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

Release 10.7

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 10.7

Publication date: November 28, 2025

Contents

About this guide **15**
Intended audience 15
Related documents 15
Contacting Infor 15

Chapter 1 Introduction **16**
Public Interfaces 16
Public Interfaces explained 16
How to request a new Public Interface 16
Public Interface example 17
Available Public Interfaces 18
Public Interface to call BDE methods 18
Proxy DLL 19
Coding example 19
Process Extensions 23
Process Extensions explained 23
How to request a new Process Extension 24
Available Process Extensions 24

Chapter 2 Public Interfaces for Extensibility **25**
Public Interfaces for Exception 25
Exception.Delete 25
Exception.GetMessage 25
Exception.NumberOfMessages 26
Public Interfaces for ProcessingOptionSet 26
ProcessingOptionSet.Create 27
ProcessingOptionSet.Delete 28

Chapter 3 Public Interfaces for Common **29**
Public Interfaces for Common 29
Common.ConvertAmount 29

Common.ConvertQuantity 30
Common.GenerateFirstFreeNumber 31
Common.GetConversionFactor 32
Common.GetCurrencyRates 32
Common.GetFinancialCompanyOfEntity 34
Common.GetFormattedAddress 34
Common.GetISOCodeOfLNLanguage 36
Common.GetParameters 36
Common.RoundAmount 37
Common.RoundQuantity 38
Common.ValidateNumberGroupSeries 38
Public Interfaces for HoldReason 39
HoldReasons.StartOverview 39

Chapter 4 Public Interfaces for Calendar 41

Public Interfaces for Calendar 41
Calendar.CalculateDaysFromHours 41

Chapter 5 Public Interfaces for Item 43

Public Interfaces for Item 43
Item.GetCostingData 43
Item.GetData 45
Item.GetOrderingData 46
Item.GetPurchaseData 47
Item.GetRevision 49
Item.GetSalesData 49
Item.GetServiceData 51
Item.GetSupplier 52
Item.GetWarehousingData 53
Public Interfaces for ItemSerial 55
ItemSerial.SetStatus 55

Chapter 6 Public Interfaces for CRM 56

Public Interfaces for Opportunity 56
Opportunity.StartMultiMain 56

Chapter 7 Public Interfaces for Sales 58

Public Interfaces for SalesQuote 58
SalesQuote.StartMultiMain 58

Public Interfaces for SalesOrder 59
SalesOrder.Approve 59
SalesOrder.Block 60
SalesOrder.ReleaseBlocking 61
Public Interfaces for SalesOrderLine 62
SalesOrderLine.Block 62
SalesOrderLine.ReleaseBlocking 63
Public Interfaces for SalesOrderLineComponent 64
SalesOrderLineComponent.Block 64
SalesOrderLineComponent.ReleaseBlocking 65

Chapter 8 Public Interfaces for Commissions and Rebates 67

Public Interfaces for Rebate 67
Rebates.StartOverview 67
Public Interfaces for Commission 69
Commissions.StartOverview 69

Chapter 9 Public Interfaces for Purchase 71

Public Interfaces for PurchaseRequisition 71
PurchaseRequisition.ApproveApprovalRecord 71
Public Interfaces for PurchaseRequisitionLine 72
PurchaseRequisitionLine.ConvertToPurchaseOrder 72
PurchaseRequisitionLine.ConvertToPurchaseRFQ 73
Public Interfaces for PurchaseContract 75
PurchaseContract.GetTotalAmounts 75
Public Interfaces for PurchaseOrder 76
PurchaseOrder.PrintPurchaseOrder 76
PurchaseOrder.StartAutomaticProcessing 78
Public Interfaces for PurchaseOrderLine 78
PurchaseOrderLine.Approve 79
PurchaseOrderLine.Cancel 79
PurchaseOrderLine.Split 82

Chapter 10 Public Interfaces for Pricing 84

Public Interfaces for Pricing 84
Pricing.SimulatePurchasePrice 84
Pricing.SimulateSalesPrice 87

Chapter 11 Public Interfaces for Planning 91

Public Interfaces for PlannedPeggingRelations 91

PlannedPeggingRelations.FindDemandForOrder 91

PlannedPeggingRelations.FindSupplyForOrder 93

Chapter 12 Public Interfaces for Standard Costs 96

Public Interfaces for StandardCosts 96

StandardCosts.Calculate 96

StandardCosts.CalculateForNewItem 98

Chapter 13 Public Interfaces for Manufacturing Master 99

Public Interfaces for ProductionBillOfMaterial 99

ProductionBillOfMaterial.ApproveRevision 99

ProductionBillOfMaterial.CopyToJobShop 100

ProductionBillOfMaterial.GenerateProductSubcontractingModel 101

ProductionBillOfMaterial.ValidateRevision 102

Chapter 14 Public Interfaces for Job Shop 103

Public Interfaces for JobShopBillOfMaterial 103

JobShopBillOfMaterial.ApproveRevision 103

JobShopBillOfMaterial.Copy 104

JobShopBillOfMaterial.ExpireRevision 105

JobShopBillOfMaterial.Explode 106

JobShopBillOfMaterial.SetUseForCosting 108

JobShopBillOfMaterial.SetUseForPlanning 109

JobShopBillOfMaterial.ValidateRevision 109

Public Interfaces for JobShopRouting 110

JobShopRouting.ApproveRevision 111

JobShopRouting.ClearUseForPlanning 111

JobShopRouting.Copy 112

JobShopRouting.CopyOperations 113

JobShopRouting.SetUseForCosting 115

JobShopRouting.SetUseForPlanning 115

Public Interfaces for ProductVariant 116

ProductVariant.GenerateProductStructureWithoutProject 116

ProductVariant.GenerateProductStructureWithoutProjectV2 117

ProductVariant.GenerateStructureV2 118

Public Interfaces for ProductVariantOptions 119

ProductVariantOptions.StartOverview 119
Public Interfaces for ProductionOrder 120
ProductionOrder.FinishOperationAddition 120
ProductionOrder.InitiateMaterialInventoryIssue 121
ProductionOrder.ReportProductV2 121
ProductionOrder.StartOperationAddition 123
ProductionOrder.StartReportOperationsCompleted 125
ProductionOrder.StartReportOrdersCompleted 126
Public Interfaces for ProductionOrderOperation 127
ProductionOrderOperation.GenerateSubcontractingPurchaseDocument 127
ProductionOrderOperation.ResetStatus 129
ProductionOrderOperation.SetStatusCompleted 129

Chapter 15 Public Interfaces for Subcontracting 132

Public Interfaces for SubcontractingModel 132
SubcontractingModel.ApproveRevision 132
SubcontractingModel.SetUseForCosting 133
SubcontractingModel.SetUseForPlanning 134
SubcontractingModel.ValidateRevision 135

Chapter 16 Public Interfaces for Manufacturing Project 136

Public Interfaces for ProjectPCS 136
ProjectPCS.CalculateCost 136
ProjectPCS.CopyCustomizedProductStructureToCustomizedStructure 137
ProjectPCS.CopyCustomizedProductStructureToStandardStructure 139
ProjectPCS.CopyStandardProductStructureToCustomizedStructure 141

Chapter 17 Public Interfaces for Warehousing 144

Public Interfaces for AdjustmentOrder 144
AdjustmentOrder.Process 144
Public Interfaces for Inventory 144
Inventory.ChangeMAUC 145
Public Interfaces for StockPoint 147
StockPoint.Move 147
Public Interfaces for HandlingUnit 149
HandlingUnit.StartAutomaticInboundProcessing 149
Public Interfaces for InboundOrderLineActivity 150
InboundOrderLineActivity.ModifyAutomaticProcessing 151

Public Interfaces for InboundAdvice 152
InboundAdvice.StartAutomaticInboundProcessing 152
Public Interfaces for InboundInspection 154
InboundInspection.StartAutomaticInboundProcessing 154
Public Interfaces for InboundOrderLine 156
InboundOrderLine.StartAutomaticInboundProcessing 156
Public Interfaces for InboundRun 158
InboundRun.StartAutomaticInboundProcessing 158
Public Interfaces for AllocationChangeOrder 159
AllocationChangeOrder.Process 160
Public Interfaces for Load 160
Load.StartAutomaticOutboundProcessing 160
Public Interfaces for Shipment 162
Shipment.StartAutomaticOutboundProcessing 162
Public Interfaces for OutboundAdvice 163
OutboundAdvice.StartAutomaticOutboundProcessing 164
Public Interfaces for OutboundRun 165
OutboundRun.StartAutomaticOutboundProcessing 165
Public Interfaces for QuarantineInventory 167
QuarantineInventory.StartAutomaticInboundProcessing 167
Public Interfaces for OutboundOrderLineActivity 169
OutboundOrderLineActivity.ModifyAutomaticProcessing 169
Public Interfaces for OutboundOrderLine 170
OutboundOrderLine.StartAutomaticOutboundProcessing 171
Public Interfaces for ReceiptLine 172
ReceiptLine.StartAutomaticInboundProcessing 172
Public Interfaces for StorageList 174
StorageList.StartAutomaticInboundProcessing 174

Chapter 18 Public Interfaces for Freight 177

Public Interfaces for FreightOrderCluster 177
FreightOrderCluster.ConfirmDelivery 177
Public Interfaces for FreightPlanning 178
FreightPlanning.ActualizePlan 178
FreightPlanning.GenerateForSpecificOrderSelection 179
FreightPlanning.MoveShipmentToLoad 181

Public Interfaces for Load 182
Load.ConfirmDelivery 182
Public Interfaces for Shipment 183
Shipment.ConfirmDelivery 183

Chapter 19 Public Interfaces for Taxation 185

Public Interfaces for Tax 185
Tax.Calculate 185

Chapter 20 Public Interfaces for BOD & BDE 190

Public Interfaces for BOD 190
BOD.ActionsAfterProcessingIncomingRequest 191
BOD.ActionsAfterProcessingIncomingRequestWithAutomaticProcessing 191
BOD.ActionsBeforeProcessingIncomingRequest 192
BOD.ComposeLocationId 193
BOD.ConvertFromERPDateToISODate 194
BOD.ConvertFromERPItem 195
BOD.ConvertFromISODateToERP 196
BOD.ConvertToERPItem 196
BOD.ExecuteMethod 197
BOD.ExecuteOnAcknowledgeForChameleon 198
BOD.ExecuteOnLoadForChameleon 199
BOD.ExecuteOnProcessForChameleon 199
BOD.ExecuteOnSyncForChameleon 200
BOD.ExecuteOnUpdateForChameleon 201
BOD.ExecutePublishForChameleon 201
BOD.ExecuteShowForChameleon 202
BOD.GetIdAccountingEntity 203
BOD.GetIdLocation 204
BOD.GetIdLogicalID 204
BOD.GetPublishingAllowed 205
BOD.HandleStagingAfterProcessingIncomingRequest 206
BOD.HandleStagingBeforeProcessingIncomingRequest 206
BOD.InterruptPublishing 207
BOD.Publish 208
BOD.PublishLNMessage 209
BOD.ResumePublishing 211
Public Interfaces for BDE 212

BDE.ExecuteMethod 212

Chapter 21 Public Interfaces for Service 213

Public Interfaces for Service 213
Service.DetermineStartOrFinishTime 213
Service.GetResourceCalendarCode 215

Public Interfaces for ServiceContract 216
ServiceContract.Activate 216
ServiceContract.GenerateConfigurationLines 218
ServiceContract.UnpackObjectReference 220

Public Interfaces for ServiceOrder 221
ServiceOrder.Close 221

Public Interfaces for WorkOrder 222
WorkOrder.Complete 222
WorkOrder.Release 225

Public Interfaces for PhysicalBreakdown 227
PhysicalBreakdown.ItemExistsInBreakdownOfParent 227

Chapter 22 Public Interfaces for Product Testing and Control 229

Public Interfaces for OrderInspection 229
OrderInspection.Close 229
OrderInspection.Complete 230
OrderInspection.Process 231

Chapter 23 Public Interfaces for Project 233

Public Interfaces for Project 233
Project.StartCopy 233

Public Interfaces for ProjectEstimate 235
ProjectEstimate.GenerateStructuralElements 235

Chapter 24 Public Interfaces for General Ledger 237

Public Interfaces for GeneralLedger 237
GeneralLedger.CheckDocumentIsBalanced 237

Chapter 25 Public Interfaces for Fixed Assets 238

Public Interfaces for AssetBook 238
AssetBook.CalculateRemainingDepreciationLife 238

Chapter 26 Public Interfaces for Object Configuration Management 240

Public Interfaces for Workflow 240
Workflow.DisableSubmit 240
Workflow.EnableSubmit 242
Workflow.Recall 242
Workflow.RecallByTypedObjectID 243
Workflow.Submit 244

Chapter 27 Public Interfaces for Time Management 245

Public Interfaces for ProductionOrderHours 245
ProductionOrderHours.CloseActiveLine 245
ProductionOrderHours.InsertActiveLine 246
ProductionOrderHours.StartOverview 247
Public Interfaces for GeneralHours 248
GeneralHours.StartOverview 248

Chapter 28 Public Interfaces for ZWF AG Shipment and Customs 251

Public Interfaces for SanctionList 251
SanctionList.Check 251
Public Interfaces for ExportControl 252
ExportControl.Check 252

