustomCompose](#)

PurchaseSelfBilledInvoice.CustomCompose

Define a custom composing order for Purchase Self-Billing.

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to set an own custom compose criteria in the Self-Billing Method. The Purchase Self-Billing (tfacp2290m000) process will compose the invoices based on the defined custom composing criteria.

To implement this process extension, you need to implement the following method(s):

tfext.acp0001.set.custom.composing.fields

Syntax:	<pre>long tfext.acp0001.set.custom.composing.fields(ref string o.composing.fields())</pre>
Usage:	Expl: Use this method in order to use a Custom Compose Criteria for the Self-Billing Method. If an implementation of this method is done, and the string with composing fields is filled, and this method is returning 0, then in the Self-Billing Method (tcmcs0157m000) session the Compose Criteria (tcmcs057.cmpc) field can be set as Custom Compose. Via this method, one or more fields can be defined in a (comma separated) string. Those defined fields will be used as composing criteria for creating the Purchase Self-Billed Invoice in the Purchase Self-Billing (tfacp2290m000) process. In the Custom Compose flow, the standard Self-Billing will use the below pre-defined fields already for composing the Purchase Self-Billed Invoice (as these are invoice header attributes): <pre>tfacp240.ifbp, tfacp240.otbp, tfacp240.ccur, tfacp240.vatc, tfacp240.ptyp, tfacp240.rtyp, tfacp240.mcfr</pre>

	So, via this method, EXTRA composing attributes can be defined. Note that only attributes of below tables may be used: - Order Data for Approval (tfacp240) - Purchase Consumptions (tfacp249) Also Customer Defined Fields (CDF), added to the above mentioned tables (tfacp240 and tfacp249), can be used as extra composing fields. EXAMPLE: If the composing must also be done on Receipt (tfacp249.rcno) and an own CDF field (tfacp249.cdf_0001), this method should return: <pre>o.composing.fields = "tfacp249.rcno, tfacp249.cdf_0001" return(0)</pre> Pre: N.A. Post: N.A. Input: N.A. Output: o.composing.fields - String with composing composing fields (comma separated) Maximum string length is 500 (sb). Return: 0 - Success DALHOOKERROR - When an error occurs in setting the composing fields.

Process Extensions for QualityFailureDocument

The following process extension(s) is/are available:

[QualityFailureDocument.SkipPrint](#)

QualityFailureDocument.SkipPrint

Skips Printing of FRACAS.

To implement this process extension, you can use the information below:

Usage:	Process Extension QualityFailureDocument.SkipPrint can be used to skip certain FRACAS Reports when Printing Failure Document Report. Sessions where this Process Extension can be implemented: - Print Failure Report Analysis and Corrective Action System (qmpqm2400m000) Fields that are available to be used in this Process Extension: - All fields of table: Failure Report Analysis and Corrective Action System

	(qmpqm200) Failure and Action Details (qmpqm201)

Process Extensions for QualityManagement

The following process extension(s) is/are available:

[QualityManagement.DefaultStandardTestProcedure](#)

QualityManagement.DefaultStandardTestProcedure

Determine customer specific default Standard Test Procedure for Order Inspection Creation.

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to set a customer determined standard test procedure from an extension to a value other than the defaulted value based on LN logic . This standard test procedure (if valid) is then used in all flows where a standard test procedure is defaulted.i.e., Order Inspections Creation.

To implement this process extension, you need to implement the following method(s):

qmext.ptc0001.get.customer.determined.default.standard.test.procedure

Syntax:	<pre>long qmext.ptc0001.get.customer.determined.default.standard.test.procedure(domain qmptc.orgn i.origin, domain tcitem i.item, domain tcorno i.order, domain tcpono i.line, domain tccom.bpid i.business.partner, domain tcitem i.sub.item, domain tcedm.revi i.revision, domain tcuef.effn i.effectivity.unit, domain tcedm.revi i.sub.item.revision, domain tcuef.effn i.sub.item.effectivity.unit, domain tirou.opro i.routing, domain tcopno i.operation, domain tctano i.task, domain tcsite i.from.site, domain tcsite i.to.site,</pre>

	<pre> domain tccwar i.warehouse.from, domain tccwar i.warehouse.to, domain qmrpt.wstt i.work.station, domain qmptc.quid i.stp.from.stnd.logic, domain qmptc.srno i.version.from.stnd.logic, ref domain qmptc.quid o.standard.test.procedure, ref domain qmptc.srno o.version) </pre>
Usage:	Expl: Use this method to get a customer determined default standard test procedure from an extension. This standard test procedure will be used in creation of Order Inspection in Quality Management. The standard logic in LN determines the standard test procedure for the given set of data. Using this process extension, the defaulted value can be changed to customer given value. Note: 1. The input arguments can be used to determine the option. 2. LN will check if the provided default is valid. 3. The standard test procedure defaulted based on standard LN logic will overrule the selected standard test procedure from the extension when it is not valid or not given, otherwise standard logic will fail for creation on QM Order Inspection. Pre: NA Post: NA Input: <pre> i.origin -> Origin i.item -> Item i.business.partner -> Business Partner (For Sales/Purchase) i.order -> order i.line -> Line (For Material (BOM)) i.sub.item -> Sub Item (For Material (BOM)) i.routing -> Routing (For Routing) i.operation -> Operation (For Routing) i.task -> Task (For origin Routing) i.work.station -> Work Station (for origin Routing (RPT)) i.warehouse.from -> Source warehouse (For Warehouse Transfer) i.warehouse.to -> To Warehouse (For Warehouse Transfer) i.work.station -> Work Station i.stp.from.stnd.logic -> As defaulted under the standard logic i.version.from.stnd.logic -> As defaulted under the standard logic </pre> Output: o.standard.test.procedure - The standard test procedure determined by the extension. Return: <pre> 0 - Success DALHOOKERROR - Otherwise. </pre>

Process Extensions for QualityNonConformance

The following process extension(s) is/are available:

[QualityNonConformance.SkipPrint](#)

QualityNonConformance.SkipPrint

Skips Printing of Non-Conformance Reports.

To implement this process extension, you can use the information below:

Usage:	Process Extension QualityNonConformance.SkipPrint can be used to skip certain Non-Conformance (Either Material or Non-Material) Reports when Printing Non-Conformance Report. Sessions where this Process Extension can be implemented: - Print Non-Conformance Reports (qmncml400m000) Fields that are available to be used in this Process Extension: - All fields of table: Non-Conformance Report (NCR) (qmncml100)

Process Extensions for ReceivableInvoice

The following process extension(s) is/are available:

[ReceivableInvoice.SkipPrintSpecification](#)

ReceivableInvoice.SkipPrintSpecification

Skip Print Receivable Invoice Specification.

This process extension is available from [KB2329986](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension ReceivableInvoice.SkipPrintSpecification can be used to skip Printing Receivable Invoice Specification. Sessions where this Process Extension can be implemented: - Print Receivable Invoice Specifications (tfacp1432m000). - Checklist Reconciliation Goods Received Not Invoiced (tfgld4495m200) - Print Receivable Invoices Up to Period (tfacp1439m000) Note: For sessions tfgld4495m200 and tfacp1439m000, PurchaseOrder.SkipPrintReceivableInvoice is available to skip printing based on table: Purchase Order History (tdpur450) Fields that are available to be used in this Process Extension: - All fields of table: Order Data for Approval (tfacp240) External variables that are available to be used in the Process Extension: - proc ext called from [type: string(13)] : this variable is filled

	with one of next values: - "tfacp1432m000" - "tfgld4495m200" - "tfacp1439m000" Note: Tables and external variables must also be declared in the Process Extension. Pseudocode: Below you can find an example: Hook: Declarations <pre> table tfacp240 * Order Data for Approval extern string proc_ext_called_from(13) </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { if proc_ext_called_from = "tfacp1439m000" and <condition on tfacp240 = true> then return(true) endif return (false) } </pre>

Process Extensions for ReceivedCustomerOrderProcess

The following process extension(s) is/are available:

[ReceivedCustomerOrderProcess.Validate](#)

ReceivedCustomerOrderProcess.Validate

Validate received customer order during processing.

This process extension is available from [KB2323033](#).

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to implement additional validation checks when processing Received Customer Orders.

To implement this process extension, you need to implement the following method(s):

tdext.sls0005.validate.received.customer.order.during.processing

Syntax:	<pre> long tdext.sls0005.validate.received.customer.order.during.processing(domain tcorno i.received.customer.order, domain tdsls.obty i.received.customer.order.document.type, domain tdwcom i.received.customer.order.origin, domain tcproc.actn i.process.action, domain tcyesno i.change.order, domain tcmcs.st30m i.customer.order.number mb, domain tccom.bpid i.sold.to.bp, domain tccwoc i.sales.office) </pre>
Usage:	Expl: This function executes customer defined validations during processing the given Received Customer Order. To indicate that the validation failed, the extension must put an error-message on the stack and return a non-zero value. In that case, the error will be logged in the message log, and the system continues with the next Received Customer Order (if any). Pre: Not Applicable. Post: Not Applicable. Input: i.received.customer.order - Received Customer Order - i.received.customer.order.document.type - Received Customer Order Document Type. - Possible value(s): * tdsls.obty.order (Sales Order) - i.received.customer.order.origin - Received Customer Order Origin. - Possible values: * tdwcom.edi (EDI) * tdwcom.bod (BOD) * tdwcom.manual (Manual) - i.process.action - Process Action. - Possible values: * tcproc.actn.create (Create) * tcproc.actn.update (Update) * tcproc.actn.cancel (Cancel) * tcproc.actn.not.applicable (Not Applicable) - i.change.order - Change Order (Yes/No) - i.customer.order.number - Customer Order - i.sold.to.bp - Sold-to Business Partner (Mandatory) - i.sales.office - Sales Office Output: Not Applicable Return: 0 - Validation was successful. - The extension indicates that the given Received Customer Order is not valid. An error message has been put on the stack by the extension. LN will skip this Received Customer Order.

Process Extensions for RequestForQuoteConvert

The following process extension(s) is/are available:

[RequestForQuoteConvert.CustomSorting](#)

RequestForQuoteConvert.CustomSorting

Define a customer specific sorting for Converting RFQs.

