Infor Enterprise Server Technical Manual

Publication Information

Release: Infor Enterprise Server 10.3.1

Publication Date: January 15, 2014

Document Code: U8172M US

Contents

About this guide

This document is a Technical Manual that describes, for various areas of the Infor Enterprise Server, how to configure and adjust the Enterprise Server to meet the environment specific needs. This document is primarily intended for administrators of the Infor Enterprise Server. Some parts can also be of interest for the Infor LN Development Tools developers.

This document contains the following chapters:

Chapter 1, “Database Tools,” describes Infor Enterprise Server tools you can use for the following tasks:

- Export a database table to a sequential file (**bdbpre**).
- Create a database table from a sequential dump or append data to an existing database table (**bdbpost**).
- Reconfigure a database table according to a new data dictionary (**bdbreconfig**).
- Check or repair the referential integrity within the database (**refint**).
- Check if the database metadata is in line with the Enterprise Server application data dictionary (**bdbvalidate**)
- Add or remove translations for multi language fields (MLFs) in ‘shadow tables’ (**bdbmlf**).

Chapter 2, “Login Process,” describes the mechanisms to communicate with the back-end.

Chapter 3, “Inter Process Communication,” describes how process communication between Enterprise Server processes, both local and remote is set up.

Chapter 4, “Native Language Support,” describes the character set translation during the communication between the Bshell and a printer.

Chapter 5, “Directory Structure,” describes how the data is structured, and where the data is located.

Chapter 6, “Information Files,” describes printer information files, which store information such as the type of printer and control characters.

Chapter 7, “Executable Programs,” describes the executable programs.

Chapter 8, “Errors,” provides a list of error codes that can occur when you work in the Enterprise Server environment.

Chapter 9, “Shared Memory Management,” discusses shared memory, which is a part of the internal memory intended for common usage.

Chapter 10, “Multibyte Management,” explains how to use the 32 bit-wide character space, which is implemented in Enterprise Server to support all possible character sets.

Chapter 11, “Audit Management,” describes the audit management facility, including the audit server, the audit management utility, and where audit-data is stored.

Chapter 12, “OLE Automation,” describes how Enterprise Server works with OLE Automation.

Chapter 13, “LN Environment Variables and Resources,” describes Environment variables you can use in the LN environment.

Chapter 14, “Document Authorization” introduces Document Authorization, also known as Database Change Management (DBCM).

Contacting Infor

If you have questions about Infor products, go to the Infor Xtreme Support portal at http://www.infor.com/inforxtreme.

If we update this product or document after the product release, we will post the new version on this Web site. We recommend that you check this Web site periodically for updates.

If you have comments about Infor documentation, contact documentation@infor.com.

Database Tools

1

General

The database tools are designed for database management, and are independent of how the database is organized. At run time, the system selects a database driver, depending on the contents of the Tabledef file. You can then use these database tools to do the following:

- Export data from a database table to a sequential (**bdbpre**) file.
- Create a database table from a sequential file or append data to an existing database table (**bdbpost**).
- Reconfigure a database table according to a new data dictionary (**bdbreconfig**).
- Check or repair the referential integrity in the database (**refint**).

If changes are made in the data dictionary, for example, if fields are added or field lengths are changed, you can reconfigure the old table to a new one. The data dictionary changes are incorporated in the new table, which also contains the data dictionary table data.

The system you use determines the extension for the executables. For example, **bdbpre** on Windows is called bdbpre.exe. On UNIX the file is called bdbpre6.2.

The remainder of the document uses the short naming convention, without 6.2 and/or .exe extension. Some binaries have a slightly different name on Windows, for example:

Unix name	Windows name
bshell6.2	ntbshell.exe
explode6.2	bic_explode.exe

In this document, the term UNIX is used frequently. This term comprises various UNIX versions, including Linux if applicable/supported.

The following sections describe each of the database tools.

bdbpre

Name

bdbpre

Export database tables to a sequential file.

Related session: Create Sequential dump of Table (ttaad4226m000)

Synopsis

bdbpre [**–uUvVsxkK**] [**–p** *package combination*] [**–t** *sep*] [**–o** *dir*] [**–N** *table* [*table ..*]] [**–I** *file*] [**–E** *file*] [**–O** *file*] [**–M** *size*] [**-L** *lang*] **–g** [**-W** *dbcmoptions*] [**-X** *traceoptions*] [**-Y**] [**–C** *company number list/range*]

Description

The **bdbpre** tool reads selected tables for given company numbers, exports the data from the tables into sequential data, and writes the data to a standard output. You can then redirect this output to a file that can be used as input for **bdbpost**.

The **bdbpre** tool prints information such as names, the number of records, and any errors. You can use the **–s** option to suppress the messages produced by **bdbpre** at run time. This option is useful when you use the output of **bdbpre** as the direct input to **bdbpost**.

ION Workflow operation

If a table is part of a Workflow model, it is checked whether all tables are part of the workflow model in the table dump. The table list is verified against the workflow models to see whether all tables of a model are included in a dump, and whether a table has been deployed in a company.

If a table is part of a Workflow model, additional information is dumped in the table dump headers. This information includes the model and the deployment related to the tables dumped. This information is used by **bdbpost** to validate the dumped model and deployment against the model and deployment in the environment during import.