Chapter 29 Process Extensions 254

Process Extensions for AssemblyOrder 254
AssemblyOrder.SkipDelete 254
Process Extensions for BillOfLading 254
BillOfLading.SkipShipmentLine 255
Process Extensions for BOD 255
BOD.PublishCustomBOD 256
tcext.bod0001.get.custom.bod.properties 257
tcext.bod0001.get.custom.bod.publish.function 258
tcext.bod0001.get.custom.bodnames 260
tcext.bod0001.get.initial.load.query 261
Process Extensions for ConfigurableItem 262
ConfigurableItem.SkipCompileConstraints 262
Process Extensions for ExceptionMessagesByItem 263
ExceptionMessagesByItem.SkipPrint 263
Process Extensions for HandlingUnitBuilding 263
HandlingUnitBuilding.CustomShipmentLine 263

whext.wmd0001.custom.shipment.line 264
Process Extensions for InboundAdvice 265
InboundAdvice.SkipPutawayAdvice 265
Process Extensions for IntrastatTransaction 266
IntrastatTransaction.CheckBlockedForReporting 266
lpext.ita0001.check.intrastat.transaction.blocked.for.reporting 266
Process Extensions for InventorySearchEngine 267
InventorySearchEngine.CheckHandlingUnit 268
whext.dll0002.check.handling.unit 268
InventorySearchEngine.CheckStockPoint 269
whext.dll0001.check.stock.point 269
Process Extensions for Invoice 270
Invoice.CustomComposingCriteriaMet 270
ciext.sli0003.invoice.custom.composing.criteria.met 271
Invoice.CustomXMLInvoicing 273
ciext.sli0001.define.custom.xml.data.elements 273
ciext.sli0001.get.value.for.custom.xml.data.element 275
Process Extensions for InvoiceFromBP 278
InvoiceFromBP.SkipPrintAgingAnalysis 278
InvoiceFromBP.SkipPrintOpenEntries 279
Process Extensions for InvoiceToBP 279
InvoiceToBP.SkipPrintAgingAnalysis 280
InvoiceToBP.SkipPrintOpenEntries 280
Process Extensions for ItemMasterPlan 281
ItemMasterPlan.SkipCopyToScenario 281
Process Extensions for OutboundAdvice 282
OutboundAdvice.SkipGeneratePickingList 282
OutboundAdvice.SkipPrint 282
OutboundAdvice.SkipReleaseAdvice 283
Process Extensions for OutboundOrderLine 283
OutboundOrderLine.SkipGenerateAdvice 283
Process Extensions for PaymentReceipt 284
PaymentReceipt.CustomXMLHandling 284
tfext.cmg0001.define.custom.payment.elements 284
tfext.cmg0001.define.custom.receipt.elements 286
tfext.cmg0001.get.value.for.custom.payment.element 287

tfext.cmg0001.get.value.for.custom.receipt.element 289
Process Extensions for Pricing 290
Pricing.RetrievePriceAndDiscounts 290
tdext.pcg0001.get.customer.determined.price.and.discounts 290
Process Extensions for Procurement 295
Procurement.DefaultPurchaseOffice 295
tdext.pur0001.get.customer.determined.default.purchase.office 296
Process Extensions for ProductionOrder 297
ProductionOrder.PrintCustomReports 297
tiext.sfc0001.check.print.condition 298
tiext.sfc0001.print.custom.report 299
tiext.sfc0001.register.custom.reports 301
tiext.sfc0001.update.print.status 302
ProductionOrder.SkipGenerateSubcDocuments 303
ProductionOrder.SkipPrintMaterialShortage 304
ProductionOrder.SkipReportComplete 304
ProductionOrder.SkipWarehouseForMaterialShortage 305
Process Extensions for ProductVariant 306
ProductVariant.AddCustomCPQIntegrationParameters 306
tiext.pcf0003.get.cpq.integration.parameter.value 306
tiext.pcf0003.register.cpq.integration.parameters 308
ProductVariant.AdjustPrice 308
tiext.pcf0001.adjust.variant.price 309
ProductVariant.SkipDeleteOrExpireJSBOM 309
tiext.pcs0002.product.variant.skip.delete.or.expire.jsbom 310
Process Extensions for ProjectPCS 311
ProjectPCS.SkipClose 311
tiext.pcs0001.project.pcs.skip.close 311
tiext.pcs0001.project.pcs.skip.close.implemented 312
Process Extensions for PurchaseOrder 312
PurchaseOrder.SkipApprove 312
PurchaseOrder.SkipPrintPurchaseInvoice 313
PurchaseOrder.SkipPrintPurchaseOrder 314
PurchaseOrder.SkipReleaseToWarehousing 314
Process Extensions for PurchaseOrderLine 315
PurchaseOrderLine.SkipPrintPurchaseInvoice 316
PurchaseOrderLine.SkipPrintPurchaseOrder 316

PurchaseOrderLine.SkipPrintPurchaseOrderReminder 317
Process Extensions for PurchaseSelfBilledInvoice 318
PurchaseSelfBilledInvoice.CustomCompose 318
tfext.acp0001.set.custom.composing.fields 318
Process Extensions for QualityManagement 319
QualityManagement.DefaultStandardTestProcedure 319
qmext.ptc0001.get.customer.determined.default.standard.test.procedure 320
Process Extensions for ReservedApprovedCommissionsRebates 321
ReservedApprovedCommissionsRebates.SkipPrint 321
Process Extensions for ReservedCommissionsRebates 322
ReservedCommissionsRebates.SkipReservationApproval 323
Process Extensions for SalesCheckInventory 323
SalesCheckInventory.CustomSorting 324
tdext.sls0001.get.customer.defined.order.by 324
Process Extensions for SalesOrder 325
SalesOrder.SkipApprove 325
SalesOrder.SkipPrint 326
SalesOrder.SkipPrintAcknowledgement 326
SalesOrder.SkipReleaseToWarehousing 327
Process Extensions for SalesOrderLine 329
SalesOrderLine.SkipPrint 329
SalesOrderLine.SkipPrintAcknowledgement 330
Process Extensions for SalesQuote 330
SalesQuote.SkipPrintSalesQuote 331
Process Extensions for SalesQuoteLine 331
SalesQuoteLine.SkipPrintSalesQuote 332
Process Extensions for ServiceOrderActivity 332
ServiceOrderActivity.SkipComplete 332
Process Extensions for StandardCost 333
StandardCost.SkipCalculate 334
tiext.cpr0002.standard.cost.skip.calculate 334
tiext.cpr0002.standard.cost.skip.calculate.item 336

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. 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    tncmp         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 otcextextapi. 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 "otcextemmapi"

                 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 (10.7) 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()
{
```

```
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 ottextextapi 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 import 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 reponse 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 (10.7) 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 if 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 refers 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 KB3554192.

Syntax:

```
long ProcessingOptionSet.Create(
        ref             long            oProcessingOptionSet,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID,
                                        ... )
```

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():

```
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: -

Post: Use ProcessingOptionSet.Delete() to release memory for the allocated Processing Option Set.

Input: Variable Arguments - repetition of pairs:

OptionName	- option name of type string
OptionValue	- option value, type as defined by Public Interface usage.

Output:

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.

Return:

0	- successfully created
<> 0	- failed.

ProcessingOptionSet.Delete

DLL:	tcextextapi This function is available from KB3554192.
Syntax:	long ProcessingOptionSet.Delete(ref long ioProcessingOptionSet)
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 Common

The following functions are available:

Common.ConvertAmount

Common.ConvertQuantity

Common.GenerateFirstFreeNumber

Common.GetConversionFactor

Common.GetCurrencyRates

Common.GetFinancialCompanyOfEntity

Common.GetFormattedAddress

Common.GetISOCodeOfLNLanguage

Common.GetParameters

Common.RoundAmount

Common.RoundQuantity

Common.ValidateNumberGroupSeries

Common.ConvertAmount

DLL:	tcextemmapi

Syntax:	`long Common.ConvertAmount(`
	`  domain tncmp 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 )
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:
                        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
                Output:
                        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.
                Return: 0                       - Amount is converted.
                        DALHOOKERROR            - Otherwise.
```

Common.ConvertQuantity

DLL: tcextcomapi

This function is available from KB2042397.

```
Syntax: long Common.ConvertQuantity(
                        domain  tcitem          iItem,
                        domain  tccuni          iFromUnit,
                        domain  tcqst1          iFromQuantity,
                        domain  tccuni          iToUnit,
                ref     domain  tcqst1          oToQuantity,
                ref     domain  tcmcs.s999m     oExceptionMessage mb,
                ref             long            oExceptionID )
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.GenerateFirstFreeNumber

DLL: tcextmcsapi

Syntax:

```
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 )
```

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 cashing is used.
        Post:   Transaction handling must be done when no cashing is used.
        Input:
                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)
        Output: 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.
        Return values:
                0                       - First Free Number is generated
                DALHOOKERROR            - on errors
```

Common.GetConversionFactor

DLL: tcextcomapi

This function is available from KB3562862.

Syntax:

```
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 )
```

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 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                       - Conversion Factor successfully
                                          retrieved.
                <> 0                    - Otherwise.
```

Common.GetCurrencyRates

DLL: tcextemmapi

This function is available from KB2080844.

Syntax:

```
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 )
```

Usage:

```
        Expl:   Get the rate data between a given transaction currency and the
                home currencies. The home currencies are defined in session
                "Companies" (tcemm1170m000).

                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 (tcemm1170m000)
                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 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                       - Currency rates are found.
                <> 0                    - Otherwise.
```

Common.GetFinancialCompanyOfEntity

DLL:	tcextemmapi
Syntax:	long Common.GetFinancialCompanyOfEntity(domain tncmp iLogisticCompany, domain tcemm.enty iEntityType, domain tcemm.enio iEntity, ref domain tncmp oFinancialCompany, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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 Exception to get all relevant information. Return: 0 - Financial company is found. DALHOOKERROR - Otherwise.

Common.GetFormattedAddress

DLL:	tcextcomapi
Syntax:	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)
Usage:	Expl.: This functions formats an given addresscode into a formatted address array of a given number of lines. Pre: iAddress must exists in tccom130 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 formated
                                          address when it differs from
                                          iFromCountry.
                                          When left empty always the country
                                          information is used in the formated
                                          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".

                        Moonen Shipyards BV
                        Kade 35
                        1780 AA Den Helder

                        The addresses outside "NLD" will have country information in
                        the formatted address:

                        Moonen F›¼□ördertechnik GmbH
                        Balhorner Feld 1
                        33106 Paderborn
                        Deutschland

                if i.from.country = empty both addresses will have country
                information in the formatted address:

                        Moonen Shipyards BV
                        Kade 35
                        1780 AA Den Helder
                        Nederland

                        Moonen F›¼□ördertechnik 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
                DALHOOKERROR            - on errors
```

Common.GetISOCodeOfLNLanguage

DLL:	tcextmcsapi
Syntax:	long Common.GetISOCodeOfLNLanguage(domain tcclan iLNLanguage, boolean iForceReading, ref domain tcilng oISOLanguage, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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 DALHOOKERROR - on Errors

Common.GetParameters

DLL:	tcextcomapi This function is available from KB2074177.
Syntax:	long Common.GetParameters(domain tcncmp iCompany, domain tctabl.c iTableCode, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID, ...)
Usage:	Expl: This function reads one or more parameters from a 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. Example: When the 'Contract Line Step Size' and 'Log History' are needed from the Project Contract paramaeters, the call of this public interface is as follows: if Common.GetParameters(|* Fixed arguments: company, --> input

```
                                        "tpctm000",             --> input
                                        exception.message,      --> output
                                        exception.id,           --> output
                                |* Variable arguments:
                                        "clss",                 --> input
                                        step.size,              --> output
                                        "lghs",                 --> input
                                        log.history) <> 0 then  --> output
                        |* Error, do something
                        Exception.Delete(exception.id)
                endif

        Pre:    None
        Post:   None
        Input:
                iCompany          - Company: Mandatory
                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.
                DALHOOKERROR      - Error occurred.
```

Common.RoundAmount

DLL: tcextcomapi

Syntax:

```
long Common.RoundAmount(
                        double          iUnroundedAmount,
                domain  tcmcs.st14      iDomainName,
                domain  tcccur          iCurrency,
        ref             double          oRoundedAmount,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

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
                DALHOOKERROR            - on errors
```

Common.RoundQuantity

DLL: tcextcomapi

Syntax:
```
long Common.RoundQuantity(
                        double          iUnroundedQuantity,
                domain  tcmcs.st14      iDomainName,
                domain  tccuni          iUnit,
        ref             double          oRoundedQuantity,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

Usage:
```
        Expl.:  This function rounds a quantity dependending 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
                DALHOOKERROR            - on errors
```

Common.ValidateNumberGroupSeries

DLL: tcextmcsapi

This function is available from KB2201864.

Syntax:
```
long Common.ValidateNumberGroupSeries(
                domain  tcseri          iSeries,
                domain  tcnrgr          iNumberGroup,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

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 HoldReason

The following functions are available:

HoldReasons.StartOverview

HoldReasons.StartOverview

DLL: tcextmcsapi

This function is available from KB2257724.

Syntax:

```
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 )
```

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.

Chapter 4 Public Interfaces for Calendar

Public Interfaces for Calendar

The following functions are available:

Calendar.CalculateDaysFromHours

Calendar.CalculateDaysFromHours

DLL:	tcextccpapi This function is available from KB3568051.
Syntax:	long Calendar.CalculateDaysFromHours(domain tcccp.ract iAvailabilityType, double iHours, domain tcutcs iEffectiveDate, ref double oDays, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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.

Chapter 5 Public Interfaces for Item

Public Interfaces for Item

The following functions are available:

Item.GetCostingData

Item.GetData

Item.GetOrderingData

Item.GetPurchaseData

Item.GetRevision

Item.GetSalesData

Item.GetServiceData

Item.GetSupplier

Item.GetWarehousingData

Item.GetCostingData

DLL:	tiextcprapi This function is available from KB2066826.

Syntax:	`long Item.GetCostingData(`
	`  domain  tcncmp  iLogisticCompany,`
	`  domain  tcitem  iItem,`
	`  domain  tcemm.grid  iEnterpriseUnit,`
	`  boolean  iForceRead,`
	`  ref  domain  tcmcs.s999m  oExceptionMessage mb,`
	`  ref  long  oExceptionID,`
	`  ... )`

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.

                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              -

                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,
                the function must be called as follows:

                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:    None
        Post:   None
        Input:
                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.
```

```
        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
                DALHOOKERROR            - Otherwise.
```

Item.GetData

DLL: tcextibdapi

This function is available from KB2048310.

Syntax:

```
long Item.GetData(
                domain  tcncmp          iLogisticCompany,
                domain  tcitem          iItem,
                domain  tcsite          iSite,
                        boolean         iForceRead,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID,
                                        ... )
```

Usage:

```
        Expl:   Retrieve data from Items
                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 project and customized are needed, the
                function must be called as follows:

                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.
                DALHOOKERROR            - Otherwise.
```

Item.GetOrderingData

DLL:	tcextibdapi This function is available from KB2043720.

Syntax:	`long Item.GetOrderingData(`
	`domain tcncmp iLogisticCompany,`
	`domain tcitem iItem,`
	`domain tcsite iSite,`
	`boolean iForceRead,`
	`ref domain tcmcs.s999m oExceptionMessage mb,`
	`ref long oExceptionID,`
	`... )`

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:

```
                if Item.GetData(
                                        |* 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.
                DALHOOKERROR            - Otherwise.
```

Item.GetPurchaseData

DLL:	tdextipuapi

Syntax:

```
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,
                                        ... )
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:

                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:    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
                DALHOOKERROR            - Otherwise.
```

Item.GetRevision

DLL: tcextibdapi

This function is available from KB2295055.

Syntax:

```
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 )
```

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 Revison
                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:

```
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,
```

```
                                                ... )
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:

                        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:    None
                Post:   None
                Input:
                        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.
                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
                DALHOOKERROR            - Otherwise.
```

Item.GetServiceData

DLL: tsextmdmapi

Syntax:

```
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,
                                        ... )
```

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:

                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:    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
                                          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
                DALHOOKERROR            - Otherwise.
```

Item.GetSupplier

DLL:	tdextipuapi

Syntax:	`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 )`

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: oBusinessPartnerFoundOnItemBPLevel
                                          True: Found on Item BP level.
                                          False: No BP found or not found on
                                          Item - BP level.
                oBuyFromBusinessPartner - Buy-from Business Partner
                oShipFromBusinessPartner - Ship-from Business Partner
        Return: 0                       - Data read
                DALHOOKERROR            - Otherwise.
```

Item.GetWarehousingData

DLL:	whextwmdapi

Syntax:	`long Item.GetWarehousingData(`
	`domain tcncmp iLogisticCompany,`
	`domain tcitem iItem,`
	`domain tcsite iSite,`
	`boolean iForceRead,`
	`ref domain tcmcs.s999m oExceptionMessage mb,`
	`ref long oExceptionID,`
	`... )`

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                       - Data read
                DALHOOKERROR            - Otherwise.
```

Public Interfaces for ItemSerial

The following functions are available:

ItemSerial.SetStatus

ItemSerial.SetStatus

DLL:	tcextibdapi This function is available from KB2331329.