This process extension is available from [KB2328648](#).

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to set a customer defined sorting from an extension. This sorting will be used in 'Convert RFQs' (tdpur1202m000).

To implement this process extension, you need to implement the following method(s):

tdext.pur0004.get.order.by.for.convert

Syntax:	<code>long tdext.pur0004.get.order.by.for.convert(ref string o.order.by.string.mb())</code>
Usage:	Expl: Use this method to define a customized ordering of RFQ responses. This order-by clause will be used in session 'Convert RFQs' (tdpur1202m000). But note that this Process Extension will not be considered in scenarios where the quantity is to be split among the bidders (Setting on the ConvertRFQs session). The standard order by clause that is used is: " order by tdpur106._index2 with retry " When implementing this method, an alternative ordering could be, as an example: " order by tdpur106.cdf_0001 desc, tdpur106.otbp asc " Notes: 1. Only attributes from tables tdpur100, tdpur105 and tdpur106 can be used, including customer defined fields. 2. Run time errors occur when specifying attributes from other tables or when wrong syntax is constructed. 3. It is advised to include 'with retry' at the end of the order-by string. This will make sure that only part of the query is repeated after a database update resulted in a retry. Pre: NA Post: NA Input: NA

	Output: o.order.by.string.mb - Maximum string length is 500. Multi byte attributes are allowed. Return: 0 - Success <> 0 - When an error occurs in determination of the order by clause.

Process Extensions for ReservedApprovedCommissionsRebates

The following process extension(s) is/are available:

[ReservedApprovedCommissionsRebates.SkipPrint](#)

ReservedApprovedCommissionsRebates.SkipPrint

Skips Reserved and Approved Commissions/Rebates when Printing.

This process extension is available from [KB2302057](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension ReservedApprovedCommissionsRebates.SkipPrint can be used to skip Reserved and Approved Commissions/Rebates when Printing the Reserved Commissions/Rebates and Approved Commissions/Rebates. Session where this Process Extension can be implemented: - Print Approved Commissions/Rebates (tdcms2402m000) - Print Reserved Commissions/Rebates (tdcms2403m000) Fields that are available to be used in this Process Extension: - All fields of table Commissions/Rebates (tdcms050) Note: Table must also be declared in the Process Extension. So skip conditions can be built on current tdcms050 data as instructed below. Pseudocode: Here you can find an example how to handle the conditions for reserved commissions/rebates and approved commissions/rebates. Hook: Declarations <pre>table tdcms050</pre> Hook: ext.skip <pre>function extern boolean ext.skip() {</pre>

	<pre> on case tdcms050.resv case tcyesno.yes: if <condition on tdcms050 = true> then return(true) endif So in case of field Reserve (tdcms050.resv) = "Yes", Reserved Commissions/Rebates can be skipped based on tdcms050 data. break case tcyesno.no: if <condition on tdcms050 = true> then return(true) endif So in case of field Reserve (tdcms050.resv) = "No", Approved Commissions/Rebates can be skipped based on tdcms050 data. break endcase return (false) } </pre>

Process Extensions for ReservedCommissionsRebates

The following process extension(s) is/are available:

[ReservedCommissionsRebates.SkipReservationApproval](#)

ReservedCommissionsRebates.SkipReservationApproval

Skips Reserved Commissions/Rebates when Reserving and Approving.

This process extension is available from [KB3598645](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension ReservedCommissionsRebates.SkipReservationApproval can be used to skip Reserved Commissions/Rebates when Reserving and Approving the Reserved Commissions/Rebates. Session where this Process Extension can be implemented: - Reservation and Approval of Reserved Commissions/Rebates (tdcms2202m000)

	Fields that are available to be used in this Process Extension: - All fields of table Commissions/Rebates (tdcms050) Note: Table must also be declared in the Process Extension. So skip conditions can be built on current tdcms050 data as instructed below. Pseudocode: Here you can find an example how to handle the conditions for reserved commissions/rebates. Hook: Declarations <pre> table tdcms050 </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { if <condition on tdcms050 = true> then return(true) endif return (false) } </pre>

Process Extensions for Sales

The following process extension(s) is/are available:

[Sales.SkipRetroBilledAdviceLineGeneration](#)

Sales.SkipRetroBilledAdviceLineGeneration

Skips Delivery Lines when Generating Advice Lines.

This process extension is available from [KB3626008](#).

To implement this process extension, you can use the information below:

Usage:	The Sales.SkipRetroBilledAdviceLineGeneration process extension can be used to skip Delivery Lines during the generation of retro-billed advice lines. Session where this Process Extension can be implemented: - Generate Retro-Billed Price Change Advice (tdsls3270d000) - Generate Retro-Billed Price Change Advice (tdsls3270m100)

External variable that is available to be used in this Process Extension:

- proc_ext_skip_sales_advice_line_generation_based_on [domain: tctabl.c]

Supported values are:

- "tdsls357"
- "tdsls456"
- "tdsls340"
- "tdsls406"

The fields accessible within this process extension depend on the context, which is determined by the external variable: "proc_ext_skip_sales_advice_line_generation_based_on".

Accessible fields:

(Please note that in some contexts, all fields are accessible, while in other contexts, only the primary key fields are accessible).

- "tdsls357" - All fields of table "Sales Schedule Actual Delivery Lines History" (tdsls357)
- "tdsls456" - All fields of table "Sales Order Actual Delivery Lines History" (tdsls456)
- "tdsls340" - Primary key fields of table "Sales Schedule Actual Delivery Lines" (tdsls340)
- "tdsls406" - Primary key fields of table "Sales Order Actual Delivery Lines" (tdsls406)

Note: tables and external variable must also be declared in the Process Extension.

Pseudocode:

Below you can find an example how to handle the conditions for the various contexts when LN is in the process of generating retro-billed advice lines.

Hook: Declarations

```
table    ttdsls357      |* Sales Schedule Actual Delivery Lines History
table    ttdsls456      |* Sales Order Actual Delivery Lines History
table    ttdsls340      |* Sales Schedule Actual Delivery Lines
table    ttdsls406      |* Sales Order Actual Delivery Lines
```

```
extern domain tctabl.c
```

```
proc_ext_skip_sales_advice_line_generation_based_on
```

```
Hook: ext.skip
```

```
function extern boolean ext.skip()
{
    on case proc_ext_skip_sales_advice_line_generation_based_on
    case "tdsls357":
        if <condition on tdsls357 = true> then
            return(true)
        endif
        break
    case "tdsls456":
        if <condition on tdsls456 = true> then
            return(true)
        endif
        break
    case "tdsls340":
        read data using:
```

	<pre> - tdsls340.schn - tdsls340.sctp - tdsls340.revn - tdsls340.spon - tdsls340.wpon - tdsls340.wsqn - tdsls340.seqn - tdsls340.invl if <condition on the data read = true> then return(true) endif break case "tdsls406": read data using: - tdsls406.orno - tdsls406.pono - tdsls406.sqnb - tdsls406.dsqn - tdsls406.invl if <condition on the data read = true> then return(true) endif break default: break endcase return (false) } </pre>

Process Extensions for SalesCheckInventory

The following process extension(s) is/are available:

[SalesCheckInventory.CustomSorting](#)

[SalesCheckInventory.DefaultInventoryShortageOption](#)

[SalesCheckInventory.SkipShortageLine](#)

SalesCheckInventory.CustomSorting

Define a customer specific 'Sort Lines By' for Sales Check Inventory.

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to set a customer defined sorting from an extension. This sorting will be used in 'Check Inventory for Sales Orders' (tdsls4217m000) when the 'Sort Lines By' option 'Customer Defined' is selected.

To implement this process extension, you need to implement the following method(s):

tdext.sls0001.get.customer.defined.order.by

Syntax:	long tdext.sls0001.get.customer.defined.order.by(ref string o.order.by.string.mb())
Usage:	Expl: Use this method to get a customer defined order by clause from an extension. This order by clause will be used in 'Check Inventory for Sales Orders' (tdsls4217m000) when the 'Sort Lines By' option 'Customer Defined' is selected. The standard order by clauses used by the 'Sort Lines By' are: - Order Date: " order by tdsls417.odat asc " - Customer Requested Date: " order by tdsls417.ddtc asc " - Planned Delivery Date: " order by tdsls417.ddta asc " Using 'Customer Defined' option, an order by constructed by the extension could be, as an example: " order by tdsls417.cdf_0001 desc, tdsls417.odat asc " Note: 1. Only attributes from table tdsls417 can be used, including customer defined fields. 2. Run time errors occur when specifying attributes from other tables or when wrong syntax is constructed. Pre: NA Post: NA Input: NA Output: o.order.by.string.mb - Maximum string length is 500. Multi byte attributes are allowed. Return: 0 - Success DALHOOKERROR - When an error occurs in determination of the order by clause.

SalesCheckInventory.DefaultInventoryShortageOption

Determine customer specific default 'Automatic Inventory Shortage Option' for Sales Check Inventory.

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to set a customer determined automatic inventory option (enumerated value of domain tdsls.ssop) to a value other than the defaulted value based on LN Master Data (sales order type) from an extension. This option will be used in 'Check Inventory for Sales Orders' (tdsls4217m000) and any other process triggering the automatic inventory handling in Sales.

To implement this process extension, you need to implement the following method(s):

tdext.sls0002.get.customer.determined.default.automatic.inventory.shortage.option

Syntax:	<pre>long tdext.sls0002.get.customer.determined.default.automatic.inventory.shortage.option(domain tdsls.koor i.document.type, domain tcorno i.document, domain tcpono i.line, domain tcpono i.sequence, domain tcpono i.component.sequence, domain tccotp i.order.type, domain tcitem i.item, boolean i.recheck.inventory.promised.line, domain tdsls.ssop i.automatic.inventory.shortage.option.from.master.data, ref domain tdsls.ssop o.automatic.inventory.shortage.option)</pre>
Usage:	Expl: Use this method to get a customer determined default automatic inventory handling option from an extension. This option will be used in 'Check Inventory for Sales Orders' (tdsls4217m000) and any other process triggering the automatic inventory handling in Sales. The standard logic in LN determines the automatic inventory shortage handling option (enumerated value of domain tdsls.ssop) from the sales order type and is also determined by the item and if inventory is rechecked. Using this process extension, the defaulted value can be changed to another option of the existing options of the domain. As an example: the option defaulted by standard LN may be 'ATP - When Available' (tdsls.ssop.atp.fixed.wh). Based on logic in the extension, this value may be overruled for specific order lines by 'ATP - When Available - Single Delivery' (tdsls.ssop.del.date). Note: 1. The input arguments can be used to determine the option. 2. Only values from domain tdsls.ssop can be used. 3. Run time errors occur when non-existing enumerated options of domain tdsls.ssop are provided; an empty value is not overruling the default value from the master data. 4. Standard LN logic may overrule the selected option, otherwise standard logic may fail.