Syntax:	`long ItemSerial.SetStatus(`
	`  domain  tcitem  iItem,`
	`  domain  tcibd.sern  iSerialNumber,`
	`  domain  tcibd.ssts  iStatus,`
	`  ref  domain  tcmcs.s999m  oExceptionMessage mb,`
	`  ref  long  oExceptionID )`

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:`
	`  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.`

Chapter 6 Public Interfaces for CRM

Public Interfaces for Opportunity

The following functions are available:

Opportunity.StartMultiMain

Opportunity.StartMultiMain

DLL:	tdextsmiapi This function is available from KB2127961.
Syntax:	long Opportunity.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcopty iOpportunity, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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 Exception to get all relevant information.
Return:	0	Session started
	<> 0	An error occurred

Chapter 7 Public Interfaces for Sales

Public Interfaces for SalesQuote

The following functions are available:

SalesQuote.StartMultiMain

SalesQuote.StartMultiMain

DLL:	tdextslsapi This function is available from KB2127961.
Syntax:	long SalesQuote.StartMultiMain(long iStartMode, domain tcmcs.st30 iStartFilter, const string iQueryExtend(), domain tcqono iSalesQuote, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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 — Session started <> 0 — An error occurred

Public Interfaces for SalesOrder

The following functions are available:

SalesOrder.Approve

SalesOrder.Block

SalesOrder.ReleaseBlocking

SalesOrder.Approve

DLL:	tdextslsapi This function is available from KB2200009.
Syntax:	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)
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:	`long SalesOrder.Block(`
	`domain tcorno iSalesOrder,`
	`domain tcmcs.hrea iHoldReason,`
	`domain tctrns.date iExpectedReleaseDate,`
	`ref domain tcmcs.s999m oExceptionMessage mb,`
	`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.ReleaseBlocking

DLL:	tdextslsapi This function is available from KB2223400.

Syntax:	`long SalesOrder.ReleaseBlocking(`
	`domain tcorno iSalesOrder,`
	`domain tcmcs.hrea iHoldReason,`
	`domain tdsls.rlty iReleaseType,`
	`ref domain tcmcs.s999m oExceptionMessage mb,`
	`ref long oExceptionID )`

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
```

Public Interfaces for SalesOrderLine

The following functions are available:

SalesOrderLine.Block

SalesOrderLine.ReleaseBlocking

SalesOrderLine.Block

DLL: tdextslsapi

This function is available from KB2257724.

Syntax:

```
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 )
```

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.
                                        If more than one  message is given,
                                        these are present 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
```

SalesOrderLine.ReleaseBlocking

DLL: tdextslsapi

This function is available from KB2223400.

Syntax:

```
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 )
```

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
```

Public Interfaces for SalesOrderLineComponent

The following functions are available:

SalesOrderLineComponent.Block

SalesOrderLineComponent.ReleaseBlocking

SalesOrderLineComponent.Block

DLL: tdextslsapi

This function is available from KB2257724.

Syntax:

```
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 )
```

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 - 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

SalesOrderLineComponent.ReleaseBlocking

DLL:	tdextslsapi This function is available from KB2223400.
Syntax:	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)
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

Chapter 8 Public Interfaces for Commissions and Rebates

Public Interfaces for Rebate

The following functions are available:

Rebates.StartOverview

Rebates.StartOverview

DLL:	tdextcmsapi This function is available from KB2262331.
Syntax:	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)
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
                                        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
                                        value is not equal to 0.
                                        If more than one  message is
                                        given, these are present 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 Commission

The following functions are available:

Commissions.StartOverview

Commissions.StartOverview

DLL:	tdextcmsapi This function is available from KB2262331.
Syntax:	long Commissions.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)
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
                                    value is not equal to 0.
                                    If more than one  message is
                                    given, these are present 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 9 Public Interfaces for Purchase

Public Interfaces for PurchaseRequisition

The following functions are available:

PurchaseRequisition.ApproveApprovalRecord

PurchaseRequisition.ApproveApprovalRecord

DLL:	tdextpurapi This function is available from KB2185794.
Syntax:	long PurchaseRequisition.ApproveApprovalRecord(domain tcrqno iPurchaseRequisition, domain tcsern iSequenceNumber, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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 - The Approval Progress record is approved.

	<> 0 - An error occurred

Public Interfaces for PurchaseRequisitionLine

The following functions are available:

PurchaseRequisitionLine.ConvertToPurchaseOrder

PurchaseRequisitionLine.ConvertToPurchaseRFQ

PurchaseRequisitionLine.ConvertToPurchaseOrder

DLL:	tdextpurapi This function is available from KB2185794.
Syntax:	long PurchaseRequisitionLine.ConvertToPurchaseOrder(domain tcrqno iPurchaseRequisition, domain tcpono iRequisitionLine, 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 This function is available from KB2185794.

Syntax:

```
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 )
```

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

Public Interfaces for PurchaseContract

The following functions are available:

PurchaseContract.GetTotalAmounts

PurchaseContract.GetTotalAmounts

DLL:	tdextpurapi This function is available from KB2042557.
Syntax:	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)
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. DALHOOKERROR - Error occurred.

Public Interfaces for PurchaseOrder

The following functions are available:

PurchaseOrder.PrintPurchaseOrder

PurchaseOrder.StartAutomaticProcessing

PurchaseOrder.PrintPurchaseOrder

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

        NAME                                    TYPE                    DEFAULT

        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.quan
        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
        PrintAppendixAdditionalInformation      domain tcyesno          tcyesno.yes
        PrintAppendixLettersOfCredit            domain tcyesno          tcyesno.no
        PrintToPredefinedDevice                 domain tcyesno          tcyesno.yes
        PrintingDevice                          domain tcmcs.str14      ""
        PrintingFileoutPathAndName              domain tcmcs.str100     ""

        - 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.StartAutomaticProcessing

DLL:	tdextpurapi This function is available from KB2225144.
Syntax:	long PurchaseOrder.StartAutomaticProcessing(domain tcorno iPurchaseOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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

Public Interfaces for PurchaseOrderLine

The following functions are available:

PurchaseOrderLine.Approve

PurchaseOrderLine.Cancel

PurchaseOrderLine.Split

PurchaseOrderLine.Approve

DLL: tdextpurapi

This function is available from KB3544566.

Syntax:
```
long PurchaseOrderLine.Approve(
                domain  tcorno          iPurchaseOrder,
                domain  tcpono          iOrderLine,
                domain  tcpono          iSequence,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        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.Cancel

DLL: tdextpurapi

This function is available from KB2225144.

Syntax:
```
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,
                domain  tcmcs.str2      iSalesOrderLineAcknowledgementCode,
                domain  tcmcs.str8      iSalesOrderLineChangeOrderSequence,
                domain  tccdis          iSalesOrderLineChangeReason,
                domain  tccdis          iSalesOrderLineChangeType,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

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:
                  'PurchaseOrderLine.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
                      'PurchaseOrderLine.StartAutomaticProcessing'.
        Pre:    Caller must set retry-point
        Post:   Caller must commit/abort transaction
        Input:  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
                                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, these are present 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(s) is/are cancelled. <> 0 An error occurred

PurchaseOrderLine.Split

DLL:	tdextpurapi This function is available from KB3544357.
Syntax:	long PurchaseOrderLine.Split(domain tcorno iPurchaseOrder, domain tcpono iOrderLine, domain tcpono iOrderLineSequence, domain tcqrd1 iQuantityForNewDetail, domain tcdate iPlannedReceiptDateForNewDetail, domain tccdis iPegChangeReason, domain tcyesno iKeepConfirmedReceiptDates, domain tcyesno iApproveAndProcessChangeRequestAutomatically, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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: 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

Chapter 10 Public Interfaces for Pricing

Public Interfaces for Pricing

The following functions are available:

Pricing.SimulatePurchasePrice

Pricing.SimulateSalesPrice

Pricing.SimulatePurchasePrice

DLL:	tdextpcgapi This function is available from KB2042557.
Syntax:	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,
        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 )
```

Usage:

```
        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                - Logictic 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
                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; memeory 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
                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.
                DALHOOKERROR                    - Error occurred.
```

Pricing.SimulateSalesPrice

DLL:	tdextpcgapi This function is available from KB2042557.

Syntax:

```
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,
        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  tcprip          oTotalMaterialPriceSurcharges,
        ref     domain  tcprip          oTotalMaterialPrice,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

Usage:

```
        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
                iChannel                        - Channel
                iProductVariant                 - Product Variant
                iEffectivityUnit                - Effectivity Unit
                iSite                           - Site
                iPriceDate                      - Price Date; of zero, then
                                                  current date and time 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
        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; memeory 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.
                |*****************************************************************
                oExceptionMessage               - The last message if the return
                                                  value is not equal to 0.
                                                  If more than one  message is
                                                  given, these are present 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.
                DALHOOKERROR                    - Error occurred.
```

Chapter 11 Public Interfaces for Planning

Public Interfaces for PlannedPeggingRelations

The following functions are available:

PlannedPeggingRelations.FindDemandForOrder

PlannedPeggingRelations.FindSupplyForOrder

PlannedPeggingRelations.FindDemandForOrder

DLL:	cpextrrpapi This function is available from KB2305358.
Syntax:	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)
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:  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.

        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).
                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:

                        oParentTransactionArray oChildTransactionArray
```

```
                                0                       40
                                40                      50
                                50                      60
                                40                      70
                                70                      80

                        The transaction id's in the arrays refer to
                        table cprrp041 (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.
```

PlannedPeggingRelations.FindSupplyForOrder

DLL: cpextrrpapi

This function is available from KB2305358.

Syntax:

```
long PlannedPeggingRelations.FindSupplyForOrder(
                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 )
```

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).
                iOrderType              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).
                oChildScenarioArray     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:
                            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)

                            (N) = transaction number

                        Resulting Parent and Child Transaction arrays
                        when executing the public interface for the sls order:

                        oParentTransactionArray oChildTransactionArray
                                0                       10
                                10                      11
                                11                      12
                                10                      13
                                13                      12
```

	The transaction id's in the arrays refer to table cprrp041 (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.

Chapter 12 Public Interfaces for Standard Costs

Public Interfaces for StandardCosts

The following functions are available:

StandardCosts.Calculate

StandardCosts.CalculateForNewItem

StandardCosts.Calculate

DLL:	tiextcprapi This function is available from KB2135360.
Syntax:	long StandardCosts.Calculate(domain tcitem iItem, domain tcemm.grid iEnterpriseUnit, domain tcrvdt iWhichDateForRevaluate, domain tcdate 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)
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 revaluate 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)
                iWhichDateForRevaluate  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.
                iSpecificDate           Only applicable if iWhichDateForRevaluate
                                        has the value 'a specific date'.
                iDeleteHistory          Delete history, as defined in the
                                        Calculation Parameters (ticpr0100m000).
                iRevaluationReport      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.
                                        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.
                iDevice                 The device for the RevaluationReport.
                                        Mandatory when iRevaluationReport = True
                                        and iReportOpenMode = 0 and no device
                                        is opened yet.
                iLanguage               Optional. If empty then the user's
                                        language will be used.
                iReportOpenMode         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
        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.CalculateForNewItem

DLL: tiextcprapi

This function is available from KB2130365.

Syntax:

```
long StandardCosts.CalculateForNewItem(
                domain  tcitem          iItem,
                domain  tcemm.grid      iEnterpriseUnit,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

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
```

Chapter 13 Public Interfaces for Manufacturing Master

Public Interfaces for ProductionBillOfMaterial

The following functions are available:

ProductionBillOfMaterial.ApproveRevision

ProductionBillOfMaterial.CopyToJobShop

ProductionBillOfMaterial.GenerateProductSubcontractingModel

ProductionBillOfMaterial.ValidateRevision

ProductionBillOfMaterial.ApproveRevision

DLL:	tiextmfcapi This function is available from KB2189918.
Syntax:	long ProductionBillOfMaterial.ApproveRevision(domain tcitem iProduct, domain tibmrv iRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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. Return: 0 - PBOM Revision is set to Approved <> 0 - PBOM Revision could not be Approved

ProductionBillOfMaterial.CopyToJobShop

DLL:	tiextmfcapi This function is available from KB2155728.
Syntax:	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)
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

	Production BOM). This applies only when iCreateNewBillOfMaterial is no. Output: oGeneratedModel - The generated BOM model oGeneratedRevision - The generated 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 - PBOM copied succesfully <> 0 - PBOM could not be copied

ProductionBillOfMaterial.GenerateProductSubcontracting Model

DLL:	tiextmfcapi This function is available from KB2292734.
Syntax:	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)
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 - For future use. iSubcontractorWarehouse - Subcontractor Warehouse (Mandatory). iShipFromBusinessPartner - Ship From Business Partner (Mandatory) 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 succesfully generated. <> 0 - Generate failed.

ProductionBillOfMaterial.ValidateRevision

DLL:	tiextmfcapi This function is available from KB2189918.
Syntax:	long ProductionBillOfMaterial.ValidateRevision(domain tcitem iProduct, 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.

Chapter 14 Public Interfaces for Job Shop

Public Interfaces for JobShopBillOfMaterial

The following functions are available:

JobShopBillOfMaterial.ApproveRevision

JobShopBillOfMaterial.Copy

JobShopBillOfMaterial.ExpireRevision

JobShopBillOfMaterial.Explode

JobShopBillOfMaterial.SetUseForCosting

JobShopBillOfMaterial.SetUseForPlanning

JobShopBillOfMaterial.ValidateRevision

JobShopBillOfMaterial.ApproveRevision

DLL:	tiextbomapi This function is available from KB2151944.
Syntax:	long JobShopBillOfMaterial.ApproveRevision(domain tcsite iSite, domain tcitem iProduct, domain tibmdl iBillOfMaterialCode, domain tibmrv iRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: Use this public interface to approve the specified Job Shop Bill of Material Revision. 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 Main item (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                       - JS BOM Revision is set to Approved
                DALHOOKERROR            - JS BOM Revision could not be Approved
```

JobShopBillOfMaterial.Copy

DLL: tiextbomapi

This function is available from KB2156270.

Syntax:

```
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 )
```

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:  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
                        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.ExpireRevision

DLL:	tiextbomapi This function is available from KB2270392.

Syntax:	`long JobShopBillOfMaterial.ExpireRevision(`
	`domain tcsite iSite,`
	`domain tcitem iProduct,`
	`domain tibmdl iBillOfMaterialCode,`
	`domain tibmrv iRevision,`
	`ref domain tcmcs.s999m oExceptionMessage mb,`
	`ref long oExceptionID )`

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 This function is available from KB2221191.

Syntax:

```
long JobShopBillOfMaterial.Explode(
                domain  tcsite          iSite,
                domain  tcitem          iProduct,
                domain  tibmdl          iBillOfMaterialCode,
                domain  tibmrv          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  tcqiv1          oBomQuantity(),
        ref             long            oNumberOfMaterials,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

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.SetUseForCosting

DLL: tiextbomapi

This function is available from KB2156270.

Syntax:

```
long JobShopBillOfMaterial.SetUseForCosting(
                domain  tcsite          iSite,
                domain  tcitem          iProduct,
                domain  tibmdl          iBillOfMaterialCode,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

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

This function is available from KB2156270.

Syntax:

```
long JobShopBillOfMaterial.SetUseForPlanning(
                domain  tcsite          iSite,
                domain  tcitem          iProduct,
                domain  tibmdl          iBillOfMaterialCode,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

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.ValidateRevision

DLL: tiextbomapi

This function is available from KB2163805.

Syntax:

```
long JobShopBillOfMaterial.ValidateRevision(
                domain  tcsite          iSite,
                domain  tcitem          iProduct,
                domain  tibmdl          iBillOfMaterialCode,
                domain  tibmrv          iRevision,
                domain  tcyesno         iCheckUseUp,
                domain  tcqiv1          iThresholdQuantity,
```

	ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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 information. Use the functions in Exception to get all relevant information. Return: 0 — JS BOM Revision passed all the validation checks. <> 0 — 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.SetUseForCosting

JobShopRouting.SetUseForPlanning

JobShopRouting.ApproveRevision

DLL: tiextrouapi

This function is available from KB2163805.

Syntax:
```
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 )
```

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                Main item (not mandatory).
                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

	This function is available from KB2156270.

Syntax:	`long JobShopRouting.ClearUseForPlanning(`
	`  domain  tcsite  iSite,`
	`  domain  tcitem  iProduct,`
	`  domain  tirou.rouc  iJobShopRouting,`
	`ref  domain  tcmcs.s999m  oExceptionMessage mb,`
	`ref  long  oExceptionID )`

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 This function is available from KB2156270.

Syntax:	`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 )`

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:  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
        Output: 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.
                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 Routing could not be
                        copied.
```

JobShopRouting.CopyOperations

DLL: tiextrouapi

This function is available from KB2202744.

Syntax:

```
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 )
```

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:  iSourceSite             Original Site (Mandatory).
                iSourceProduct          Original Product (Mandatory).
                iSourceRouting          Original Job Shop Routing (Mandatory).
                iSourceRevision         Original Revision (Mandatory).
                iFromOperation          Start Operation range to be copied (Mandatory).
                iToOperation            End Operations to be copied (Mandatory).
                iTargetProduct          Target Product (Mandatory).
                iTargetRouting          Target Job Shop Routing (Mandatory).
                iTargetRevision         Target Revision (Mandatory).
                iAppend                 True (append) or False (overwrite).
                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.
        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.
```

JobShopRouting.SetUseForCosting

DLL:	tiextrouapi This function is available from KB2156270.
Syntax:	long JobShopRouting.SetUseForCosting(domain tcsite iSite, domain tcitem iProduct, domain tirou.rouc iJobShopRouting, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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 This function is available from KB2156270.
Syntax:	long JobShopRouting.SetUseForPlanning(domain tcsite iSite, domain tcitem iProduct, domain tirou.rouc iJobShopRouting, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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 Exception to get all relevant information. Return: 0 - Use for Planning set to true. <> 0 - Otherwise.

Public Interfaces for ProductVariant

The following functions are available:

ProductVariant.GenerateProductStructureWithoutProject

ProductVariant.GenerateProductStructureWithoutProjectV2

ProductVariant.GenerateStructureV2

ProductVariant.GenerateProductStructureWithoutProject

DLL:	tiextpcfapi This function is available from KB2295738.
Syntax:	long ProductVariant.GenerateProductStructureWithoutProject(domain tccpva iProductVariant, domain tcdate iReferenceDate, ref domain tcitem oItem, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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. 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:

```
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 )
```

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.GenerateStructureV2

DLL: tiextpcfapi

This function is available from KB3500971.

Syntax:

```
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,
        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.
                iCheckStock             - Check the Standard Item Inventory.
        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.
```

Public Interfaces for ProductVariantOptions

The following functions are available:

ProductVariantOptions.StartOverview

ProductVariantOptions.StartOverview

DLL:	tiextpcfapi This function is available from KB2283720.
Syntax:	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)
Usage:	Expl: This function starts the session Product Variant Options in overview mode (tipcf5520m000). 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 iProductVariant The Product Variant for which the session will be started. Mandatory. iOptionSet The Option Set 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.

Public Interfaces for ProductionOrder

The following functions are available:

ProductionOrder.FinishOperationAddition

ProductionOrder.InitiateMaterialInventoryIssue

ProductionOrder.ReportProductV2

ProductionOrder.StartOperationAddition

ProductionOrder.StartReportOperationsCompleted

ProductionOrder.StartReportOrdersCompleted

ProductionOrder.FinishOperationAddition

DLL:	tiextsfcapi This function is available from KB3511148.
Syntax:	long ProductionOrder.FinishOperationAddition(domain tcsite iSite, domain tcpdno iProductionOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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.InitiateMaterialInventoryIssue

DLL:	tiextsfcapi This function is available from KB2298219.
Syntax:	long ProductionOrder.InitiateMaterialInventoryIssue(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcpono iMaterialPosition, domain tcutcs iIssueDate, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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.ReportProductV2

DLL:	tiextsfcapi

This function is available from KB2326926.

Syntax:

```
long ProductionOrder.ReportProductV2(
                domain  tcsite          iSite,
                domain  tcpdno          iProductionOrder,
                domain  tcclot          iLotCode,
                domain  tiqep2          iProducedQuantity,
                domain  tiqep2          iRejectedQuantity,
                domain  tccdis          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 )
```

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:  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. 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 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 reported quantities and / or status change is succssfully processed. <> 0 — Errors occurred.

ProductionOrder.StartOperationAddition

DLL:	tiextsfcapi This function is available from KB3511148.
Syntax:	long ProductionOrder.StartOperationAddition(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcopno iPrecedingOperation, domain tcopno iNextOperation, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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
                ProductionOrder.FinishOperationAddition() must be called to
                reset the system to normal.

                Start Example Usage:
                        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 )

                        |* 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.


        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.StartReportOperationsCompleted

DLL: tiextsfcapi

This function is available from KB2285288.

Syntax:

```
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,
        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:	tiextsfcapi This function is available from KB2288027.
Syntax:	long ProductionOrder.StartReportOrdersCompleted(long iStartMode, domain tcsite iSite, domain tcpdno iProductionOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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

Public Interfaces for ProductionOrderOperation

The following functions are available:

ProductionOrderOperation.GenerateSubcontractingPurchaseDocument

ProductionOrderOperation.ResetStatus

ProductionOrderOperation.SetStatusCompleted

ProductionOrderOperation.GenerateSubcontractingPurchaseDocument

DLL:	tiextsfcapi This function is available from KB2302527.
Syntax:	long ProductionOrderOperation.GenerateSubcontractingPurchaseDocument(domain tcsite iSite, domain tcpdno iProductionOrder, domain tcopno iOperation, domain tisfc.sdoc iGenerateRequisitionOrOrder, domain tcyesno iCheckPreviousOperations, domain tcyesno iCombineOperations, domain tcyesno iGenerateForBlockedOperations, domain tcyesno iCheckSubcontractorStatus, domain tcdate iStatusCheckDate, domain tcseri iPurchaseOrderSeries, domain tccotp iPurchaseOrderType, domain tcseri iPurchaseRequisitionSeries, ref domain tcorno oPurchaseOrder, ref domain tcrqno oPurchaseRequisition, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl.: Use this Public Interface to generate 1 Subcontracting Purchase Order or 1 Subcontracting Purchase Requisition for the input production order operation. The status of this operation must be less than completed. It must have a subcontracting work center. Transaction management is handled within this function. Errors which occurred during the actual subcontracting purchase requisition / order creation process are included in oExceptionID. 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 of Production Order - mandatory when site
                        concept is active; must be filled with site of
                        production order
                iProductionOrder
                        Production Order - mandatory; must be a valid
                        production order number
                iOperation
                        Operation - mandatory; must be filled with operation
                        number, which exists for Production Order
                iGenerateRequisitionOrOrder
                        Generate Requisition or Order - mandatory; must have
                        value 'Purchase Order' or 'Purchase Requisition'
                iCheckPreviousOperations
                        Check Previous Operations - mandatory; when set to
                        'Yes', it will check if previous operations
                        have been completed already
                iCombineOperations
                        Combine Operations - mandatory; when set to 'Yes',
                        adjacent operations, to be subcontracted to
                        same subcontractor, will be combined on 1
                        purchase order line
                iGenerateForBlockedOperations
                        Generate for Blocked Operations - mandatory; when
                        set to 'No', no purchase document will be
                        generated for blocked operations
                iCheckSubcontractorStatus
                        Check Subcontractor Status - mandatory; when set
                        to 'Yes', the status of the subcontractor
                        will be checked
                iStatusCheckDate
                        Status Check Date - mandatory when Check
                        Subcontractor Status is set to 'Yes'
                iPurchaseOrderSeries
                        Purchase Order Series - mandatory when Generate
                        Requisition or Order has been set to 'Purchase
                        Order'
                iPurchaseOrderType
                        Purchase Order Type - mandatory when Generate
                        Requisition or Order has been set to 'Purchase
                        Order'
                iPurchaseRequisitionSeries
                        Purchase Requisition Series - mandatory when
                        Generate Requisition or Order has been set to
                        'Purchase Requisition'
        Output: oPurchaseOrder
                        Purchase Order - number of generated purchase order
                oPurchaseRequisition
                        Purchase Requisition - number of generated purchase
                        requisition
                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       purchase order/requisition has been generated
                <> 0    order/requisition could not be generated
```

ProductionOrderOperation.ResetStatus

DLL: tiextsfcapi

This function is available from KB2232034.

Syntax:

```
long ProductionOrderOperation.ResetStatus(
                domain  tcsite          iSite,
                domain  tcpdno          iProductionOrder,
                domain  tcopno          iOperation,
                        boolean         iResetPlannedQuantities,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

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:  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.
        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                       Operation is reset.
                <> 0                    Operation could not be reset.
```

ProductionOrderOperation.SetStatusCompleted

DLL: tiextsfcapi

This function is available from KB2299421.

Syntax:

```
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 )
```

Usage:

```
        Expl:   With this Public Interface, the Production Order Operation
                status of the specified Operation will be set to .Completed..

                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.

Chapter 15 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:	long SubcontractingModel.ApproveRevision(domain tcsite iProductionSite, domain tcitem iProduct, domain tccom.bpid iSubcontractor, domain tccom.bpid iShipFromBusinessPartner, domain tirpt.revi iRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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).
                iShipFromBusinessPartner
                                        Ship From Business Partner (mandatory).
                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:

```
long SubcontractingModel.SetUseForCosting(
                domain  tcsite          iProductionSite,
                domain  tcitem          iProduct,
                domain  tccom.bpid      iSubcontractor,
                domain  tccom.bpid      iShipFromBusinessPartner,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

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).
                iShipFromBusinessPartner
                                        Ship From Business Partner (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 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:	long SubcontractingModel.SetUseForPlanning(domain tcsite iProductionSite, domain tcitem iProduct, domain tccom.bpid iSubcontractor, domain tccom.bpid iShipFromBusinessPartner, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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. This function follows the same logic as the form command "Use for Planning" 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). iShipFromBusinessPartner Ship From Business Partner (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 Use for Planning flag has been set. <> 0 The Use for Planning flag has not been set.

SubcontractingModel.ValidateRevision

DLL:	tiextsubapi This function is available from KB2282924.
Syntax:	long SubcontractingModel.ValidateRevision(domain tcsite iProductionSite, domain tcitem iProduct, domain tccom.bpid iSubcontractor, domain tccom.bpid iShipFromBusinessPartner, domain tirpt.revi iRevision, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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). iShipFromBusinessPartner — Ship From Business Partner (mandatory). 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 is valid. Non-blocking messages can still be returned. <> 0 — The Subcontracting Model Revision is not valid.

Chapter 16 Public Interfaces for Manufacturing Project

Public Interfaces for ProjectPCS

The following functions are available:

ProjectPCS.CalculateCost

ProjectPCS.CopyCustomizedProductStructureToCustomizedStructure

ProjectPCS.CopyCustomizedProductStructureToStandardStructure

ProjectPCS.CopyStandardProductStructureToCustomizedStructure

ProjectPCS.CalculateCost

DLL:	tiextpcsapi This function is available from KB2332148.

Syntax:	`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 )`

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 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                       Successful completion
                <> 0                    Failure
```

ProjectPCS.CopyCustomizedProductStructureToCustomizedStructure

DLL:	tiextpcsapi This function is available from KB3555140.

Syntax:	`long ProjectPCS.CopyCustomizedProductStructureToCustomizedStructure(`
	`domain tcsite iSite,`
	`domain tcitem iSourceCustomizedItem,`
	`domain tcitem iTargetCustomizedItem,`
	`domain tcyesno iUseProjectReferenceDate,`

```
                domain  tcdate          iReferenceDate,
                domain  tcyesno         iCopyAllComponentsAndEffectiveUnits,
                domain  tcuef.effn      iUnit,
                domain  ticpst          iCopyMethod,
                domain  tcyesno         iCopyEitemRelationships,
                domain  tcyesno         iCopyProductVariantStructure,
                domain  tcyesno         iIncludeAlternatives,
                domain  tcyesno         iIncludeUseUpMaterials,
                domain  tcyesno         iGenerateNewItemCodes,
                domain  tcyesno         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.CopyCustomizedProductStructureToStandardStructure

DLL: tiextpcsapi

This function is available from KB3555140.

Syntax:

```
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 )
```

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.CopyStandardProductStructureToCustomized Structure

DLL:	tiextpcsapi This function is available from KB3555140.
Syntax:	long ProjectPCS.CopyStandardProductStructureToCustomizedStructure(domain tcsite iSite, domain tcitem iStandardItem, domain tcitem iCustomizedItem, domain tcyesno iUseProjectReferenceDate, domain tcdate iReferenceDate, domain tcyesno iCopyAllComponentsAndEffectiveUnits, domain tcuef.effn iUnit, domain ticpst iCopyMethod, domain ticmpm iCopyModels, domain tcyesno iApproveModels, domain ticpit 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)
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.

Chapter 17 Public Interfaces for Warehousing

Public Interfaces for AdjustmentOrder

The following functions are available:

AdjustmentOrder.Process

AdjustmentOrder.Process

DLL:	whextinhapi This function is available from KB2069063.
Syntax:	long AdjustmentOrder.Process(domain tcorno iAdjustmentOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
Usage:	Expl: This API will process a given adjustment order. Pre: db.retry.point() Post: abort.transaction() or commit.transaction() Input: iAdjustmentOrder - The Adjustment Order which need to be processed. 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 succesfully DALHOOKERROR - Otherwise.

Public Interfaces for Inventory

The following functions are available:

Inventory.ChangeMAUC

Inventory.ChangeMAUC

DLL:	whextinaapi This function is available from KB2327731.
Syntax:	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 boolean oDataProcessed, ref domain tcccur oReportCurrency, ref domain tcccur oHomeCurrencyArray() fixed, ref domain tccopr oOldMAUC, 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. 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 iItem is
                                                set.
                                                Otherwise it should be empty.
                iProject
                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.
                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
```


Public Interfaces for StockPoint

The following functions are available:

StockPoint.Move

StockPoint.Move

DLL:	whextinrapi This function is available from KB2303316.
Syntax:	long StockPoint.Move(domain tccwar iWarehouse, domain tcitem iItem, domain whloca iOriginatingLocation, domain whhuid iHandlingUnit, domain tcclot iLot, domain tcinvt.date iInventoryDate, domain whwmd.pkdf iPackageDefinition, domain tcyesno iFullStockPoint, domain tcqst1 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)
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 - Optional 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 - Optional
                                    When set to tcyesno.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
                                    tcyesno.no.
                                    Should be greater than zero when
                                    iHandlingUnit is empty and
                                    iFullStockPoint is set to tcyesno.no.
          iStorageUnit            - Storage Unit
                                    Mandatory when iHandlingUnit is empty
                                    and iFullStockPoint is set to
                                    tcyesno.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.
          iDestinationLocation    - Destination Location - Mandatory
Output:   oStockPointMoved        - Stock Point Moved, true/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: Error
```

Public Interfaces for HandlingUnit

The following functions are available:

HandlingUnit.StartAutomaticInboundProcessing

HandlingUnit.StartAutomaticInboundProcessing

DLL:	whextinhapi This function is available from KB2117931.
Syntax:	long HandlingUnit.StartAutomaticInboundProcessing(domain whhuid iHandlingUnit, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)

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:
	
	--- 	
	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(..)

            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 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/DALHOOKERROR
```

Public Interfaces for InboundOrderLineActivity

The following functions are available:

InboundOrderLineActivity.ModifyAutomaticProcessing

InboundOrderLineActivity.ModifyAutomaticProcessing

DLL: whextinhapi

This function is available from KB2117931.