Pre:	NA	
Post:	NA	
Input:	i.document.type	- Sales Quote Line or Sales Order Line or Sales Order Line Component
	i.document	- Sales Quote or Sales Order
	i.line	- Sales Quote Line or Sales Order Line
	i.sequence	- Sales Quote Alternative or Sales Order Sequence
	i.component.sequence	- Sales Order Component Line
	i.order.type	- Sales Order Type
	i.item	- Sales Quote Line Item or Sales Order Line Item or Sales Order Component Item
	i.recheck.inventory.promised.line	- If inventory is being rechecked (True/False)
	i.automatic.inventory.shortage.option.from.master.data	- The option determined by standard LN.
Output:	o.automatic.inventory.shortage.option	- The option determined by the extension.
Return:	0	- Success
	DALHOOKERROR	- When an error occurs in the determination of the inventory option.

SalesCheckInventory.SkipShortageLine

Skips Sales Order Inventory Shortage Lines during Sales Check Inventory.

To implement this process extension, you can use the information below:

Usage:	Process Extension SalesCheckInventory.SkipShortageLine can be used to skip Sales Order Inventory Shortage Lines during Sales Check Inventory. Note: Shortage lines can be present for Sales Order Lines and Sales Order Line Components. Sales Order Line Components are used when 'Advanced Kitting' is implemented and Component Handling of the sales order line is set to 'Component Lines'. Shortage Lines for Sales Quotes are not supported/handled in this flow. Session where this Process Extension can be implemented: - Check Inventory Sales Orders (tdsls4217m000) Fields that are available to be used in this Process Extension: - Only the primary key fields of the Sales Order Inventory Shortage Lines (tdsls417) are current. These fields are: - tdsls417.orno - tdsls417.pono - tdsls417.sqnb - tdsls417.csqn - tdsls417.koor - Applicable values in this process are

'Sales Order Line' and
'Sales Order Line Component'

The Check Inventory Sales Orders process checks and rechecks (if applicable) the inventory for sales order lines and sales order component lines, based on shortage lines in table tdsls417. The primary key fields of this table are current and can e.g., be used to read data from the sales order (line) and also sales order component line in case of component handling to build skip conditions.

External variables that are available to be used in this Process Extension:

- proc_ext_skip_shortage_line_rechecking [type: boolean].
Supported values are:

- True - The process is rechecking inventory for (promised) lines.
- False - The process is checking the inventory for lines.

Note: tables and external variables must also be declared in the Process Extension.

Pseudocode:

Below you can find an example how to handle the conditions for sales order lines and component lines.

Hook: Declarations

```
table tdsls417
table tdsls401
table tdsls463
```

```
extern boolean      proc_ext_skip_shortage_line_rechecking
```

Hook: ext.skip

```
function extern boolean ext.skip()
{
    on case tdsls417.koor
    case tdsls.koor.sls.line:      |* Sales Order Line

        read data using tdsls417.orno, tdsls417.pono,
        tdsls417.sqnb

        if <condition on the data read = true> then
            return(true)
        endif

        break

    case tdsls.koor.sls.line.comp: |* Sales Order Line Component

        read data using tdsls417.orno, tdsls417.pono,
        tdsls417.sqnb, tdsls417.csqn

        if <condition on the data read = true> then
            return(true)
        endif

        So in case of a component line, component lines
        can also be skipped based on tdsls401 data
        if the data is read.
```

	<pre> break default: break endcase return (false) } </pre>

Process Extensions for SalesContract

The following process extension(s) is/are available:

[SalesContract.SkipPrintSalesContract](#)

SalesContract.SkipPrintSalesContract

Skips Sales Contract when Printing Sales Contracts.

This process extension is available from [KB3617247](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension SalesContract.SkipPrintSalesContract can be used to skip Sales Contract when printing the Sales Contracts. Session where this Process Extension can be implemented: - Print Sales Contract Reports (tdsls3401m000) Fields that are available to be used in this Process Extension: - All fields of table Sales Contracts (tdsls300) Note: Table must also be declared in the Process Extension. So skip conditions can be built on current tdsls300 data as instructed below. Pseudocode: Here you can find an example how to handle the conditions for sales contracts. Hook: Declarations <pre> table ttdsls300 </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { if <condition on tdsls300 = true> then </pre>

	<pre> return(true) endif return (false) } </pre>

Process Extensions for SalesContractDeliveryScheme

The following process extension(s) is/are available:

[SalesContractDeliveryScheme.SkipGenerateOrder](#)

SalesContractDeliveryScheme.SkipGenerateOrder

Skips Sales Contract Delivery Scheme Lines when Generating Sales Order.

This process extension is available from [KB2285776](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension SalesContractDeliveryScheme.SkipGenerateOrder can be used to skip Sales Contract Delivery Scheme Lines when Generating Sales Order. Sessions where this Process Extension can be implemented: - Generate Sales Orders from Delivery Schemes (tdsln3204m000) - Sales Contract Delivery Lines (tdsln3104m000) Fields that are available to be used in this Process Extension: - All fields of table Sales Contracts (tdsln300) - All fields of table Sales Contract Lines (tdsln301) - All fields of table Sales Contract Delivery Lines (tdsln304) Note: tables must also be declared in the Process Extension. Pseudocode: Below you can find an example: Hook: Declarations <pre> table ttdsln304 </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { </pre>

	<pre> if <condition on tds1s304 = true> then return(true) endif return (false) } </pre>

Process Extensions for SalesOrder

The following process extension(s) is/are available:

[SalesOrder.DetermineSupplier](#)

[SalesOrder.SkipAdditionalCostSet](#)

[SalesOrder.SkipApprove](#)

[SalesOrder.SkipPrint](#)

[SalesOrder.SkipPrintAcknowledgement](#)

[SalesOrder.SkipReleaseToWarehousing](#)

SalesOrder.DetermineSupplier

Use custom logic to determine the supplier for a Sales Order.

This process extension is available from [KB3536033](#).

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to implement custom logic to determine the supplier for a Sales Order.

To implement this process extension, you need to implement the following method(s):

`tdext.sls0006.sales.order.determine.supplier.for.direct.delivery`

Syntax:	<pre> long tdext.sls0006.sales.order.determine.supplier.for.direct.delivery(domain tccwoc i.sales.office, </pre>

	<pre> domain tccom.bpid i.sold.to.bp, domain tccom.bpid i.ship.to.bp, domain tcitem i.item, domain tcdate i.search.date, domain tccom.bpid i.buy.from.bp.from.standard.logic, domain tccom.bpid i.ship.from.bp.from.standard.logic, ref domain tccom.bpid o.buy.from.bp, ref domain tccom.bpid o.ship.from.bp) </pre>
Usage:	Expl: This method allows for custom logic to be implemented when determining the supplier for a sales order in a direct delivery scenario. The method is invoked at various stages, not just during the generation of the direct delivery purchase order line. For example, it is also called when inserting or modifying a sales order line to ensure that the correct tax values are applied. The method provides flexibility in how business partners are determined: * If only a buy-from business partner is returned, then LN will attempt to find a default ship-from business partner associated with that buy-from business partner. * If only a ship-from business partner is returned, LN will try to locate a default buy-from business partner linked to that ship-from business partner. If no default was found in either of the 2 situations above, LN will revert to its standard logic to determine the supplier. In cases where: * The extension returns a non-zero return-value, or * Both the buy-from and ship-from fields are left empty, or * One of the business partners is not present in the database, LN will automatically fall back to the standard supplier determination logic. Pre: Not Applicable. Post: Not Applicable. Input: - i.sales.office - Sales Office - i.sold.to.bp - Sold-to Business Partner - i.ship.to.bp - Ship-to Business Partner - i.item - Item - i.search.date - Search Date - i.buy.from.bp.from.standard.logic - The Buy-from Business Partner as determined under standard logic. - i.ship.from.bp.from.standard.logic - The Ship-from Business Partner as determined under standard logic. Output: o.buy.from.bp - The Buy-from Business Partner as determined by the extension. * non-empty: - if the given buy-from business partner does not exist, then LN will use standard logic to determine the supplier. * empty: - If only a ship-from business partner is returned, LN will try to locate a default buy-from business partner linked to that ship-from business partner. o.ship.from.bp - The Ship-from Business Partner as determined by the extension.

	<pre> * non-empty: if the given ship-from business partner does not exist, then LN will use standard logic to determine the supplier. * empty: If only a buy-from business partner is returned, then LN will attempt to find a default ship-from business partner associated with that buy-from business partner. Return: 0 <> 0 - Success - When an error occurs in the determination of the supplier. LN will use the standard logic to determine the supplier. </pre>

SalesOrder.SkipAdditionalCostSet

Skips Additional Cost Sets when retrieving Additional Cost Set.

This process extension is available from [KB3597761](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension SalesOrder.SkipAdditionalCostSet can be used to skip (exclude) particular Additional Cost Sets when retrieving (determining) an Additional Cost Set for a Sales Order, Shipment or Price Calculation. Both on header and line level. Processes where this Process Extension can be implemented: - Approve Sales Order - Confirm Shipment - Save Price Calculation - (Re)calculate Additional Costs Fields that are available to be used in this Process Extension are the underneath fields of table Sales Additional Cost Set Scenarios (tdsls027): - Sold-to Business Partner (tdsls027.ofbp) - Ship-to Business Partner (tdsls027.stbp) - Item (tdsls027.item) - Cost Set (tdsls027.ccos) - Interactive Adding of Additional Costs (tdsls027.inta) External variables that are available to be used in this Process Extension: - proc_ext_skip_add_cost_set [type: string(20)]

Supported values are:

- tdsls400
- tdsls401
- whinh430
- whinh431
- tdpcg200
- tdpcg201

This external variable indicates the context from which the function is being called.