Syntax:
```
long InboundOrderLineActivity.ModifyAutomaticProcessing(
                domain  whinh.oorg      iOrderOrigin,
                domain  tcorno          iOrderNumber,
                domain  tcpono          iOrderLine,
                domain  tcpono          iOrderSequence,
                domain  whinh.prty      iProcedureType,
                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.prty.receipt.line)
                                                Inspection Procedure
                                                (whinh.prty.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 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/DALHOOKERROR

Public Interfaces for InboundAdvice

The following functions are available:

InboundAdvice.StartAutomaticInboundProcessing

InboundAdvice.StartAutomaticInboundProcessing

DLL:	whextinhapi This function is available from KB2117931.
Syntax:	long InboundAdvice.StartAutomaticInboundProcessing(domain tcorno iInboundAdvice, domain tcpono iInboundAdviceLine, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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:

                ---
                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 = 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/DALHOOKERROR

Public Interfaces for InboundInspection

The following functions are available:

InboundInspection.StartAutomaticInboundProcessing

InboundInspection.StartAutomaticInboundProcessing

DLL:	whextinhapi This function is available from KB2117931.
Syntax:	long InboundInspection.StartAutomaticInboundProcessing(domain tcorno iInspection, domain tcpono iInspectionSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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:

                ---
                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()

                |* 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 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/DALHOOKERROR
```

Public Interfaces for InboundOrderLine

The following functions are available:

InboundOrderLine.StartAutomaticInboundProcessing

InboundOrderLine.StartAutomaticInboundProcessing

DLL:	whextinhapi This function is available from KB2117931.
Syntax:	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)

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 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 = 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)
                ---

                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:
                ReceiptLine.StartAutomaticInboundProcess
                InboundInspection.StartAutomaticInboundProcess
                InboundAdvice.StartAutomaticInboundProcess
                InboundRun.StartAutomaticInboundProcess
                StorageList.StartAutomaticInboundProcess
                HandlingUnit.StartAutomaticInboundProcess
                QuarantineInventory.StartAutomaticInboundProcess

                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/DALHOOKERROR
```

Public Interfaces for InboundRun

The following functions are available:

InboundRun.StartAutomaticInboundProcessing

InboundRun.StartAutomaticInboundProcessing

DLL:	whextinhapi This function is available from KB2117931.

Syntax:	`long InboundRun.StartAutomaticInboundProcessing(`
	`domain whinh.btno iInboundRun,`
	`ref domain tcmcs.s999m oExceptionMessage mb,`
	`ref long oExceptionID )`

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:
	
	--- 	
	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 = 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/DALHOOKERROR
```

Public Interfaces for AllocationChangeOrder

The following functions are available:

AllocationChangeOrder.Process

AllocationChangeOrder.Process

DLL:	whextinhapi This function is available from KB3516844.
Syntax:	long AllocationChangeOrder.Process(domain tcorno iAllocationChangeOrder, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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 Load

The following functions are available:

Load.StartAutomaticOutboundProcessing

Load.StartAutomaticOutboundProcessing

DLL:	whextinhapi This function is available from KB2117931.
Syntax:	long Load.StartAutomaticOutboundProcessing(domain whinh.load iLoad, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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
                OutboundOrderLineActivity.ModifyAutomaticProcessing() for the
                activity that should be automatically processed.
                See below code example:

                ---
                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(..)

                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 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/DALHOOKERROR

Public Interfaces for Shipment

The following functions are available:

Shipment.StartAutomaticOutboundProcessing

Shipment.StartAutomaticOutboundProcessing

DLL:	whextinhapi This function is available from KB2117931.
Syntax:	long Shipment.StartAutomaticOutboundProcessing(domain whinh.shpm iShipment, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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: --- 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)
                ---

                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/DALHOOKERROR
```

Public Interfaces for OutboundAdvice

The following functions are available:

OutboundAdvice.StartAutomaticOutboundProcessing

OutboundAdvice.StartAutomaticOutboundProcessing

DLL: whextinhapi

This function is available from KB2117931.

Syntax:

```
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 )
```

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 OutboundOrderLineActivity.ModifyAutomaticProcessing() for the activity that should be automatically processed.
See below code example:

```
---
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()

|* 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/DALHOOKERROR
```

Public Interfaces for OutboundRun

The following functions are available:

OutboundRun.StartAutomaticOutboundProcessing

OutboundRun.StartAutomaticOutboundProcessing

DLL:	whextinhapi

	This function is available from KB2117931.
Syntax:	long OutboundRun.StartAutomaticOutboundProcessing(domain whinh.btno iOutboundRun, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)

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:
	
	--- 	
	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(..)
	
	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)
	--- 	
	
	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/DALHOOKERROR
```

Public Interfaces for QuarantineInventory

The following functions are available:

QuarantineInventory.StartAutomaticInboundProcessing

QuarantineInventory.StartAutomaticInboundProcessing

DLL: whextinhapi

This function is available from KB2117931.

Syntax:

```
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 )
```

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/DALHOOKERROR
```

Public Interfaces for OutboundOrderLineActivity

The following functions are available:

OutboundOrderLineActivity.ModifyAutomaticProcessing

OutboundOrderLineActivity.ModifyAutomaticProcessing

DLL: whextinhapi

This function is available from KB2117931.

Syntax:

```
long OutboundOrderLineActivity.ModifyAutomaticProcessing(
                domain  whinh.oorg      iOrderOrigin,
                domain  tcorno          iOrderNumber,
                domain  tcpono          iOrderLine,
                domain  tcpono          iOrderSequence,
                domain  whinh.prty      iProcedureType,
                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
                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:    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:
                                                "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/DALHOOKERROR
```

Public Interfaces for OutboundOrderLine

The following functions are available:

OutboundOrderLine.StartAutomaticOutboundProcessing

OutboundOrderLine.StartAutomaticOutboundProcessing

DLL:	whextinhapi This function is available from KB2117931.
Syntax:	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)

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:
	
	--- 	
	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()
	
	|* 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 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/DALHOOKERROR
```

Public Interfaces for ReceiptLine

The following functions are available:

ReceiptLine.StartAutomaticInboundProcessing

ReceiptLine.StartAutomaticInboundProcessing

DLL:	whextinhapi This function is available from KB2117931.
Syntax:	`long ReceiptLine.StartAutomaticInboundProcessing(`

```
                domain  whinh.shpm      iReceipt,
                domain  tcpono          iReceiptLine,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

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:

                ---
                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 = 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/DALHOOKERROR
```

Public Interfaces for StorageList

The following functions are available:

StorageList.StartAutomaticInboundProcessing

StorageList.StartAutomaticInboundProcessing

DLL: whextinhapi

This function is available from KB2117931.

Syntax:

```
long StorageList.StartAutomaticInboundProcessing(
                domain  whinh.btno      iInboundRun,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

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:

                ---
                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

                commit.transaction()

                return(ret.val)
                ---

                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/DALHOOKERROR

Chapter 18 Public Interfaces for Freight

Public Interfaces for FreightOrderCluster

The following functions are available:

FreightOrderCluster.ConfirmDelivery

FreightOrderCluster.ConfirmDelivery

DLL:	fmextlbdapi This function is available from KB2050777.
Syntax:	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)
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: Transaction handling is done within this function! Calling this Public Interface from inside a transaction is forbidden and will be blocked Post: Exception handling can be done, when needed. Input: iFreightOrderCluster - Freight Order Cluster: 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 Freight Order Cluster 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
                DALHOOKERROR            - No data found or error occured
```

Public Interfaces for FreightPlanning

The following functions are available:

FreightPlanning.ActualizePlan

FreightPlanning.GenerateForSpecificOrderSelection

FreightPlanning.MoveShipmentToLoad

FreightPlanning.ActualizePlan

DLL: fmextlbdapi

This function is available from KB2070843.

Syntax:

```
long FreightPlanning.ActualizePlan(
                domain  tcorno          iPlan,
                        boolean         iReplanAllowed,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

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                   - Plan which must be actualized:
                                          Mandatory
                iReplanAllowed          - Should potential freight order lines
                                          in the plan which must be replanned
                                          be replanned before actualization
                                          takes place, 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/DALHOOKERROR
```

FreightPlanning.GenerateForSpecificOrderSelection

DLL: fmextlbdapi

This function is available from KB2070843.

Syntax:

```
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,
                        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,
                        boolean         iShowProgressIndicator,
        ref     domain  tcorno          oPlan,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

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 normaly 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.
                iReplanAdditionThroughStatus
                                        - Replanning, allow updating of existing
```

```
                                        Loads/Shipments till this status.
                iReplanningMethod       - Replanning Method
                                          Possible Values
                                          - Replan Freight Order Lines Separatly
                                            (fmlbd.replan.separate)
                                          - Combine with Planned Freight Orders
                                            (fmlbd.replan.combine)
                iCommittedInventoryOnly - Plan for Committed Inventory Only
                iDetailedPlanningLog    - Plan with the detailed planning log
                iShowProgressIndicator  - 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: oPlan                   - The generated plan
                oExceptionMessage       - The last message if the return
                                          value is not equal to 0.
                                          If more than one  message is
                                          given, these are present in the
                                          oExceptionID
                oExceptionID            - An ID that refers to all error
                                          information. Use the functions in
                                          Exception to get all relevant
                                          information.
        Return: 0/DALHOOKERROR
```

FreightPlanning.MoveShipmentToLoad

DLL: fmextlbdapi

This function is available from KB2070843.

Syntax:

```
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 )
```

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:  iShipment               - Shipment which must be moved:
                                          Mandatory
                iDestinationLoad        - Load to which the shipment must be
                                          moved: Mandatory
                iAllowMergeShipmentLines- When iDestinationLoad already has a
                                          shipment line which has the same
                                          freight order line of any of the
                                          iSourceShipment lines, the lines must
                                          be merged. This flag determines if
                                          that should be allowed (true/false)
                iDeleteEmptyLoad        - 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/DALHOOKERROR
```

Public Interfaces for Load

The following functions are available:

Load.ConfirmDelivery

Load.ConfirmDelivery

DLL:	fmextlbdapi This function is available from KB2050777.

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 is 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
                DALHOOKERROR            - No data found or error occured
```

Public Interfaces for Shipment

The following functions are available:

Shipment.ConfirmDelivery

Shipment.ConfirmDelivery

DLL:	fmextlbdapi This function is available from KB2050777.
Syntax:	`long Shipment.ConfirmDelivery(` `domain  tcorno  iShipment,`

```
                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
                shipment and it's shipment 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:    Transaction handling is done within this function! Calling this
                Public Interface from inside a transaction is forbidden and will
                be blocked
        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
                DALHOOKERROR            - No data found or error occured
```

Chapter 19 Public Interfaces for Taxation

Public Interfaces for Tax

The following functions are available:

Tax.Calculate

Tax.Calculate

DLL:	tcexttaxapi This function is available from KB2233830.
Syntax:	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 tcficu 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, domain tcdate iTaxDate, domain tcorno iOrderNumber, domain tcpono iOrderLine, domain tcinvn iInvoiceNumber, domain tcmcs.long iInvoiceLine, 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,
                        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 )
```

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
                                          case of Purchase Tax Origins.
                                          (I/E)
                iFinancialBPGroup       - Financial BP Group
                                          (E)
                iWarehouse              - Warehouse
                                          (E)
```

```
                  iDepartment             - Department, fill this field only if the
Warehouse
                                            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                         Succes
                <> 0                      Error
```

Chapter 20 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.ConvertToERPItem

BOD.ExecuteMethod

BOD.ExecuteOnAcknowledgeForChameleon

BOD.ExecuteOnLoadForChameleon

BOD.ExecuteOnProcessForChameleon

BOD.ExecuteOnSyncForChameleon

BOD.ExecuteOnUpdateForChameleon

BOD.ExecutePublishForChameleon

BOD.ExecuteShowForChameleon

BOD.GetIdAccountingEntity

BOD.GetIdLocation

BOD.GetIdLogicalID

BOD.GetPublishingAllowed

BOD.HandleStagingAfterProcessingIncomingRequest

BOD.HandleStagingBeforeProcessingIncomingRequest

BOD.InterruptPublishing

BOD.Publish

BOD.PublishLNMessage

BOD.ResumePublishing

BOD.ActionsAfterProcessingIncomingRequest

DLL:	tcextbodapi This function is available from KB2040021.
Syntax:	long BOD.ActionsAfterProcessingIncomingRequest(long iXMLRequest, ref long oResult, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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 / DALHOOKERROR

BOD.ActionsAfterProcessingIncomingRequestWithAutomaticProcessing

DLL:	tcextbodapi This function is available from KB2040021.

Syntax:	`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 )`

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 / DALHOOKERROR

BOD.ActionsBeforeProcessingIncomingRequest

DLL:	tcextbodapi This function is available from KB2040021.

Syntax:	`long BOD.ActionsBeforeProcessingIncomingRequest(`
	`  long  iXMLRequest,`
	`  ref  boolean  oCancel,`
	`  ref  domain  tcmcs.s999m  oExceptionMessage mb,`
	`  ref  long  oExceptionID )`

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 - 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 oExceptionID - An ID that refers to all error information. Use the functions in Exception to get all relevant information. Return: 0 - No error occurred while executing the actions. DALHOOKERROR - 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:	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)
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:

```
long BOD.ConvertFromERPDateToISODate(
                domain  tcdate          iERPDate,
                        long            iDateConversion,
                        long            iDateFormat,
                        boolean         iERPDateIsUTC,
        ref     domain  tcmcs.str30     oISODate,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

Usage:

```
        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.
```

BOD.ConvertFromERPItem

DLL: tcextbodapi

This function is available from KB2044306.

Syntax:

```
long BOD.ConvertFromERPItem(
                domain  tcitem          iERPItem,
        ref     domain  tcmcs.str50     oBODItem,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

Usage:

```
        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.
                                        If more than one  message is
                                        given, these are present 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.ConvertFromISODateToERP

DLL: tcextbodapi

This function is available from KB2308619.

Syntax:

```
long BOD.ConvertFromISODateToERP(
                domain  tcmcs.str30      iISODate,
        ref     domain  tcdate           oERPDate,
        ref     domain  tcmcs.s999m      oExceptionMessage mb,
        ref             long             oExceptionID )
```

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"
        Output  : oERPDate              - 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.
                  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.
```

BOD.ConvertToERPItem

DLL: tcextbodapi

This function is available from KB2044306.

Syntax:

```
long BOD.ConvertToERPItem(
                domain  tcmcs.str50      iBODItem,
        ref     domain  tcitem           oERPItem,
```

```
        ref     domain  tcmcs.s999m      oExceptionMessage mb,
        ref             long             oExceptionID )
```

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.s treated as standard item and leading spaces
                  are added.

                  EXAMPLES:
                        CBO item string         ERP item
                        ---------------------------------------
                        "PR0000001BIKE"         "PR0000001BIKE"
                                                Project = PR0000001

                        "PR1      BIKE"         "PR1      BIKE"
                                                Project = PR1

                        "         BIKE"         "         BIKE"
                        "BIKE"                  "         BIKE"
                                                Project = ""
        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
```

BOD.ExecuteMethod

DLL: tcextbodapi

Syntax:

```
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 )
```

Usage:

```
        Expl:   This function executes a non-batch method for a specified
                BOD.
        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
                oExceptionID            - An ID that refers to all error
                                          information. Use the functions in
                                          Exception to get all relevant
                                          information.
        Return values:
                0                       - method is executed successfully
                DALHOOKERROR            - on errors
```

BOD.ExecuteOnAcknowledgeForChameleon

DLL:	tcextbodapi

Syntax:

```
long BOD.ExecuteOnAcknowledgeForChameleon(
                domain  tcbod.name      iProtectedNoun,
                        long            iXMLRequest,
        ref             long            oXMLResponse,
        ref             long            oXMLResult,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

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 / DALHOOKERROR

BOD.ExecuteOnLoadForChameleon

DLL:	tcextbodapi
Syntax:	long BOD.ExecuteOnLoadForChameleon(domain tcbod.name iProtectedNoun, long iXMLRequest, ref long oXMLResponse, ref long oXMLResult, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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 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 / DALHOOKERROR

BOD.ExecuteOnProcessForChameleon

DLL:	tcextbodapi
Syntax:	long BOD.ExecuteOnProcessForChameleon(domain tcbod.name iProtectedNoun, long iXMLRequest, ref long oXMLResponse, ref long oXMLResult, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)

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 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 / DALHOOKERROR

BOD.ExecuteOnSyncForChameleon

DLL:	tcextbodapi
Syntax:	long BOD.ExecuteOnSyncForChameleon(domain tcbod.name iProtectedNoun, long iXMLRequest, ref long oXMLResponse, ref long oXMLResult, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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 / DALHOOKERROR
```

BOD.ExecuteOnUpdateForChameleon

DLL: tcextbodapi

Syntax:
```
long BOD.ExecuteOnUpdateForChameleon(
                domain  tcbod.name      iProtectedNoun,
                        long            iXMLRequest,
        ref             long            oXMLResponse,
        ref             long            oXMLResult,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

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
                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 / DALHOOKERROR
```

BOD.ExecutePublishForChameleon

DLL: tcextbodapi

Syntax:
```
long BOD.ExecutePublishForChameleon(
                domain  tcbod.name      iProtectedNoun,
                        long            iXMLRequest,
        ref             long            oXMLResponse,
        ref             long            oXMLResult,
```

```
        ref     domain  tcmcs.s999m      oExceptionMessage mb,
        ref             long             oExceptionID )
```

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 / DALHOOKERROR
```

BOD.ExecuteShowForChameleon

DLL: tcextbodapi

Syntax:

```
long BOD.ExecuteShowForChameleon(
                domain  tcbod.name       iProtectedNoun,
                        long             iXMLRequest,
        ref             long             oXMLResponse,
        ref             long             oXMLResult,
        ref     domain  tcmcs.s999m      oExceptionMessage mb,
        ref             long             oExceptionID )
```

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
                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 / DALHOOKERROR
```

BOD.GetIdAccountingEntity

DLL: tcextbodapi

Syntax:

```
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 )
```

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.
                DALHOOKERROR            - Otherwise.
```

BOD.GetIdLocation

DLL: tcextbodapi

Syntax:
```
long BOD.GetIdLocation(
                domain  tncmp           iCompany,
                domain  tcbod.name      iNoun,
                        long            iEntityType,
                domain  tcmcs.str30     iEntityCode,
                domain  tcmcs.tabl      iRootTable,
        ref     domain  tcbod.lctn      oLocation,
        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.
                DALHOOKERROR            - Otherwise.
```

BOD.GetIdLogicalID

DLL: tcextbodapi

Syntax:
```
long BOD.GetIdLogicalID(
                domain  tncmp           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.
                DALHOOKERROR            - Otherwise.
```

BOD.GetPublishingAllowed

DLL: tcextbodapi

This function is available from KB2267552.

Syntax:

```
long BOD.GetPublishingAllowed(
        ref             boolean          oAllowed,
        ref     domain  tcmcs.s999m      oExceptionMessage mb,
        ref             long             oExceptionID )
```

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.HandleStagingAfterProcessingIncomingRequest

DLL:	tcextbodapi This function is available from KB2040021.
Syntax:	long BOD.HandleStagingAfterProcessingIncomingRequest(domain tcbod.name iNoun, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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 / DALHOOKERROR

BOD.HandleStagingBeforeProcessingIncomingRequest

DLL:	tcextbodapi This function is available from KB2040021.
Syntax:	long BOD.HandleStagingBeforeProcessingIncomingRequest(domain tcbod.name iNoun, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)

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 / DALHOOKERROR

BOD.InterruptPublishing

DLL:	tcextbodapi This function is available from KB2267552.
Syntax:	long BOD.InterruptPublishing(boolean iAllBods, domain tcbod.name iNoun, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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: 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.
        Return:
                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:	`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,`
	`... )`

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 - Noun: 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.
                DALHOOKERROR            - Otherwise.
```

BOD.PublishLNMessage

DLL: tcextbodapi

This function is available from KB2040021.

Syntax:

```
long BOD.PublishLNMessage(
                domain  tcmcs.str30m     iMessageType mb,
        ref     domain  tcmcs.s999m      oExceptionMessage mb,
        ref             long             oExceptionID,
                                         ... )
```

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.:
                        "Description",          tfgld011.desc,
                The number of input arguments can vary.

                The DocumentID is mandatory as one of the xml_element names:
                        "DocumentID",           table.key1,
                        "DocumentID",           table.key2,

                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:
                        "MessageType"           "TestType",
                        "Element.1"             field.1,
                        "Element.2"             field.2,
                result node:
                        <MessageTypeDetails>
                                <TestTypeGroup>
                                        <Element.1>field.1</Element.1>
                                        <Element.2>field.2</Element.2>
                                </TestTypeGroup>
                        </MessageTypeDetails>

                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.:
                        "PurchaseInvoiceAmount",                "10.000",
                        "PurchaseInvoiceAmount@currencyID",     "EUR",

                        <PurchaseInvoiceAmount currencyID="EUR">10.000
                                                </PurchaseInvoiceAmount>


                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"
                    -  "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.
                DALHOOKERROR            - Error.
```

BOD.ResumePublishing

DLL: tcextbodapi

This function is available from KB2267552.

Syntax:

```
long BOD.ResumePublishing(
                        boolean         iAllBods,
                domain  tcbod.name      iNoun,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

Usage:

```
        Expl:   This function resumes the publishing of all BODs or one
                specified BOD in the current process.
        Pre:    Function BOD.InterruptPublishing() should have been called to
                interrupt the publishing.
        Post:   N/A
        Output:
                iAllBods                - 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.
                iNoun                   - The (protected) Noun for which the
                                          publishing must be resumed.
                                          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.
        Return:
                0                       - The action to resume the publishing was set
successfully.
                <> 0                    - The action to resume the publishing can not be
set.
```

Public Interfaces for BDE

The following functions are available:

BDE.ExecuteMethod

BDE.ExecuteMethod

DLL:	tcextbdeapi This function is available from KB2061133.

Syntax:	`long BDE.ExecuteMethod(`
	`  domain  tcbod.name  iBusinessObject,`
	`  domain  tcmcs.str30  iMethod,`
	`  long  iXMLRequest,`
	`  ref  long  oXMLResponse,`
	`  ref  long  oXMLResult,`
	`  ref  domain  tcmcs.s999m  oExceptionMessage mb,`
	`  ref  long  oExceptionID )`

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`
	`DALHOOKERROR  - on errors`

Chapter 21 Public Interfaces for Service

Public Interfaces for Service

The following functions are available:

Service.DetermineStartOrFinishTime

Service.GetResourceCalendarCode

Service.DetermineStartOrFinishTime

DLL:	tsextsocapi This function is available from KB2117506.
Syntax:	long Service.DetermineStartOrFinishTime(domain tcncmp iLogisticCompany, domain tsbsc.clst iInstallationGroup, domain tcitem iItem, domain tcibd.sern iSerialNumber, domain tcemno iAssignedServiceEngineer, domain tcemno iPreferredServiceEngineer, domain tccwoc iServiceDepartment, domain tcccp.ccap iDuration, domain tcccp.ract iAvailabilityType, domain tcplcd iPlanMethod, domain tsmdm.pldt iTime, ref domain tsmdm.pldt oTime, ref domain tcmcs.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/
                                                  finish date): Not Mandatory
        Output  : oTime                         - Result Time (planned start/
                                                  finish date)
                  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                               - Data read
                DALHOOKERROR                    - Switch to company failed.
```


Service.GetResourceCalendarCode

DLL:	tsextsocapi This function is available from KB2117506.
Syntax:	long Service.GetResourceCalendarCode(domain tcncmp iLogisticCompany, domain tsclm.crtc iResponseType, domain tsbsc.clst iInstallationGroup, domain tcitem iItem, 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
                                                  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
                DALHOOKERROR                    - Switch to company failed.
```

Public Interfaces for ServiceContract

The following functions are available:

ServiceContract.Activate

ServiceContract.GenerateConfigurationLines

ServiceContract.UnpackObjectReference

ServiceContract.Activate

DLL:	tsextctmapi This function is available from KB2285960.

Syntax:	`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 )
```

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.GenerateConfigurationLines

DLL:	tsextctmapi This function is available from KB2289318.

Syntax:

```
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.cgru       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 )
```

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.UnpackObjectReference

DLL: tsextctmapi

This function is available from KB3644790.

Syntax:

```
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  tcemm.seqn      oCostLine,
        ref     domain  tcccp.yrno      oYear,
        ref     domain  tcccp.peri      oPeriod,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

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 ServiceOrder

The following functions are available:

ServiceOrder.Close

ServiceOrder.Close

DLL: tsextsocapi

This function is available from KB2297823.

Syntax:

```
long ServiceOrder.Close(
                domain  tcorno          iServiceOrder,
                domain  tcyesno         iDeleteServiceOrder,
                domain  tcyesno         iCopyServiceOrderToHistory,
                domain  tcyesno         iClosePhysicalBreakdownChanges,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

Usage:

```
        Expl:   Use this function to Close a Service Order.

                The same functionality is offered as available in session
                Close Service Orders (tssoc2201m000).
        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

Public Interfaces for WorkOrder

The following functions are available:

WorkOrder.Complete

WorkOrder.Release

WorkOrder.Complete

DLL:	tsextwcsapi This function is available from KB2162018.
Syntax:	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 )
```

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. Must be >= 0
                        (mandatory)
                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
                        statusses 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.

                        Note: This input is not used. It is there for future use.
                iCreateFollowupWorkOrderForRemainingQuantity
                        Create Follow-up Work Order for remaining quantity.
                        (mandatory Yes/No)
```

	Only applicable for non-serialized controlled items. Note: This input is not used. It is there for future use. 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 - work order set to completed succesfully <> 0 - Error during completing work order occurred

WorkOrder.Release

DLL:	tsextwcsapi This function is available from KB2162018.
Syntax:	long WorkOrder.Release(domain tcorno iWorkOrder, domain tsmdm.acln iWorkOrderActivityLine, domain tccotp iPurchaseOrderType, domain tcseri iPurchaseOrderSeries, domain tcyesno iATPCheck, domain tcyesno iCheckPlannedInventory, 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)
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. 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 successfull. 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

                        Note: This input is not used. It is there for future use.
                iUpdatePlannedQuantityToIssuedQuantity (Yes/No)
                        Update Planned Quantity to Issued Quantity.
                        (mandatory Yes/No)

                        Note: This input is not used. It is there for future use.
                iShiftWorkOrderToActualReceiptTime
                        Shift Work Order to Actual Receipt Time.
                        (mandatory Yes/No)

                        Note: This input is not used. It is there for future use.
                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 - Released succesfully <> 0 - Error during release occurred

Public Interfaces for PhysicalBreakdown

The following functions are available:

PhysicalBreakdown.ItemExistsInBreakdownOfParent

PhysicalBreakdown.ItemExistsInBreakdownOfParent

DLL:	tsextcfgapi This function is available from KB2331323.
Syntax:	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)
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

Chapter 22 Public Interfaces for Product Testing and Control

Public Interfaces for OrderInspection

The following functions are available:

OrderInspection.Close

OrderInspection.Complete

OrderInspection.Process

OrderInspection.Close

DLL:	qmextptcapi This function is available from KB2181089.
Syntax:	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)
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: 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 succesfully <> 0 - Error

OrderInspection.Complete

DLL:	qmextptcapi This function is available from KB2181089.
Syntax:	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)
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
                                          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 completed succesfully
                <> 0                    - Error
```

OrderInspection.Process

DLL: qmextptcapi

This function is available from KB2181089.

Syntax:

```
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 )
```

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,
                                          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 processed succesfully <> 0 - Error

Chapter 23 Public Interfaces for Project

Public Interfaces for Project

The following functions are available:

Project.StartCopy

Project.StartCopy

DLL:	tpextpdmapi This function is available from KB2231502.

Syntax:	`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,`
	`                                        ... )`

Usage:	`        Expl    This function starts the Copy Project session (tppdm7840m000).`
	`                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
                                date.time.utc)                  - input

        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 ProjectEstimate

The following functions are available:

ProjectEstimate.GenerateStructuralElements

ProjectEstimate.GenerateStructuralElements

DLL:	tpextestapi This function is available from KB2220662.
Syntax:	long ProjectEstimate.GenerateStructuralElements(domain tccprj iProject, 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)
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
```

Chapter 24 Public Interfaces for General Ledger

Public Interfaces for GeneralLedger

The following functions are available:

GeneralLedger.CheckDocumentIsBalanced

GeneralLedger.CheckDocumentIsBalanced

DLL:	tfextgldapi This function is available from KB2285516.
Syntax:	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)
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

Chapter 25 Public Interfaces for Fixed Assets

Public Interfaces for AssetBook

The following functions are available:

AssetBook.CalculateRemainingDepreciationLife

AssetBook.CalculateRemainingDepreciationLife

DLL:	tfextfamapi This function is available from KB2287701.
Syntax:	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)
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:  iAssetCompany           - Asset Company: Mandatory
                iAssetNumber            - Asset Number: Mandatory
                iAssetExtension         - Asset Extension: Mandatory
                iAssetBook              - Asset Book: Mandatory
        Output: 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.
        Return: 0                       - Data read
                <> 0                    - An error occurred
```

Chapter 26 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:	tcextocmapi This function is available from KB2220087.

Syntax:	`long Workflow.DisableSubmit(`
	`  domain  tcmcs.str6  iObjectType,`
	`  ref  domain  tcmcs.s999m  oExceptionMessage mb,`
	`  ref  long  oExceptionID )`

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:
                        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
```


Workflow.EnableSubmit

DLL:	tcextocmapi This function is available from KB2220087.
Syntax:	long Workflow.EnableSubmit(ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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: oExceptionMessage - The last message if the return value is not equal to 0. If more than one message is given, these are present 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

Workflow.Recall

DLL:	tcextocmapi This function is available from KB2289885.
Syntax:	long Workflow.Recall(domain tcmcs.tabl iRootTable, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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              - 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
```

Workflow.RecallByTypedObjectID

DLL: tcextocmapi

This function is available from KB2310367.

Syntax:

```
long Workflow.RecallByTypedObjectID(
                domain  ttocm.toid      iTypedObjectID,
        ref     domain  tcmcs.s999m     oExceptionMessage mb,
        ref             long            oExceptionID )
```

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          - Workflow Typed Object ID belonging to
                                          the record that must be recalled.
                                          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
```

Workflow.Submit

DLL: tcextocmapi

This function is available from KB2289885.

Syntax:

```
long Workflow.Submit(
                domain  tcmcs.tabl       iRootTable,
        ref     domain  tcmcs.s999m      oExceptionMessage mb,
        ref             long             oExceptionID )
```

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 27 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 KB2299983.
Syntax:	long ProductionOrderHours.CloseActiveLine(domain tcemno iEmployee, domain tcccp.yrno 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: iEmployee - Employee (Mandatory) iYear - Year (Mandatory) iPeriod - Period (Mandatory) iSequence - Sequence (Mandatory) iEndDateTime - End Date and Time (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                       - Active Production Order Hours Line
                                          has been closed.
                <> 0                    - Error while closing active Production
                                          Order Hours Line.
```

ProductionOrderHours.InsertActiveLine

DLL: bpexttmmapi

This function is available from KB2299983.