Additionally, the primary key fields of the relevant object and its line (if applicable) are populated using the corresponding input arguments.

Note: tables and external variables must also be declared in the Process Extension.

Pseudocode:

Below you can find an example:

Hook: Declarations

```
table tdsls027

extern          string  proc_ext_skip_add_cost_set(20)
```

Hook: ext.skip

```
function extern boolean ext.skip()
{
  on case proc_ext_skip_add_cost_set
  case "tds400":
    if <condition on tds400 = true> then
      return(true)
    endif

    Retrieval (determining) of Additional Cost
    Set is called for Sales Order header.

    break
  case "tds401":
    if <condition on tds401 = true> then
      return(true)
    endif

    Retrieval (determining) of Additional Cost
    Set is called for Sales Order Line.

    break
  case "whinh430":
    if <condition on whinh430 = true> then
      return(true)
    endif

    Retrieval (determining) of Additional Cost
    Set is called for Shipment header.

    break
  case "whinh431":
```

	<pre> if <condition on tdsls027 = true> then return(true) endif Retrieval (determining) of Additional Cost Set is called for Shipment Line. break case "tdpcg200": if <condition on tdsls027 = true> then return(true) endif Retrieval (determining) of Additional Cost Set is called for Price Calculation header. break case "tdpcg201": if <condition on tdsls027 = true> then return(true) endif Retrieval (determining) of Additional Cost Set is called for Price Calculation Line. break default: break endcase return (false) } </pre>

SalesOrder.SkipApprove

Skips Sales Order when Approving.

To implement this process extension, you can use the information below:

Usage:	Process Extension SalesOrder.SkipApprove can be used to skip Sales Orders when Approving the Sales Order. Sessions where this Process Extension can be implemented: - Approve Sales Orders (tdsls421lm000). - All sessions and processes that trigger the Approve. Fields that are available to be used in this Process Extension: - All fields of table Sales Order (tdsls400). Note: tables must also be declared in the Process Extension. Pseudocode: Below you can find an example. Hook: Declarations

	<pre> table tdsls400 Hook: ext.skip function extern boolean ext.skip() { if <condition on tdsls400 = true> then return(true) endif return (false) } </pre>

SalesOrder.SkipPrint

Skips Sales Order when Printing.

This process extension is available from [KB2300881](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension SalesOrder.SkipPrint can be used to skip Sales Orders when Printing the Sales Order. Session where this Process Extension can be implemented: - Print Sales Orders (tds405m000) Fields that are available to be used in this Process Extension: - All fields of table Sales Orders (tds400) Note: Table must also be declared in the Process Extension. So skip conditions can be built on current tds400 data as instructed below. Pseudocode: Here you can find an example how to handle the conditions for sales orders. Hook: Declarations <pre> table tds400 Hook: ext.skip function extern boolean ext.skip() { if <condition on tds400 = true> then return(true) endif return (false) } </pre>

SalesOrder.SkipPrintAcknowledgement

Skips Sales Order when Printing Sales Order Acknowledgement.

To implement this process extension, you can use the information below:

Usage:	Process Extension SalesOrder.SkipPrintAcknowledgement can be used to skip Sales Orders when printing the Sales Order Acknowledgement. Sessions where this Process Extension can be implemented: - Print Sales Order Acknowledgements/RMAs (tdsils4401m000) - All sessions and processes that trigger the printing of the sales order acknowledgement (like automatic processing logic). Fields that are available to be used in this Process Extension: - All fields of table Sales Order (tdsils400) Note: tables must also be declared in the Process Extension. Pseudocode: Below you can find an example: Hook: Declarations <pre> table tdsils400 </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { if <condition on tdsils400 = true> then return(true) endif return (false) } </pre>

SalesOrder.SkipReleaseToWarehousing

Skips Sales Order Lines and Sales Order Line Components when Releasing to Warehousing.

To implement this process extension, you can use the information below:

Usage:	Process Extension SalesOrder.SkipReleaseToWarehousing can be used to skip Sales Order Lines and Sales Order Line Components when Releasing to Warehousing. Note: Sales Order Line Components are used when 'Advanced Kitting' is implemented and Component Handling of the sales order line is set to 'Component Lines'.

Sessions where this Process Extension can be implemented:

- Release Sales Orders to Warehousing (tdspls4246m000)
- All sessions and processes that trigger the release of lines to Warehousing (like automatic processing logic).

Fields that are available to be used in this Process Extension:

- All fields of table Sales Order Lines (tdspls401) when releasing order lines or component lines.
- All fields of table Sales Order Line Components (tdspls463) when releasing component lines.

External variables that are available to be used in this Process Extension:

- `proc_ext_skip_rtw_so_line_type` [type: string(20)].
- Supported values are:

- `order_line`
- `component_line`

Note: tables and external variables must also be declared in the Process Extension.

The sales order release to warehousing process releases sales order lines. In case of component handling (tdspls401.cphl = tdcphl.component.lines), sales order line is selected first and then the component lines are released.

The sales order line table is tdspls401.

The sales order line components table is tdspls463.

So in case of component lines, skip conditions can be built both on current tdspls401 data and tdspls463 data as instructed below.

The process extension must be applied to these two tables.

Pseudocode:

Below you can find an example how to handle the conditions for sales order lines and component lines.

Hook: Declarations

```
table tdspls401
table tdspls463

extern          string  proc_ext_skip_rtw_so_line_type(20)
```

Hook: ext.skip

```
function extern boolean ext.skip()
{
    on case proc_ext_skip_rtw_so_line_type
    case "order_line":
        if <condition on tdspls401 = true> then
            return(true)
        endif

        So in case of order_line, component handling
        sales order lines can also be skipped based on
        tdspls401 data.

        break
    case "component_line":
        if <condition on tdspls463 = true> then
```

	<pre> return(true) endif AND / OR if <condition on tdsls401 = true> then return(true) endif So in case of component_line, component lines can also be skipped based on tdsls401 data. break default: break endcase return (false) }</pre>

Process Extensions for SalesOrderGenerate

The following process extension(s) is/are available:

[SalesOrderGenerate.DetermineOrderForCombining](#)

[SalesOrderGenerate.SuppressCombiningOrderLines](#)

SalesOrderGenerate.DetermineOrderForCombining

Determine customer specific sales order for combining sales order lines when generating orders.

This process extension is available from [KB2317750](#).

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to let the extension decide which sales order must be used to combine a sales order line on an existing sales order. It is implemented for generate processes that can create a new sales order header.

To implement this process extension, you need to implement the following method(s):

tdext.sls0004.determine.sales.order.for.combining.order.lines.during.generation

Syntax:	<pre> long tdext.sls0004.determine.sales.order.for.combining.order.lines.during.generation(domain tcorno i.sales.order, domain tcpono i.sales.order.line, domain tdsls.corg i.sales.order.origin, long i.source.generating.process, ref domain tcorno o.sales.order) </pre>																		
Usage:	Expl: This method can be implemented to get a customer determined sales order that will be used when generating sales order lines. If this extension returns a sales order, then that sales order will be used to add the new sales order line to. Note that after calling this extension, LN will also consider the extension SalesOrderGenerate.SuppressCombiningOrderLines. If implemented, that extension could prevent combining the order line. If the extension returns a sales order that is cancelled, closed or not updatable, then an error is returned during generation. This extension is implemented for multiple sales order generation processes. See below. Input: <code>i.sales.order</code> - Sales Order (Optional) If filled, this can refer to either the sales order that would be used by the standard logic, or it refers to a sales order series. In the latter case, LN would generate a new sales order. If empty, LN standard logic would generate a new sales order. <code>i.sales.order.line</code> - Sales Order Line (Optional) If filled, this refers to the sales order line that would be used by standard logic; LN would create a new detail-line. <code>i.sales.order.origin</code> - Sales Order Origin (tdsls400.corg) <code>i.source.generating.process</code> - Indicates the source process from / for which the sales order is generated and where an order header can be created. The following generation sources can support creation of multiple order headers and can be affected by the process extension: <table> <thead> <tr> <th>Source Process</th><th>Value of i.source.generating.process</th></tr> </thead> <tbody> <tr><td>- Quote</td><td>1</td></tr> <tr><td>- Retro-Billing</td><td>10</td></tr> <tr><td>- Contract Delivery Scheme</td><td>12</td></tr> <tr><td>- Template</td><td>14</td></tr> <tr><td>- Subcontracting Purchase Order</td><td>15</td></tr> <tr><td>- Sales Transfer</td><td>17</td></tr> <tr><td>- Price Calculation</td><td>30</td></tr> <tr><td>- Received Customer Order</td><td>31</td></tr> </tbody> </table>	Source Process	Value of i.source.generating.process	- Quote	1	- Retro-Billing	10	- Contract Delivery Scheme	12	- Template	14	- Subcontracting Purchase Order	15	- Sales Transfer	17	- Price Calculation	30	- Received Customer Order	31
Source Process	Value of i.source.generating.process																		
- Quote	1																		
- Retro-Billing	10																		
- Contract Delivery Scheme	12																		
- Template	14																		
- Subcontracting Purchase Order	15																		
- Sales Transfer	17																		
- Price Calculation	30																		
- Received Customer Order	31																		

	Other generation sources can support creation of order headers but cannot be affected by the process extension and will not invoke the process extension because: - LN standard creates one order and line (hard coded) - Multiple orders may cause conflicts with current standard behavior or maybe unwanted. These are: - Return Rejected - Direct Delivery from Procurement - Consumption (Cons. Inv.) - Copy Template - Schedule Return - Price Calculator - Shipment Additional Costs Output: o.sales.order - The sales Order as determined by the extension. * non-empty: The sales order line will be added to the given sales order. if the given sales order does not exist, then LN will create a new sales order. * empty: Standard logic from LN will be used to decide which existing sales order (if any) will be used to add the sales order line to. Return: 0 - Success - When an error occurs in the determination of the sales order. The standard logic for combining will be applied. <> 0

SalesOrderGenerate.SuppressCombiningOrderLines

Suppress combining of sales order lines when generating orders.