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 - Active Production Order Hours Line Inserted. <> 0 - Active Production Order Hours Line not inserted.

ProductionOrderHours.StartOverview

DLL:	bpexttmmapi This function is available from KB2299983.
Syntax:	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.serd oSequence, ref domain tcmcs.s999m oExceptionMessage mb, ref long oExceptionID)
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:	bpexttmmapi This function is available from KB2299983.

Syntax:

```
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 )
```

Usage:

```
        Expl:   This function starts session General Hours -
                bptmm1111m200 (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. Return: 0 - Session started <> 0 - Otherwise.

Chapter 28 Public Interfaces for ZWF AG Shipment and Customs

Public Interfaces for SanctionList

The following functions are available:

SanctionList.Check

SanctionList.Check

DLL:	tzextzfcapi This function is available from KB3637797.
Syntax:	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)
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 Adress ID
        iShowPopupMessage       - Show poupmessage with result y/n
        iIsAdressCheck          - Is check for LN adress 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 occured 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 Adress ID
        iBaseType               - Base document type
        iAdditionalIntegration  - Additional Customised Integration Type
        iShowPoupMessage        - Show poupmessage 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 occured in check
```

Chapter 29 Process Extensions

Process Extensions for AssemblyOrder

The following process extension(s) is/are available:

AssemblyOrder.SkipDelete

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

Process Extensions for BillOfLading

The following process extension(s) is/are available:

BillOfLading.SkipShipmentLine

BillOfLading.SkipShipmentLine

Skips Shipment Lines when Printing Bill of Lading.

This process extension is available from KB2120253.

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

        table twhinh430

Hook: ext.skip

        function extern boolean ext.skip()
        {
                if <condition on whinh430 = true> then
                        return(true)
                endif

                return (false)
        }
```

Process Extensions for BOD

The following process extension(s) is/are available:

BOD.PublishCustomBOD

BOD.PublishCustomBOD

Publish Custom BOD.

This process extension is available from KB2164036.

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: - tcext.bod0001.get.custom.bodnames() 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). - tcext.bod0001.get.initial.load.query() 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: - tcext.bod0001.get.custom.bod.publish.function() This method must be used if the SQL query in the tcext.bod0001.get.initial.load.query() 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 BOD.Publish() 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. - tcext.bod0001.get.custom.bod.properties() This method must be used if a BOD must be published for each record that is returned by the SQL query in the tcext.bod0001.get.initial.load.query() 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 tcext.bod0001.get.custom.bod.properties() method should be implemented. Implementation of the tcext.bod0001.get.custom.bod.publish.function() 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 tcext.bod0001.get.custom.bodnames() - Create custom fields for the selection ranges - Implement function tcext.bod0001.get.initial.load.query() - Implement either function tcext.bod0001.get.custom.bod.properties() or implement function tcext.bod0001.get.custom.bod.publish.function() - If function tcext.bod0001.get.custom.bod.publish.function() 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):

tcext.bod0001.get.custom.bod.properties

Syntax:	long tcext.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)
Usage:	Use this method to return meta data fields that are needed in the BOD. Input: - i.bodname - custom BOD name Output: - o.documentID - documentID - o.action.code - BOD action code, possible values "Add", "Change", "Delete", "Replace" or "Canceled" - o.bod.entity.type - 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.
      - o.bod.entity.code - code related to the entity type.
                              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)
```

tcext.bod0001.get.custom.bod.publish.function

Syntax:

```
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 )
```

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:
               functionname = "tcbod.dll0005.publish.custom.bod"
               instruction = 1

      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:

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)


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

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(
                                "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)
       }
```

tcext.bod0001.get.custom.bodnames

Syntax:

```
long tcext.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 )
```

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:

```
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
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)
```

tcext.bod0001.get.initial.load.query

Syntax:

```
long tcext.bod0001.get.initial.load.query(
                 domain  tcbod.name       i.bodname,
 ref             string           o.query() )
```

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 tcext.bod0001.get.custom.bod.publish.function() 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:

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 " &
                      "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)
```

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:

```
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
```

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 (cproa1425m000) 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 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.

This process extension is available from KB2086985.

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:	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)
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 i.handling.unit.reused 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: 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

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 KB2327467.

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.
This extension can only be implemented when the tools version
is 10.8.4 or higher.

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

      table twhinh215

Hook: ext.skip.with.reason

      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)
      }
```

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 KB3549440.

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:	long lpext.ita0001.check.intrastat.transaction.blocked.for.reporting(ref boolean o.blocked.for.reporting)
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 o.blocked.for.reporting 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 lpita360 table are already available. Note that dal messages set in this function will be ignored by the standard. ---------- Start of Example of Implementation

```
Pseudocode:

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

                |* 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                       - Success
        <> 0                    - Returning a value <> 0 will abort the
                                  creation of Intrastat Transactions.
```

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.

This process extension is available from KB2073261.

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:	long whext.dll0002.check.handling.unit(ref boolean o.skip.handling.unit, ref domain tcmcs.str132m o.error.message.array() fixed mb)
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 additional checks or skips of found Stock Points.

This process extension is available from KB2073261.

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.
```

To implement this process extension, you need to implement the following method(s):

whext.dll0001.check.stock.point

Syntax:

```
long whext.dll0001.check.stock.point(
        ref             boolean          o.skip.stock.point,
        ref     domain  tcmcs.str132m    o.error.message.array() fixed mb )
```

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

Process Extensions for Invoice

The following process extension(s) is/are available:

Invoice.CustomComposingCriteriaMet

Invoice.CustomXMLInvoicing

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 Extenstion 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 extenstion 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:

```
long ciext.sli0003.invoice.custom.composing.criteria.met(
        ref             boolean          o.custom.composing.criteria.met )
```

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
                extenstion 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 Billale line
                                and is available in the Invoice line Additional
                                Fields (cisli311).
                        Pseudocode  -

                        Hook: Declarations
                                table   tcisli305       |* Invoice Header
                                table   tcisli810       |* Billable Lines

                        Hook: ciext.sli0003.invoice.custom.composing.criteria.met

                                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

                                        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.sfcp,
                                                                :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                       - Success
                        DALHOOKERROR            - When an error occurs during checking;
```

	the invoice will not be composed together.

Invoice.CustomXMLInvoicing

Define custom elements to be used in XML invoicing.

This process extension is available from KB2241507.

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:	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())
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: ciext.sli0001.get.value.for.custom.xml.data.element 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

            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 wil 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
           (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(  "cislis0266")
                |* 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:

```
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 )
```

Usage:

```
        Expl:   Use this method to get a value and type for the given custom
                XML data element field.

                ---------------------------------------------------------------
                Start of Example of Implementation

        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$(i.custom.element.field.name)
        case "ext_fld_h001":
                o.custom.element.value = "This is the string value of field ext_fld_h001"
                o.custom.element.type = NO_FIELD_TYPE
                break
        case "ext_fld_h002":
                o.custom.element.value = "8.8"
                o.custom.element.type = AMOUNT_FIELD
                break
        case "ext_fld_h003":
                o.custom.element.value = "1"                    |* = true
                o.custom.element.type = BOOLEAN_FIELD
                break
        case "ext_fld_h004":
                o.custom.element.value = enum.descr$(
                                         "tcinvt",
                                         tcinvt.travel,
                                         "2")
                o.custom.element.type = NO_FIELD_TYPE
                break
        case "ext_fld_h005":
                o.custom.element.value = str$(utc.num())
                o.custom.element.type = UTC_DATE_FIELD
                break
        case "ext_fld_h007":
                o.custom.element.value = "12.43"
                o.custom.element.type = QUANTITY_FIELD
                break
        case "ext_fld_h008":
                o.custom.element.value = "1.1"
                o.custom.element.type = PRICE_PERCENTAGE_FIELD
                break
        case "ext_fld_l001":
                o.custom.element.value = "This is the string value of ext_fld_l001"
                o.custom.element.type = NO_FIELD_TYPE
                break
        case "ext_fld_s001":
                object.to.keyfields(i.section.key.object)
                o.custom.element.value =        concat$("/",
                                                "Custom TaxSummary Field: ",
                                                cisli315.sfcp,
                                                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.

                i.custom.element.field.name - Custom XML data element for which
                                  a value must be retrieved.

        Output: o.custom.element.value  - The value of the custom element field
                o.custom.element.type   - The element type of the custom element field,
                                          the "o.custom.element.value" will be formatted
                                          or converted according the element type value.
                                        - The next element types are supported:
                                          #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
layout.
                                          #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"
definition.
                                                    (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"
definition.
                                                 (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. Return: 0 - Success <> 0 - When an error occurs in getting a value for custom xml data element.

Process Extensions for InvoiceFromBP

The following process extension(s) is/are available:

InvoiceFromBP.SkipPrintAgingAnalysis

InvoiceFromBP.SkipPrintOpenEntries

InvoiceFromBP.SkipPrintAgingAnalysis

Skips Invoice-from Business Partner when Printing Aging Analysis.

This process extension is available from KB2191481.

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 tccom122:       tccom122.ifbp (Invoice-from Business
Partner)
                                                tccom122.cofc (Department)
        Pseudocode:
        Below you can find an example:

        Hook: Declarations

                table ttccom122

        Hook: ext.skip

                function extern boolean ext.skip()
                {
                        |* read data using tccom122.ifbp, tccom122.cofc

                        if <condition on the data read = true> then
                                return(true)
                        endif
```

```
                return (false)
        }
```

InvoiceFromBP.SkipPrintOpenEntries

Skips Invoice-from Business Partner when Printing Open Entries.

This process extension is available from KB2191481.

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 tccom122 are available.

Pseudocode:
Below you can find an example:

Hook: Declarations

        table ttccom122

Hook: ext.skip

        function extern boolean ext.skip()
        {
                if <condition on the data read = true> then
                        return(true)
                endif

                return (false)
        }
```

Process Extensions for InvoiceToBP

The following process extension(s) is/are available:

InvoiceToBP.SkipPrintAgingAnalysis

InvoiceToBP.SkipPrintOpenEntries

InvoiceToBP.SkipPrintAgingAnalysis

Skips Invoice-to Business Partner when Printing Aging Analysis.

This process extension is available from KB2191481.

To implement this process extension, you can use the information below:

Usage:

```
        Process Extension InvoiceToBP.SkipPrintAgingAnalysis can be used
        to skip Invoice-to Business Partner when Printing Aging Analysis from
        Print Invoice-to Business Partner Aging Summary.
        Sessions where this Process Extension can be implemented:
        - Print Invoice-to Business Partner Aging Summary. (tfacr2411m000)
        - Print Invoice-to Business Partner Aging Analysis. (tfacr2420m000)

        Fields that are available to be used in this Process Extension:
        - Primary key fields of tccom112:       tccom112.itbp (Invoice-to Business
Partner)
                                                tccom112.cofc (Department)
        Pseudocode:
        Below you can find an example:

        Hook: Declarations

                table ttccom112

        Hook: ext.skip

                function extern boolean ext.skip()
                {
                        |* read data using tccom112.itbp, tccom112.cofc

                        if <condition on the data read = true> then
                                return(true)
                        endif

                        return (false)
                }
```

InvoiceToBP.SkipPrintOpenEntries

Skips Invoice-to Business Partner when Printing Open Entries.

This process extension is available from KB2191481.

To implement this process extension, you can use the information below:

Usage:

```
        Process Extension InvoiceToBP.SkipPrintOpenEntries can be used
        to skip Invoice-to Business Partner when Printing Open Entries from Print
        Invoice-to Business Partner Open Entries.
        Sessions where this Process Extension can be implemented:
        - 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

        table ttccom112

Hook: ext.skip

        function extern boolean ext.skip()
        {
                if <condition on the data read = true> then
                        return(true)
                endif

                return (false)
        }
```

Process Extensions for ItemMasterPlan

The following process extension(s) is/are available:

ItemMasterPlan.SkipCopyToScenario

ItemMasterPlan.SkipCopyToScenario

Skips copying Item Master Plan to Scenario.

This process extension is available from KB2239394.

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:
- proc_ext_skip_item_masterplan_cpy_scen_from [ type: domain cpcom.plnc ]
- proc_ext_skip_item_masterplan_cpy_scen_to   [ type: domain cpcom.plnc ]
```

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.

This process extension is available from KB2177003.

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.

This process extension is available from KB2092284.

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.

This process extension is available from KB2177003.

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.

This process extension is available from KB2177003.

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 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.

This process extension is available from KB2109354.

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:	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())
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:
        tfext.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
|*****************************************************************
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 "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.
```

tfext.cmg0001.define.custom.receipt.elements

Syntax:

```
long tfext.cmg0001.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() )
```

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:
                        tfext.cmg0001.get.value.for.custom.receipt.element

                An example implementation is given in the method
                tfext.cmg0001.define.custom.payment.elements, 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 "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.
```

tfext.cmg0001.get.value.for.custom.payment.element

Syntax:

```
domain tcmcs.s999m tfext.cmg0001.get.value.for.custom.payment.element(
                domain  tcncmp          i.payment.company,
                domain  tcmcs.st12      i.custom.element.code,
                domain  tccwoc          i.accounting.office )
```

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:
                        tfext.cmg0001.define.custom.payment.elements

                Following tables are current when this extension is triggered
                (during creation of the XML Bank File):

                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

                tfcmg103        Composed Payments
                tfcmg002        Reasons for Payment
                tfcmg025        Payment/Receipt XML Layout Details

                The below table is current only for Remittance Related elements:
                tfcmg101        Payment/Trade Notes Payable Advice

                ------------------------------------------------------------------
                Start of Example of Implementation

                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:    N.A.
        Post:   N.A.
        Input:  i.payment.company       - Payment Company
                i.custome.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:

```
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 )
```

Usage:

```
        Expl:
                Use this method to get a value for the defined custom receipt
                element code during the creation of the Payment XML File.

                The custom elements must be defined with the following method:
                        tfext.cmg0001.define.custom.receipt.elements

                Following tables are current when this extension is triggered
                (during creation of the XML Bank File):

                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

                An example implementation is given in the method
                tfext.cmg0001.get.value.for.custom.payment.element, which is
                likely same.

        Pre:    N.A.
        Post:   N.A.
        Input:  i.receipt.company       - Receipt Company
                i.custome.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 receipt element.
```

Process Extensions for Pricing

The following process extension(s) is/are available:

Pricing.RetrievePriceAndDiscounts

Pricing.RetrievePriceAndDiscounts

Determine customer specific Price and Discounts.

This process extension is available from KB2215124.

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:

```
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,
                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  tcmcs.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()
fixed,
        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(),
        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 )
```

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:
                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).
                                                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
                                                with 11 elements.
```

	o.disc.matrix.type(11) Discount Matrix Type array o.discount.origin(11) Discount Origin array o.disc.matrix.definition(11) Discount Matrix Definition array o.disc.matrix.sequence(11) Discount Matrix Sequence array o.discount.percentage(11) Discount Percentage array o.discount.amount(11) Discount Amount array o.discount.code(11) Discount Code array o.discount.method(11) Discount Method array o.discount.schedule(11) Discount Schedule array o.determining Determining (Yes/No) o.eligible Eligible (Yes/No) o.fallback.to.standard.logic.in.case.of.errors 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/"" Return: 0 Success <> 0 An error occurred

Process Extensions for Procurement

The following process extension(s) is/are available:

Procurement.DefaultPurchaseOffice

Procurement.DefaultPurchaseOffice

Determine customer specific default Purchase Office for Procurement.