This process extension is available from [KB2285776](#).

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to suppress combining sales order lines on the same sales order. It is implemented for generate processes that can create a new sales order header.

To implement this process extension, you need to implement the following method(s):

tdext.sls0003.suppress.combining.sales.order.lines.during.generation

Syntax:	<pre>boolean tdext.sls0003.suppress.combining.sales.order.lines.during.generation(domain tcorno i.sales.order, domain tcpono i.sales.order.line, domain tdsls.corg i.sales.order.origin, long i.source.generating.process)</pre>																		
Usage:	Expl.: This method can be implemented to suppress combining sales order lines on the same sales order during generation. The standard logic in LN will first determine whether combining is allowed at all. If so, this process extension will then be called to decide whether combining must be suppressed. When this method is called, the following applies: - i.sales.order is always filled; - i.sales.order.line can be zero or > zero - A record of tdsls400 is current. Input: - i.sales.order - Sales Order - i.sales.order.line - Sales Order Line Note that the sequence cannot be input for the order generation process. - i.sales.order.origin - Sales Order Origin (tdsls400.corg) - i.source.generating.process - Indicates the source process from / for which the sales order is generated and where an order header can be created. The following generation sources can support creation of multiple order headers and can be affected by the process extension: <table> <thead> <tr> <th>Source Process</th><th>Value of i.source.generating.process</th></tr> </thead> <tbody> <tr><td>- Quote</td><td>1</td></tr> <tr><td>- Retro-Billing</td><td>10</td></tr> <tr><td>- Contract Delivery Scheme</td><td>12</td></tr> <tr><td>- Template</td><td>14</td></tr> <tr><td>- Subcontracting Purchase Order</td><td>15</td></tr> <tr><td>- Sales Transfer</td><td>17</td></tr> <tr><td>- Price Calculation</td><td>30</td></tr> <tr><td>- Received Customer Order</td><td>31</td></tr> </tbody> </table> Other generation sources can support creation of order headers but cannot be affected by the process extension and will not invoke the process extension because: - LN standard creates one order and line (hard coded) - Multiple orders may cause conflicts with current standard behavior or maybe unwanted. These are: - Return Rejected - Direct Delivery from Procurement - Consumption (Cons. Inv.) - Copy Template	Source Process	Value of i.source.generating.process	- Quote	1	- Retro-Billing	10	- Contract Delivery Scheme	12	- Template	14	- Subcontracting Purchase Order	15	- Sales Transfer	17	- Price Calculation	30	- Received Customer Order	31
Source Process	Value of i.source.generating.process																		
- Quote	1																		
- Retro-Billing	10																		
- Contract Delivery Scheme	12																		
- Template	14																		
- Subcontracting Purchase Order	15																		
- Sales Transfer	17																		
- Price Calculation	30																		
- Received Customer Order	31																		

	- Schedule Return - Price Calculator - Shipment Additional Costs Output: Not Applicable. Return: True False - The extension has determined that combining must be suppressed for the given input. - The standard logic for combining applies.

Process Extensions for SalesOrderInvoiceLine

The following process extension(s) is/are available:

[SalesOrderInvoiceLine.SkipReleaseToInvoicing](#)

SalesOrderInvoiceLine.SkipReleaseToInvoicing

Skips Sales Order Invoice Line when Releasing to Invoicing.

This process extension is available from [KB3570312](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension SalesOrderInvoiceLine.SkipReleaseToInvoicing can be used to skip specific Sales Order Invoice Lines when Releasing to Invoicing. Sessions where this Process Extension can be implemented: - Release Sales Orders/Schedules to Invoicing (tdsls4247m000) for option 'Release Order' - Print Sales Draft Invoices (tdsls4447m000) for option 'Print Order' Fields that are available to be used in this Process Extension: - All fields of table Sales Order Actual Delivery Lines (tdsls406) - All fields of table Linked Ord. Line Data for Invoicing (tdsls408) Note: Tables must also be declared in the Process Extension. So skip conditions can be built on current tdsls406 and tdsls408 data as instructed below. Table tdsls408 is used for invoicing sales order lines within a set. If a sales order line that is part of a set is skipped for invoicing, all other sales order lines within that set are also skipped. Pseudocode: Here you can find an example how to handle the conditions for sales order lines. Hook: Declarations

	<pre> table tdsls406 table tdsls408 Hook: ext.skip function extern boolean ext.skip() { if <condition on tdsls406 = true> then return(true) endif return (false) } </pre>

Process Extensions for SalesOrderLine

The following process extension(s) is/are available:

[SalesOrderLine.AllowDisplaySelectedContract](#)

[SalesOrderLine.AllowSaveInCaseOfInsufficientATP](#)

[SalesOrderLine.SkipGenerateProjectStructure](#)

[SalesOrderLine.SkipPrint](#)

[SalesOrderLine.SkipPrintAcknowledgement](#)

SalesOrderLine.AllowDisplaySelectedContract

Determine whether the only one selected contract for the sales order line is allowed to be displayed.

This process extension is available from [KB3638151](#).

Technical information for this process extension:

Usage:	This process extension determines if the only one selected contract for the sales order line is allowed to start session "Selected Sales Contract Lines" (tds3512s000), while the standard would not start it when only one valid contract is found. Subsequently, the contract can be manually linked or not.

To implement this process extension, you need to implement the following method(s):

tdext.sls0008.sales.order.line.allow.display.selected.contract

Syntax:	<pre>long tdext.sls0008.sales.order.line.allow.display.selected.contract(domain tccono i.contract, domain tcpono i.contract.line, domain tccwoc i.contract.office, domain tcitem i.item, domain tccom.bpid i.sold.to.bp, domain tccom.bpid i.ship.to.bp, ref boolean i.display.selected.contract.from.standard.logic, boolean o.display.selected.contract.from.extension)</pre>																		
Usage:	Expl: Use this method to determine if the only one selected contract for the sales order line is allowed to start session "Selected Sales Contract Lines" (tdsls3512s000), while the standard would not start it when only one valid contract is found. Subsequently, the contract can be manually linked or not. Using this process extension, the 'display selected contract' variable from standard logic can be changed. The 'display selected contract' from extension is only retrieved and used if: - Process Extension SalesOrderLine.AllowDisplaySelectedContract (tdsls.sol.allow.disp.sel.contr) is implemented; - No errors are found during executing the extension. Pre: Not applicable Post: Not applicable Input: <table> <tr><td>i.contract</td><td>- Sales Contract</td></tr> <tr><td>i.contract.line</td><td>- Sales Contract Line</td></tr> <tr><td>i.contract.office</td><td>- Sales Contract Office</td></tr> <tr><td>i.item</td><td>- Item</td></tr> <tr><td>i.sold.to.bp</td><td>- Sold-to Business Partner</td></tr> <tr><td>i.ship.to.bp</td><td>- Ship-to Business Partner</td></tr> <tr><td>i.display.selected.contract.from.standard.logic</td><td>- As defaulted under the standard logic</td></tr> </table> Output: o.display.selected.contract.from.extension - The 'display selected contract' (true/false) determined by the extension Return: <table> <tr><td>0</td><td>- Success</td></tr> <tr><td>DALHOOKERROR</td><td>- When an error occurs in the determination of the 'display selected contract' from extension</td></tr> </table>	i.contract	- Sales Contract	i.contract.line	- Sales Contract Line	i.contract.office	- Sales Contract Office	i.item	- Item	i.sold.to.bp	- Sold-to Business Partner	i.ship.to.bp	- Ship-to Business Partner	i.display.selected.contract.from.standard.logic	- As defaulted under the standard logic	0	- Success	DALHOOKERROR	- When an error occurs in the determination of the 'display selected contract' from extension
i.contract	- Sales Contract																		
i.contract.line	- Sales Contract Line																		
i.contract.office	- Sales Contract Office																		
i.item	- Item																		
i.sold.to.bp	- Sold-to Business Partner																		
i.ship.to.bp	- Ship-to Business Partner																		
i.display.selected.contract.from.standard.logic	- As defaulted under the standard logic																		
0	- Success																		
DALHOOKERROR	- When an error occurs in the determination of the 'display selected contract' from extension																		

SalesOrderLine.AllowSaveInCaseOfInsufficientATP

Use custom logic to allow saving of a sales order line in case of insufficient ATP.

This process extension is available from [KB3539185](#).

Technical information for this process extension:

Usage:	With this Process Extension, it is possible to implement custom logic to allow saving of a sales order line in case of insufficient ATP.

To implement this process extension, you need to implement the following method(s):

tdext.sls0007.sales.order.line.allow.save.in.case.of.insufficient.atp

Syntax:	<pre>boolean tdext.sls0007.sales.order.line.allow.save.in.case.of.insufficient.atp(domain tcorno i.sales.order, domain tcpono i.sales.order.line, domain tcpono i.sales.order.line.sequence)</pre>
Usage:	Expl.: This function can be implemented to allow saving a sales order line in case of insufficient ATP. The standard logic of LN only allows this in case the sales order (line) is created or updated through EDI. With this function it is possible to also allow this for non EDI orders. When this function is called from the standard, the following applies: - i.sales.order is filled - i.sales.order.line is filled - i.sales.order.line sequence is filled (can be zero) - A record of tdsls401 is current Input: - i.sales.order - Sales Order - i.sales.order.line - Sales Order Line - i.sales.order.line.sequence - Sales Order Line Sequence Output: Not Applicable Return: - true - The extension has determined that it is allowed to save the order line when there is insufficient ATP. - false - The extension has determined that it is not allowed to save the order line when there is insufficient ATP. This reflects the standard LN logic for non EDI orders.

SalesOrderLine.SkipGenerateProjectStructure

Skips Sales Order Line when Generating Project Structure.