This process extension is available from KB2179522.

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:

```
long tdext.pur0001.get.customer.determined.default.purchase.office(
                domain  tcmcs.str8       i.defaulting.origin,
                domain  tccwoc           i.purchase.office.from.standard.logic,
                domain  tcemm.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 )
```

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
                                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.

Process Extensions for ProductionOrder

The following process extension(s) is/are available:

ProductionOrder.PrintCustomReports

ProductionOrder.SkipGenerateSubcDocuments

ProductionOrder.SkipPrintMaterialShortage

ProductionOrder.SkipReportComplete

ProductionOrder.SkipWarehouseForMaterialShortage

ProductionOrder.PrintCustomReports

Production Order Custom Reports.

This process extension is available from KB2225976.

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:

```
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() )
```

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 txext001) with an index of the
                production order and the report group. Add a field
                (ex. txext001.stat) in the created tx-table for the print status
                with domain tisfc.pdst.
                Write the function as follows:

                function extern 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)
                {
                                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  tcmcs.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:    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:

```
long tiext.sfc0001.print.custom.report(
                domain  tcmcs.long      i.report.group,
                domain  tcmcs.st14      i.spooler.device,
                        long            i.lfn.custom.spool,
        ref             long            o.lfn.custom.brpfile,
        ref             string          o.message() )
```

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".

                        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             long            o.lfn.custom.brpfile,
                ref             string          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)
        }

        function open.custom.brpfile(
                                boolean         i.do.brp.open,
                        domain  tcmcs.st14      i.spooler.device,
                                long            i.lfn.custom.spool,
                ref             long            o.lfn.custom.brpfile)
        {
                if i.lfn.custom.spool = 0 then
                        return
                endif

                spool.id = i.lfn.custom.spool

                if i.do.brp.open then
                        o.lfn.custom.brpfile = brp.open(
                                                report$,
                                                i.spooler.device,
                                                0)
                        if o.lfn.custom.brpfile <= 0 then
                                job.process.error = true
                                return
                        endif
                        open.custom.brp = true
                endif
        }

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.brpfile    - 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:

```
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() )
```

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".

                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:    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:

```
long tiext.sfc0001.update.print.status(
        ref             boolean          i.custom.report.printed(),
        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 txext001) with an index of the
                production order and the reportgroup. Add a field
                (ex. txext001.stat) in the created tx-table for the print status
                with domain tisfc.pdst.
                Write the function as follows:

                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  txext001.stat:status
                        from    txext001 for update
                        where   txext001._index1 = {    :i.production.order,
                                                        :i.report.group}
                        selectdo
                                ret = dal.change.object(txext001)
                                if ret <> 0 then
                                        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
```

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.

This process extension is available from KB2175031.

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)

        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)
                }
```

Process Extensions for ProductVariant

The following process extension(s) is/are available:

ProductVariant.AddCustomCPQIntegrationParameters

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:	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).

To implement this process extension, you need to implement the following method(s):

tiext.pcf0003.get.cpq.integration.parameter.value

Syntax:	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)
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:
                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

        Implementation example:
                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
                        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
```

tiext.pcf0003.register.cpq.integration.parameters

Syntax:
```
long tiext.pcf0003.register.cpq.integration.parameters(
        ref             long            o.number.of.parameters,
        ref     domain  tcmcs.str50     o.parameter.names() fixed )
```

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:
                |* 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"
                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
```

ProductVariant.AdjustPrice

Adjust the price of a product variant at the end of the price calculation.

This process extension is available from KB2085869.

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 calculation of the sales price or purchase price of a product variant, just before rounding the final result.
	
	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 and all tipfc400 fields are current (inclusing the Customer Defined Fields) and can be used in this method.
	
	EXAMPLE:
	Assume that ticpf400.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
	
	if tipcf400.kopl = tckopl.sales then
	io.price = io.price + tipfc400.acst
	endif
	
	Pre: when ticpf400 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:

```
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  tibmrv          i.revision,
                domain  tibmst          i.status,
        ref             boolean         o.skip.delete.or.expire.jsbom )
```

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:
                    function extern long tiext.pcs0002...
                    ...
                    {
                        if      str.startswith(i.parent.item, "SLO")
                        then
                                o.skip.delete.or.expire.jsbom = true
                        endif
                        return(0)
                    }
        Pre:    NA
        Post:   NA
        Input:  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

        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.
```


Process Extensions for ProjectPCS

The following process extension(s) is/are available:

ProjectPCS.SkipClose

ProjectPCS.SkipClose

Project PCS Skip Close of Project.

This process extension is available from KB2220251.

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:	long tiext.pcs0001.project.pcs.skip.close(ref boolean o.skip.close, ref string o.message())
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:	boolean tiext.pcs0001.project.pcs.skip.close.implemented( )
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

Process Extensions for PurchaseOrder

The following process extension(s) is/are available:

PurchaseOrder.SkipApprove

PurchaseOrder.SkipPrintPurchaseInvoice

PurchaseOrder.SkipPrintPurchaseOrder

PurchaseOrder.SkipReleaseToWarehousing

PurchaseOrder.SkipApprove

Skips Purchase Order when Approving.

This process extension is available from KB2169810.

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

        table ttdpur400


Hook: ext.skip

        function extern boolean ext.skip()
        {
                if <condition on tdpur400 = true> then
                        return(true)
                endif

                return (false)
        }
```

PurchaseOrder.SkipPrintPurchaseInvoice

Skips Purchase Order when Printing Purchase Invoices.

This process extension is available from KB2142810.

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

        table ttdpur400

Hook: ext.skip

        function extern boolean ext.skip()
        {
                if <condition on tdpur400 = true> then
```

```
                        return(true)
                endif

                return (false)
        }
```

PurchaseOrder.SkipPrintPurchaseOrder

Skips Purchase Order when Printing Purchase Order.

This process extension is available from KB2311745.

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

        table ttdpur400

Hook: ext.skip

        function extern boolean ext.skip()
        {
                if <condition on tdpur400 = true> then
                        return(true)
                endif

                return (false)
        }
```

PurchaseOrder.SkipReleaseToWarehousing

Skips Purchase Order Lines when Releasing to Warehousing.

This process extension is available from KB2281563.

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 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)
	}

Process Extensions for PurchaseOrderLine

The following process extension(s) is/are available:

PurchaseOrderLine.SkipPrintPurchaseInvoice

PurchaseOrderLine.SkipPrintPurchaseOrder

PurchaseOrderLine.SkipPrintPurchaseOrderReminder

PurchaseOrderLine.SkipPrintPurchaseInvoice

Skips Purchase Order Line when Printing Purchase Invoices.

This process extension is available from KB2142810.

To implement this process extension, you can use the information below:

Usage:

```
Process Extension PurchaseOrderLine.SkipPrintPurchaseInvoice 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

        table ttdpur401

Hook: ext.skip

        function extern boolean ext.skip()
        {
                if <condition on tdpur401 = true> then
                        return(true)
                endif

                return (false)
        }
```

PurchaseOrderLine.SkipPrintPurchaseOrder

Skips Purchase Order Line when Printing Purchase Order.

This process extension is available from KB2142810.

To implement this process extension, you can use the information below:

Usage:

```
Process Extension PurchaseOrderLine.SkipPrintPurchaseOrder 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

        table ttdpur401

Hook: ext.skip

        function extern boolean ext.skip()
        {
                if <condition on tdpur401 = true> then
                        return(true)
                endif

                return (false)
        }
```

PurchaseOrderLine.SkipPrintPurchaseOrderReminder

Skips Purchase Order Line when Printing Purchase Order Reminders.

This process extension is available from KB2215353.

To implement this process extension, you can use the information below:

Usage:

```
Process Extension PurchaseOrderLine.SkipPrintPurchaseOrderReminder 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

        table ttdpur401

Hook: ext.skip

        function extern boolean ext.skip()
        {
                if <condition on tdpur401 = true> then
                        return(true)
                endif

                return (false)
        }
```


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.

This process extension is available from KB2064131.

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:	long tfext.acp0001.set.custom.composing.fields(ref string o.composing.fields())
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):
                        tfacp240.ifbp,
                        tfacp240.otbp,
                        tfacp240.ccur,
                        tfacp240.vatc,
                        tfacp240.ptyp,
                        tfacp240.rtyp,
                        tfacp240.mcfr

                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:

                        o.composing.fields = "tfacp249.rcno, tfacp249.cdf_0001"
                        return(0)

        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 QualityManagement

The following process extension(s) is/are available:

QualityManagement.DefaultStandardTestProcedure

QualityManagement.DefaultStandardTestProcedure

Determine customer specific default Standard Test Procedure for Order Inspection Creation.

This process extension is available from KB2203645.

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:	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, 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)
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:  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
        Output: o.standard.test.procedure - The standard test procedure
                                            determined by the extension.
                o.version       - The Version determined by the extension
        Return: 0       -       Success
                DALHOOKERROR    - Otherwise.
```

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 KB2300881.

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

                table tdcmcs050

        Hook: ext.skip

                function extern boolean ext.skip()
                {
                        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)
                }
```

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

        table tdcmcs050

Hook: ext.skip

        function extern boolean ext.skip()
        {
                if <condition on tdcms050 = true> then
                        return(true)
                endif

                return (false)
        }
```

Process Extensions for SalesCheckInventory

The following process extension(s) is/are available:

SalesCheckInventory.CustomSorting

SalesCheckInventory.CustomSorting

Define a customer specific 'Sort Lines By' for Sales Check Inventory.

This process extension is available from KB2066317.

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.

Process Extensions for SalesOrder

The following process extension(s) is/are available:

SalesOrder.SkipApprove

SalesOrder.SkipPrint

SalesOrder.SkipPrintAcknowledgement

SalesOrder.SkipReleaseToWarehousing

SalesOrder.SkipApprove

Skips Sales Order when Approving.

This process extension is available from KB2169810.

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 (tdsls4211m000).
- 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

        table ttdsls400


Hook: ext.skip

        function extern boolean ext.skip()
        {
                if <condition on tdsls400 = true> then
                        return(true)
                endif

                return (false)
        }
```

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 (tdsls4405m000)

Fields that are available to be used in this Process Extension:
- All fields of table Sales Orders (tdsls400)

Note: Table must also be declared in the Process Extension.
      So skip conditions can be built on current tdsls400 data as
      instructed below.

Pseudocode:
Here you can find an example how to handle the conditions for sales
orders.

Hook: Declarations

        table ttdsls400

Hook: ext.skip

        function extern boolean ext.skip()
        {
                if <condition on tdsls400 = true> then
                        return(true)
                endif

                return (false)
        }
```

SalesOrder.SkipPrintAcknowledgement

Skips Sales Order when Printing Sales Order Acknowledgement.

This process extension is available from KB2169810.

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 (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)

Note: tables must also be declared in the Process Extension.

Pseudocode:
Below you can find an example:

Hook: Declarations

        table ttdsls400

Hook: ext.skip

        function extern boolean ext.skip()
        {
                if <condition on tdsls400 = true> then
                        return(true)
                endif

                return (false)
        }
```

SalesOrder.SkipReleaseToWarehousing

Skips Sales Order Lines and Sales Order Line Components when Releasing to Warehousing.

This process extension is available from KB2070268.

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 (tdsls4246m000)
- 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 (tdsls401) when releasing
  order lines or component lines.
- All fields of table Sales Order Line Components (tdsls463) 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 (tdsls401.cphl =
        tdcphl.component.lines), sales order line is selected first and then
        the component lines are released.
        The sales order line table is tdsls401.
        The sales order line components table is tdsls463.
        So in case of component lines, skip conditions can be built both on
        current tdsls401 data and tdsls463 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 ttdsls401
                table ttdsls463

                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 tdsls401 = true> then
                                        return(true)
                                endif

                                So in case of order_line, component handling
                                sales order lines can also be skipped based on
                                tdsls401 data.

                                break
                        case "component_line":
                                if <condition on tdsls463 = true> then
                                        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)
                }
```


Process Extensions for SalesOrderLine

The following process extension(s) is/are available:

SalesOrderLine.SkipPrint

SalesOrderLine.SkipPrintAcknowledgement

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: Declar1Gations

        table ttdsls400
        table ttdsls401

Hook: ext.skip

        function extern boolean ext.skip()
        {
                if <condition on tdsls401 = true> then
                        return(true)
                endif

                return (false)
```

	}

SalesOrderLine.SkipPrintAcknowledgement

Skips Sales Order Line when Printing Sales Order Acknowledgement.

This process extension is available from KB2181867.

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
	
	table ttdsls400
	table ttdsls401
	
	Hook: ext.skip
	
	function extern boolean ext.skip()
	{
	if <condition on tdsls401 = true> then
	return(true)
	endif
	
	return (false)
	}

Process Extensions for SalesQuote

The following process extension(s) is/are available:

SalesQuote.SkipPrintSalesQuote

SalesQuote.SkipPrintSalesQuote

Skips Sales Quote when Printing Sales Quote.

This process extension is available from KB2127961.

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 (tdsls1401m000)

Fields that are available to be used in this Process Extension:
- All fields of table Sales Quotation (tdsls100)

Note: tables must also be declared in the Process Extension.

Pseudocode:
Below you can find an example:

Hook: Declarations

        table ttdsls100

Hook: ext.skip

        function extern boolean ext.skip()
        {
                if <condition on tdsls100 = true> then
                        return(true)
                endif

                return (false)
        }
```

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 (tdsls1401m000)

Fields that are available to be used in this Process Extension:
- All fields of table Sales Quotation (tdsls100)
- All fields of table Sales Quotation Lines (tdsls101)

Note: tables must also be declared in the Process Extension.

Pseudocode:
Below you can find an example:

Hook: Declarations

        table ttdsls101

Hook: ext.skip

        function extern boolean ext.skip()
        {
                if <condition on tdsls101 = true> then
                        return(true)
                endif

                return (false)
        }
```

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.

This process extension is available from KB2177057.

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

table   ttssoc200       |* Service Orders
table   ttssoc210       |* Service Order Activities

Hook: ext.skip

        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)
        }
```

Process Extensions for StandardCost

The following process extension(s) is/are available:

StandardCost.SkipCalculate

StandardCost.SkipCalculate

Standard Cost Skip Calculation.

This process extension is available from KB2258152.

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:	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())
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 [type: domain tcccmt] 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 tcdate]
                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 tcdate]
                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_uef_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)
                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:

        A (EU1) (level 1) (Costing Source Intercompany Transfer)
            ^-----A(EU2) (level 2) (Costing Source Job Shop)
                      ^----B(EU2) (level 3)
```

```
                            ^----C(EU2) (level 3)

                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:
                        ticpr007.item is set to C
                        ticpr007.eunt is set to EU2
                The input parameters provide further information about the
                context:
                        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.

        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:
                    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:    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:

```
long tiext.cpr0002.standard.cost.skip.calculate.item(
                domain  tcitem          i.item,
        ref             boolean         o.skip.calculate,
        ref             string          o.message() )
```

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 tcdate]
                        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 tcdate]
                        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:
                    extern domain tccpcc proc_ext_std_cost_skip_calc_calculation_code
                Hook tiext.cpr0002.standard.cost.skip.calculate.item:
                    function extern long tiext.cpr0002...
                    ...
                    {
                        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:    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