This process extension is available from [KB3546324](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension SalesOrderLine.SkipGenerateProjectStructure can be used to skip specific Sales Order Lines when Generating a Project Structure. Session where this Process Extension can be implemented: - Generate (Project PCS) Structure for Sales Orders (tdsls4244m000)

	Fields that are available to be used in this Process Extension: - All fields of table Sales Order (tdsls400) - All fields of table Sales Order Lines (tdsls401) Note: Tables must also be declared in the Process Extension. So skip conditions can be built on current tdsls400 and tdsls401 data as instructed below. Pseudocode: Here you can find an example how to handle the conditions for sales order lines. Hook: Declarations <pre> table ttdsls400 table ttdsls401 </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { if <condition on tdsls401 = true> then return(true) endif return (false) } </pre>

SalesOrderLine.SkipPrint

Skips Sales Order Line when Printing.

This process extension is available from [KB2300881](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension SalesOrderLine.SkipPrint can be used to skip specific Sales Order Lines when Printing the Sales Order. Session where this Process Extension can be implemented: - Print Sales Order Lines (tdsls4409m000) Fields that are available to be used in this Process Extension: - All fields of table Sales Order (tdsls400) - All fields of table Sales Order Lines (tdsls401) Note: Tables must also be declared in the Process Extension. So skip conditions can be built on current tdsls400 and tdsls401 data as instructed below. Pseudocode: Here you can find an example how to handle the conditions for sales order lines. Hook: Declara1Gations

	<pre> table tdsls400 table tdsls401 Hook: ext.skip function extern boolean ext.skip() { if <condition on tdsls401 = true> then return(true) endif return (false) } </pre>

SalesOrderLine.SkipPrintAcknowledgement

Skips Sales Order Line when Printing Sales Order Acknowledgement.

To implement this process extension, you can use the information below:

Usage:	Process Extension SalesOrderLine.SkipPrintAcknowledgement can be used to skip Sales Order Lines when printing the Sales Order Acknowledgement. Sessions where this Process Extension can be implemented: - Print Sales Order Acknowledgements/RMAs (tdsls4401m000) - All sessions and processes that trigger the printing of the sales order acknowledgement (like automatic processing logic). Fields that are available to be used in this Process Extension: - All fields of table Sales Order (tdsls400) - All fields of table Sales Order Line (tdsls401) Note: tables must also be declared in the Process Extension. Pseudocode: Below you can find an example: Hook: Declarations <pre> table tdsls400 table tdsls401 Hook: ext.skip function extern boolean ext.skip() { if <condition on tdsls401 = true> then return(true) endif return (false) } </pre>

Process Extensions for SalesQuote

The following process extension(s) is/are available:

[SalesQuote.SkipPrintSalesQuote](#)

SalesQuote.SkipPrintSalesQuote

Skips Sales Quote when Printing Sales Quote.

To implement this process extension, you can use the information below:

Usage:	Process Extension SalesQuote.SkipPrintSalesQuote can be used to skip Sales Quotes when printing the Sales Quote. Sessions where this Process Extension can be implemented: - Print Sales Quotation (tds1s1401m000) Fields that are available to be used in this Process Extension: - All fields of table Sales Quotation (tds1s100) Note: tables must also be declared in the Process Extension. Pseudocode: Below you can find an example: Hook: Declarations <pre>table tds1s100</pre> Hook: ext.skip <pre>function extern boolean ext.skip() { if <condition on tds1s100 = true> then return(true) endif return (false) }</pre>

Process Extensions for SalesQuoteLine

The following process extension(s) is/are available:

[SalesQuoteLine.SkipPrintSalesQuote](#)

SalesQuoteLine.SkipPrintSalesQuote

Skips Sales Quote Lines when Printing Sales Quote.

This process extension is available from [KB2330719](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension SalesQuoteLine.SkipPrintSalesQuote can be used to skip Sales Quote Lines when printing the Sales Quote. Sessions where this Process Extension can be implemented: - Print Sales Quotation (tds1s1401m000) Fields that are available to be used in this Process Extension: - All fields of table Sales Quotation (tds1s100) - All fields of table Sales Quotation Lines (tds1s101) Note: tables must also be declared in the Process Extension. Pseudocode: Below you can find an example: Hook: Declarations <pre> table ttds1s101 </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { if <condition on tds1s101 = true> then return(true) endif return (false) } </pre>

Process Extensions for SelfBilledInvoiceLine

The following process extension(s) is/are available:

[SelfBilledInvoiceLine.SkipMatching](#)

SelfBilledInvoiceLine.SkipMatching

Skip Self-Billed Invoice Lines when Matching.

To implement this process extension, you can use the information below:

Usage:	Process Extension SelfBilledInvoiceLine.SkipMatching can be used to skip Matching of Self-Billed Invoice Line. Sessions where this Process Extension can be implemented: - Match Self-Billed Invoices (cisli5200m000) Fields that are available to be used in this Process Extension: - Self-Billed Invoice Header (cisli500) - Self-Billed Invoice Lines (cisli505) Note:- This method is not applicable for Manual Matching Process Pseudocode: In the code below, the Self-Billed Invoice line from cisli505 which have Self-Billed Invoice Type(cisli505.type) as "Price Variation" will be skipped during the matching process. Hook: Declarations <pre> table tcisli500 table tcisli505 </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { if cisli505.type = tcisli.intp.price.variation then /* If Self-Billed Invoice Type is Price /* Variation then skip matching of the /* Self-Billed Invoice Line. return(true) endif return (false) } </pre>

Process Extensions for ServiceOrderActivity

The following process extension(s) is/are available:

[ServiceOrderActivity.SkipComplete](#)

ServiceOrderActivity.SkipComplete

Skips Service Order Activity when setting Status to Completed.

To implement this process extension, you can use the information below:

Usage:	Process Extension ServiceOrderActivity.SkipComplete can be used to skip service order activities when setting to status Completed. Sessions where this Process Extension can be implemented: - Complete Service Order Activities (tssoc2206m000) - All sessions and processes that trigger a Service Order Activity to be set to status Completed. Fields that are available to be used in this Process Extension: - Key fields of tssoc210 - tssoc210.orno (Service Order) - tssoc210.acln (Activity Line) These fields can be used to read e.g. tables tssoc200 and tssoc210. Note: tables must also be declared in the Process Extension. Pseudocode: Below you can find an example. Hook: Declarations <pre> table ttssoc200 * Service Orders table ttssoc210 * Service Order Activities </pre> Hook: ext.skip <pre> function extern boolean ext.skip() { if <condition on tssoc200 = true> then return(true) endif if <condition on tssoc210 = true> then return(true) endif return (false) } </pre>

Process Extensions for ShiftsByWorkcell

The following process extension(s) is/are available:

[ShiftsByWorkcell.SkipCompleteShift](#)

ShiftsByWorkcell.SkipCompleteShift

Skips shifts based on Workcell when completing Shifts.

To implement this process extension, you can use the information below:

Usage:	Process Extension ShiftsByWorkcell.SkipCompleteShift can be used to skip certain Shifts when reporting a range of Shifts as complete. Sessions where this Process Extension can be implemented: - Complete Shifts (tirpt4236m000) Fields that are available to be used in this Process Extension: - All fields of table: Shifts by Work Cell (tirpt436)

Process Extensions for ShipmentBuilding

The following process extension(s) is/are available:

[ShipmentBuilding.CustomBuildingConstraints](#)

ShipmentBuilding.CustomBuildingConstraints

Allows to perform additional shipment building constraints during shipment building.

Technical information for this process extension:

Usage:	With this Process Extension it is possible to perform additional checks on the shipment building process. During the shipment building, the process will search for an existing shipment to ship the goods with. There are however shipment building constraints which must force Infor LN to not combine the picked inventory with a found shipment. This process extension allows a customer to define custom shipment building criteria to match the business case which is required.

To implement this process extension, you need to implement the following method(s):

whext.dll0003.custom.shipment.building.constraints

Syntax:	<pre>long whext.dll0003.custom.shipment.building.constraints(domain whinh.shpm i.found.shipment,</pre>

	domain whinh.oorg i.order.origin, domain tcorno i.order.number, domain tcpono i.order.line, domain tcpono i.order.sequence, ref boolean o.use.found.shipment)
Usage:	Expl: This process extension allows for customer specific shipment building constraints. This process extension is called from the Infor LN standard, when one of the following events are triggered: - Create Projected Shipment - Confirm Pick (when Outbound Inspections are not in use) - Approve (when Outbound Inspections are in use) - Cross Docking (when Confirmation of Receipt or Putaway of Inbound Advice is done and the goods are to be Cross-Docked or when Approve is done for the Inbound Inspection where Cross-Docking should be done in non location controlled warehouses.) - Create Outbound Order Line for Cost/Service items which are released to warehousing. Note: This process extension is not called when composing a shipment structure. The constraints that are applicable for the shipment building should also be added as extension points for composing shipments, via dedicated table / field extensions on the related field, in line with the shipment building criteria. The shipment building process will try to search for shipments which are already present to perform the shipment of the newly staged inventory to be shipped. Conditions apply on the combinations which are allowed based on order type specific shipment building criteria. This process extension allows to define custom building criteria to be validated prior to selecting the shipment for a newly created shipment line. Table "Shipments" (whinh430) is current, for the i.found.shipment. Other data that is required to determine if the shipment can be used can be retrieved by making use of the outbound order line key, which is also part of the input of this process extension. Setting the o.use.found.shipment to false will take care that the i.found.shipment will not be used because a custom criteria is not matching with the shipment and the new inventory moved to the staging location. Within Freight Management there is also a process extension called "FreightShipmentBuilding.CustomBuildingConstraints" which will be triggered during the generation of a plan. If freight is implemented and used for planning purposes, it is advisable to apply the same customer specific building criteria in freight management, as otherwise the planning will be overruled by warehousing. Note: If you execute queries on the standard Infor LN tables, bind the table fields to local variables to prevent disturbing the standard flow. Pre: N.a. Post: N.a. Input: i.found.shipment - Shipment that is found during shipment building Outbound Order Line Key i.order.origin i.order.number

	<pre> i.order.line i.order.sequence Output: o.use.found.shipment - Indicator if the found shipment should be used (true/false) Please also set an error message regarding this shipment building constraint, so the users will get feedback when required. Return: 0/DALHOOKERROR </pre>

Process Extensions for StandardCost

The following process extension(s) is/are available:

[StandardCost.SkipActualize](#)

[StandardCost.SkipCalculate](#)

StandardCost.SkipActualize

Standard Cost Skip Actualization and Inventory Revaluation.

Technical information for this process extension:

Usage:	With this process extension, the Standard Cost Actualization and Revaluation can be controlled for specific Item/Enterprise Unit combinations.

To implement this process extension, you need to implement the following method(s):

tiext.cpr0001.standard.cost.skip.actualize

Syntax:	<pre> long tiext.cpr0001.standard.cost.skip.actualize(ref boolean o.skip.actualize, ref string o.message()) </pre>
Usage:	Expl: This function is called in the process of Standard Cost Actualization and Inventory Revaluation, in order to decide if the process must be skipped for a specific Item/Enterprise Unit combination. A message may be returned, to present information about the decision to the user. When this function is called all fields of table:

	- Item Costing Data (ticpr007) are read and current.	
Pre:	NA	
Post:	NA	
Input:	NA	
Output:	o.skip.actualize	- decision result
	o.message	- message, multibyte - max 300 characters
Return:	0	- success
	DALHOOKERROR	- error

StandardCost.SkipCalculate

Standard Cost Skip Calculation.

Technical information for this process extension:

Usage:	With this process extension, the Standard Cost Calculation can be controlled for specific Item, and or Item/Enterprise Unit combinations.

To implement this process extension, you need to implement the following method(s):

tiext.cpr0002.standard.cost.skip.calculate

Syntax:	<pre> long tiext.cpr0002.standard.cost.skip.calculate(long i.level, domain tcitem i.top.item, domain tcemm.grid i.top.eu, domain tcitem i.parent.item, domain tcemm.grid i.parent.eu, ref boolean o.skip.calculate, ref string o.message()) </pre>
Usage:	Expl: This function is called in the process of Standard Cost Calculation, in order to decide if the calculation must be skipped for a specific Item/Enterprise Unit combination. A message may be returned, to present information about the decision to the user. When this function is called all fields of table: - Item Costing (ticpr007) are read and current. Note that fields ticpr007.item and ticpr007.eunt refer to the current Item/Enterprise Unit being considered for calculation. External variables available for use by this process extension function: - proc_ext_std_cost_skip_calc_calculation_code [type: domain tccpcc] Calculation code for which the calculation is executed. - proc_ext_std_cost_skip_calc_calculation_method

	<pre> [type: domain tcccm] single.level: Limited product structure explosion, standard cost at second level is read, not calculated. top.down: Explodes product structure top-down, calculates all lower levels and cost is rolled up to top level. bottom.up: Product structure determines which higher level items need to part of the calculated set. - proc_ext_std_cost_skip_calc_calculation_mode [type: long] 0: calculating only 1: printing multilevel 2: printing single level - proc_ext_std_cost_skip_calc_project [type: domain tccprj] The project, set in case the calculation is started in the context of a project, otherwise empty. - proc_ext_std_cost_skip_calc_calculation_date [type: domain tcdte] The date used for calculation. Used for selection of date-effective base data, like BOM lines/models, routing operations/models and/or operation rates. - proc_ext_std_cost_skip_calc_actualize [type: domain tcyesno] Indicates if actualization of calculated standard cost is requested. - proc_ext_std_cost_skip_calc_effective_date [type: domain tcdte] The date used for calculation. Used for selection of date-effective base data, like BOM lines/models, routing operations/models and/or operation rates. - proc_ext_std_cost_skip_calc_analyze_before_print [type: domain tcyesno] Special calculation mode, calculating before printing while in simulation. - proc_ext_std_cost_skip_calc_uf_calculation [type: boolean] Indicates if calculation is done for an effectivity unit. - proc_ext_std_cost_skip_calc_from_pcs [type: boolean] Indicates if calculation is called in Project (PCS) context. - proc_ext_std_cost_skip_calc_from_sales [type: boolean] Indicates if calculation is called from Sales - or Quotation line context. Note: tables and external variables must also be declared in the Process Extension ***** Example Product Structure. The example assumes that the actual set of Implemented Software Components (tccom0100s000) has the concepts Standard Cost by Enterprise Unit and Job Shop by Site set to Active. Product A (top-item) requires a standard cost for two enterprise units. Its enterprise units have the following properties: A/EU1 (costing source Intercompany Transfer, supplied by EU2) </pre>

	A/EU2 (costing source Job Shop) Because of the trade relation, calculation of A/EU1 uses the standard cost of A/EU2. Standard cost of A/EU2 (costing source Job Shop) is based on the Job Shop Bill of Material defined for the site linked to EU2. The materials required for production of product A are B and C. Items B and C only have standard cost in enterprise unit EU2. A representation of the product structure levels with the dependencies for standard cost calculation: <pre> A (EU1) (level 1) (Costing Source Intercompany Transfer) ^-----A(EU2) (level 2) (Costing Source Job Shop) ^----B(EU2) (level 3) ^----C(EU2) (level 3) </pre> When this function is called in a top-down calculation for top item A, then at some point, the calculation encounters item C and EU2. Besides the available ticpr007 fields: <pre> ticpr007.item is set to C ticpr007.eunt is set to EU2 </pre> The input parameters provide further information about the context: <pre> i.level is set to 3, i.top.item is set to A, i.top.eu is set to EU1, i.parent.item is set to A, i.parent.eu is set to EU2. </pre> Implementation Example: Intention: For a specific Enterprise Unit (EU2), when actualization is requested, calculation (and subsequent actualization) of standard cost is not allowed. Hook Declarations: <pre> table tticpr007 extern domain tcyesno proc_ext_std_cost_skip_calc_actualize Hook tiext.cpr0002.standard.cost.skip.calculate: function extern long tiext.cpr0002... ... { if proc_ext_std_cost_skip_calc_actualize = tcyesno.yes and str.compare(ticpr007.eunt, "EU2 ") = 0 then o.message = "calculation for EU2 is skipped" o.skip.calculate = true endif return(0) } </pre> Pre: NA Post: NA Input: - i.level - the depth in the calculation process - i.top.item - top item - i.top.eu - top enterprise unit - i.parent.item - parent item - i.parent.eu - parent enterprise unit Output: - o.skip.calculate - decision result - o.message - message, multibyte - max 300 characters Return: 0 - success

	DALHOOKERROR	- error

tiext.cpr0002.standard.cost.skip.calculate.item

Syntax:	<pre>long tiext.cpr0002.standard.cost.skip.calculate.item(domain tcitem i.item, ref o.skip.calculate, ref o.message())</pre>
Usage:	Expl: This function is called in the process of Standard Cost Calculation, during bottom-up calculation preparation, in order to decide if the calculation for the specific Item must be skipped entirely (i.e. for all its enterprise units). A message may be returned, to present information about the decision to the user. For this decision function no database record is made current. During the bottom-up calculation preparation process, only the item level dependencies defined by product structure information (Bills of Materials, Production Models, Subcontracting Models etc.) are used to define the calculation order. The first step in this preparation process is to create the set of items to calculate. This is where the skip decision is applied. External variables available for use by this process extension function: - proc_ext_std_cost_skip_calc_calculation_code [type: domain tccpcc] Calculation code for which the calculation is executed. - proc_ext_std_cost_skip_calc_calculation_date [type: domain tcdte] The date used for calculation. Used for selection of date-effective base data, like BOM lines/models, routing operations/models and/or operation rates. - proc_ext_std_cost_skip_calc_actualize [type: domain tcyesno] Indicates if actualization of calculated standard cost is requested. - proc_ext_std_cost_skip_calc_effective_date [type: domain tcdte] The date used for calculation. Used for selection of date-effective base data, like BOM lines/models, routing operations/models and/or operation rates. Note: tables and external variables must also be declared in the Process Extension Implementation Example: Intention: For a specific simulation calculation code (SIM), only a subset of items need to be included in the calculations. Hook Declarations: <pre>extern domain tccpcc proc_ext_std_cost_skip_calc_calculation_code Hook tiext.cpr0002.standard.cost.skip.calculate.item: function extern long tiext.cpr0002...</pre> ...

	<pre> { if proc_ext_std_cost_skip_calc_calculation_code = "SIM" and not item.in.simulation.set(i.item) then o.message = trim\$(i.item) & " not in simulation set" o.skip.calculate = true endif return(0) } </pre> Pre: NA Post: NA Input: i.item - item Output: o.skip.actualize - decision result o.message - message, multibyte - max 300 characters Return: 0 - success DALHOOKERROR - error

Process Extensions for Statement

The following process extension(s) is/are available:

[Statement.SkipPrintStatements](#)

Statement.SkipPrintStatements

Skip Print Statements of account.

This process extension is available from [KB2319017](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension Statement.SkipPrintStatements can be used to skip Printing Statements of Account. Sessions where this Process Extension can be implemented: - Print Statements (tfacr3440m000) Fields that are available to be used in this Process Extension: - All fields of "Open Items" (tfacr500) Note: Tables must also be declared in the Process Extension. Pseudocode: Below you can find an example: Hook: Declarations <pre> table tfacr500 * Open Items A/R </pre>

	<pre> Hook: ext.skip function extern boolean ext.skip() { if <condition on tfacr500 = true> then return(true) endif return (false) } </pre>

Process Extensions for StorageMission

The following process extension(s) is/are available:

[StorageMission.HandleAfterGenerate](#)

StorageMission.HandleAfterGenerate

Allows to add functionality after the generation of a storage mission.

Technical information for this process extension:

Usage:	With this Process Extension it is possible execute additional actions after the creation of a storage list. This process extension will only be triggered when the storage list is generated, not when a reprint or preview is done for the storage list.

To implement this process extension, you need to implement the following method(s):

whext.dll0006.storage.mission.handle.after.generate

Syntax:	<pre> long whext.dll0006.storage.mission.handle.after.generate(domain whinh.btno i.runnumber, domain tccwar i.warehouse, domain whinh.picm i.storage.mission, domain tcncmp i.company, ref boolean o.remove.storage.mission) </pre>
Usage:	Expl: This process extension allows to include customer specific functionality after the process of generating storage missions. This process extension will be called from Infor LN standard when the storage mission is generated via session

	"Generate storage List" (whinh3415m000), or when the generation of storage lists is an automatic order step. For each storage mission the process extension is executed after generation. No table fields are current, based on the input arguments the records must be fetched. When the process extension is executed, the storage missions are committed to the database. Any transactions which are required to the database (inserts, updates, deletes) must be accompanied with a new transaction (db.retry.point / abort/commit). Storage missions are removed when o.remove.storage.mission is set to True or when this function returns a value <> 0. Pre: N.a. Post: N.a. Input: i.runnumber - The run number for which the storage mission is created i.warehouse - Warehouse i.storage.mission - Storage Mission which is generated i.company - Logistic Company of the inbound advice Output: o.remove.storage.mission - Indicator if the standard should remove the storage mission. Return: 0: Success / <> 0: Error

Process Extensions for TargetPrices

The following process extension(s) is/are available:

[TargetPrices.SkipCalculateOrderLines](#)

TargetPrices.SkipCalculateOrderLines

Skips Purchase Order Lines when Calculating Target Prices.

This process extension is available from [KB3540096](#).

To implement this process extension, you can use the information below:

Usage:	Process Extension TargetPrices.SkipCalculateOrderLines can be used to skip Purchase Order Lines when Calculating Target Prices. Sessions where this Process Extension can be implemented: - Calculate Target Prices (tdpcg0233m100) External variables that are available to be used in this Process Extension: - proc_ext_skip_calc_po_lines_is_history_line [type: boolean] Supported values are: - true Current record is an order line from table Purchase Order Line History (tdpur451)

	<pre> - false Current record is a normal order line from table Purchase Order Lines (tdpur401) - proc_ext_skip_calc_po_lines_search_criterion [type: domain tdpcg.tpsp] Supported values are: - tdpcg.tpsp.order.lowest Purchase Order Price (Lowest) - tdpcg.tpsp.order.most.rec Purchase Order Price (Most Recent) - tdpcg.tpsp.order.average Purchase Order Price (Average) Fields that are available to be used in this Process Extension: - When variable proc_ext_skip_calc_po_lines_is_history_line = false: The primary key fields of table tdpur401 (Purchase Order Lines): - tdpur401.orno Purchase Order - tdpur401.pono Line - tdpur401.sqnb Sequence - When variable proc_ext_skip_calc_po_lines_is_history_line = true: The primary key fields of table tdpur451 (Purchase Order Line History): - tdpur451.orno Purchase Order - tdpur451.pono Line - tdpur451.sqnb Sequence - tdpur451.trdt Transaction Date - tdpur451.ckor Record Type - tdpur451.sern Sequence Number Note: tables and external variables must also be declared in the Process Extension. The Calculate Target Prices logic processes Purchase Order Lines for a range of Items and Order Dates when one or more of the Search Options "Purchase Order Price (Lowest)", "Purchase Order Price (Most Recent)" and "Purchase Order Price (Average)" are selected in session 'Calculate Target Prices' (tdpcg0233m100). With this Process Extension it is possible to skip Purchase Order Lines or all lines of a Purchase Order for each of these Search Options. When the option "Include History" is checked, the Purchase Order Line History is also processed. Pseudocode: Below you can find an example: Hook: Declarations table ttdpur400 table ttdpur401 table ttdpur450 table ttdpur451 extern boolean proc_ext_skip_calc_po_lines_is_history_line extern domain tdpcg.tpsp proc_ext_skip_calc_po_lines_search_criterion Hook: ext.skip function extern boolean ext.skip() { on case proc_ext_skip_calc_po_lines_search_criterion case tdpcg.tpsp.order.lowest: * Skip Order line based on Order header data * - Read Purchase Order (tdpur400) if * proc_ext_skip_calc_po_lines_is_history_line = false </pre>

	<pre> * using tdpur401.orno * - Read Purchase Order History (tdpur450) if * proc_ext_skip_calc_po_lines_is_history_line = true * using tdpur451.orno Read order data (tdpur400/tdpur450) if <condition on tdpur400/tdpur450 = true> then return(true) endif AND / OR * Skip order line based on Order Line data * - Read Purchase Order Lines (tdpur401) if * proc_ext_skip_calc_po_lines_is_history_line = false * using tdpur401.orno * tdpur401.pono * tdpur401.sqnb * - Read Purchase Order Line History (tdpur451) if * proc_ext_skip_calc_po_lines_is_history_line = true * using tdpur451.orno * tdpur451.pono * tdpur451.sqnb * tdpur451.trdt * tdpur451.ckor * tdpur451.sern Read order line (history) data (tdpur401/tdpur451) if <condition on tdpur401/tdpur451 = true> then return(true) endif break case tdpcg.tpsp.order.most.rec: Specify another condition or use the same as above break case tdpcg.tpsp.order.average: Specify another condition or use the same as above break default: break endcase return(false) } </pre>

Process Extensions for TaxDeclaration

The following process extension(s) is/are available:

[TaxDeclaration.GetXMLTextTagContent](#)

TaxDeclaration.GetXMLTextTagContent

Get content for an XML Text Tag for the Tax Declaration.

This process extension is available from [KB3572604](#).

Technical information for this process extension:

Usage:	With this Process Extension, it is possible for the extender to re-determine and fill the Tag Content (field tfgld121.tcon) for a specified XML Tag Name (field tfgld121.xmlt) based on the parameters passed to the Process Extension method. E.g. for tax country Slovenia the tag TaxPeriodStart must be filled with the start date of the tax declaration period. - i.xml.tag.name = "TaxPeriodStart" - io.tag.content = "2025-05-01" E.g. for tax country Slovenia the tag TaxPeriodEnd must be filled with the end date of the tax declaration period. - i.xml.tag.name = "TaxPeriodEnd" - io.tag.content = "2025-05-31"

To implement this process extension, you need to implement the following method(s):

tfext.gld0001.get.xml.text.tag.content.for.tax.declaration

Syntax:	<pre>long tfext.gld0001.get.xml.text.tag.content.for.tax.declaration(domain tcncmp i.financial.company, domain tfgld.txdt i.tax.declaration.type, domain tctax.txcg i.tax.category, domain tccom.bpid i.collection.office, domain tfgld.tdvs i.tax.declaration.version, domain tfgld.year i.declaration.year, domain tfgld.prod i.declaration.period, domain tfgld.srno i.correction.number, domain tcccty i.tax.country, domain tcmcs.str132m i.xml.tag.name mb, ref domain tcmcs.str50m io.tag.content mb)</pre>
Usage:	Expl: Use this Process Extension method to re-determine and fill the Tag Content (field tfgld121.tcon) for a specified XML Tag Name (field tfgld121.xmlt) based on the parameters passed to the Process Extension method. Only if an implementation is done for this Process Extension and the Process Extension method is returning 0, the Tag Content (field tfgld121.tcon) is filled with the

	custom defined tag content (io.tag.content). (1) This Process Extension is called from session (tfgld1620m000) 'Tax Declaration Master' with satellite session (tfgld1121m000) 'Tax Positions by Tax Declaration Master' via form command 'Show file'. (2) This Process Extension is called from session (tfgld1625m000) 'Tax Declaration' via form command 'Show file'. (3) This Process Extension is called from session (tfgld1625m000) 'Tax Declaration' via form command 'Show ASCII as XML'. (4) This Process Extension is called from session (tfgld1625m000) 'Tax Declaration' via form command 'Transfer Tax Declaration'. Pre: N.A. Post: N.A. Input: - i.financial.company - Financial Company. - i.tax.declaration.type - Tax Declaration Type. (Only VAT is appl.) - i.tax.category - Tax Category. - i.collection.office - Collection Office. - i.tax.declaration.version - Tax Declaration Version. - i.declaration.year - Tax Declaration Year. Filled with 0 if called from session (tfgld1620m000) 'Tax Declaration Master'. Filled with tfgld125.vyer if called from session (tfgld1625m000) 'Tax Declaration'. - i.declaration.period - Tax Declaration Period. Filled with 0 if called from session (tfgld1620m000) 'Tax Declaration Master'. Filled with tfgld125.vprd if called from session (tfgld1625m000) 'Tax Declaration'. - i.correction.number - Correction Number. Filled with 0 if called from session (tfgld1620m000) 'Tax Declaration Master'. Filled with tfgld125.crno if called from session (tfgld1625m000) 'Tax Declaration'. - i.tax.country - Tax Country. Filled with tfgld120.ccty. - i.xml.tag.name - XML Tag Name. Filled with tfgld121.xmlt. IO: - io.tag.content - Tag Content. Filled with tfgld121.tcon. This field can be influenced by the process extension based on own logic e.g. see the example code below. Return: 0 - Success DALHOOKERROR - When an error occurs in getting the XML Text Tag content for the Tax Declaration. Example: <pre> if trim\$(i.tax.country) = "SVN" and trim\$(i.xml.tag.name) = "TaxPeriodStart" then * Determine the start date of the given declaration year & period. io.tag.content = "2025-05-01" endif </pre>

Process Extensions for WarehouseOrder

The following process extension(s) is/are available:

[WarehouseOrder.CreateNewHeader](#)

WarehouseOrder.CreateNewHeader

Allows to invoke a new order header during warehouse order creation.

This process extension is available from [KB2331636](#).

Technical information for this process extension:

Usage:	With this Process Extension it is possible trigger that a new warehouse order needs to be created instead of adding a line to an existing warehouse order.

To implement this process extension, you need to implement the following method(s):

whext.dll0012.create.new.warehouse.order

Syntax:	<pre>long whext.dll0012.create.new.warehouse.order(domain tckoor i.found.order.origin, domain tcorno i.found.order, domain tcwset i.found.order.set, domain tcpono i.order.line, domain tcpono i.order.sequence, ref boolean o.line.can.be.added.to.found.order)</pre>
Usage:	Expl: This process extension allows to include customer specific functionality during the creation of a warehouse order. This process extension will be called from Infor LN standard when a warehouse order is generated. The arguments started with "i.found" indicates a warehouse order header which is found as candidate to which a new order line will be linked. The arguments i.order.line and i.order.sequence indicate the warehouse order line which is to be created. Pre: N.a. Post: N.a. Input: i.found.order.origin - Order Origin i.found.order - Order Number i.found.order.set - Order Set i.order.line - Order Line i.order.sequence - Order Sequence Output: o.line.can.be.added.to.found.order - Indicator if a new order warehouse order needs to be created or not. Return: 0: Success / <> 0: Error