To control some of the workflow related actions, see the **–W** <dbcm options>.

To find information on DBCM related trace output, use the **–X** <trace options>. Add **-Y** to add even more trace output.

Note: The **–t** or **–x** options are not supported when dumping workflow data, unless you use the **–W ignoredbcm** option.

Table dump headers are only added to the dump if a table with DBCM is enabled and deployed.

The *fd6.2.package combination* file is always used to search dictionary information

You can use **bdbpre** in the following ways:

- Specify a range of company numbers to convert a database table for more than one company number, for example:

```
bdbpre –Ntimcs099 –C000-005 > timcs099_dump
```

- Convert a database table for given company numbers, for example:

```
bdbpre –Ntimcs088 –C000 004 005 > timcs088_dump
```

- Specify all the tables for which you want to create a sequential dump in some file in package module format. The –I option reads this file and creates a sequential dump for each table and the specified company numbers, for example:

```
bdbpre –Iseqfile –C000-009 > BIGdump
The contents of the seqfile:
timcs001
timcs002
timcs003
........
timcs099
```

Possible options:

- **–u/U**

  Print usage information.

- **–v/V**

  Print information about the version of **bdbpre**.

- **–p *package combination***

  The name of package combination that is to be used.

- **–s**

  Suppress error messages, statistics, and so on.

- **–C**

  Company numbers for a given table, in the following two formats:

  - Specific company numbers, for example, 001 002 003
  - Range of company numbers, for example, 001-999

- **–d**

  Specify a specific driver. If you use this option, bdbpre bypasses the tabledef6.2 file.

You can use the **-d** option to copy data from one instance of a database to another, for example:

```
-d "oracle(ORACLE_HOME=/usr/oracle, ORACLE_SID=D1)"
```

Convert Informix database to a sequential dump. The database driver is explicitly Informix.

```
bdbpre -dinformix -Ntimcs000 -C000-004 >
timcs000_dump
```

The database driver is retrieved from `$BSE/lib/tabledef6.2`:

```
bdbpre -Ntimcs000 -C000-005 >
timcs000_dump
```

- **-g**
  Skip export of logical linked tables.
  Without using –g all tables will be exported, also the tables that originate in another physical company.
- **–N *table* [*table …*]**
  Database table name. Specify specific table names, for example, –N timcs000 timcs016. The *–I* and *–N* options are mutually exclusive.
- **–I *file***
  Input file with table names. The *–I* and *–N* options are mutually exclusive.
- **–E *file***
  Redirects errors and information to file.
- **– O *file***
  Redirects output to file.
- **–q *output file***
  Deprecated: Use –E and –O instead.
  Redirect terminal output to the output file.
- **–k**
  Drop table after the dump is created.
- **–K**
  Drop table. Create a backup before dropping. You can only use this option if supported by the DBMS.
- **–t *separator character***

Create a sequential file in which the fields have variable length and are separated by the separator character (|). Whenever you specify the **–t** option, a sequential file (.S) is created for each table in the current directory, unless the **–o** option is specified.

```
bdbpre -t"|" -Ntimcs016
-C000
```

Creates `timcs016000.S` in the current directory. This sequential file is an ASCII dump of the timcs016000 table, in which case the separator character separates the fields.

The -x and -t options are mutually exclusive.

- **-L *lang***

  In an environment where multiple data languages are used, it is possible to make a dump with just one data language. The –L option can be used to specify the data language to be exported. The data language is a code according to ISO639.2 and must be configured in the target environment.

  The default behavior, without using the –L option, is to export all data languages.

- **–x**

  Create a sequential file in which the fields have a fixed length and are not separated with a separator character. Whenever you specify the **–x** option, a sequential file (.F) is created for each table in the current directory, unless the **–o** option is specified.

  bdbpre –x –Ntimcs016 –C000 creates timcs016000.F, which is an ASCII dump with fixed-length records.

- **–o**

  Can be specified with the options **–x** and **–t** to specify the directory name in which the sequential file is created (.S or .F).

- **–M *size***

  Can be specified with the options **–x**, and **–t** to specify the maximum size of the output file. If the maximum file size is reached, a new file with a sequence number is opened and filled. You can specify size as any number or a number followed by K, M, or G, for kilobytes, megabytes, or gigabytes, respectively. The maximum size is 2 GB.

- **-W all**

  Create a dump with both checked-in and checked-out data if applicable.

- **-W ci**

  Create a table dump with only checked-in data.

- **-W nomodelcheck**

  Allow to dump a table which is part of a model where other tables that are part of this model are not dumped. Workflow information is dumped in the table dump headers and incomplete modules are marked as incomplete.

- **-W notablesharingcheck**

Allow to dump tables and companies where not all the companies are present in the company set(s) that define a deployment. This applies only to models and deployments that use the ForeignCompnr in the Relation section of an object model.

- **-W ignoredbcm**

  Completely ignore DBCM information. No checked-out data is dumped and no workflow information is stored in the table dump headers.

- **-W type=*dbcm model type***

  Dump all tables which are found in <dbcm model type>.

  For example, this command dumps all tables found in model chm100:

```
bdbpre –W type=chm100 –C000-005 > dbchm100_dump
```

  The options –W type=*dbcm model type* -I and -N are mutually exclusive.

  You can specify multiple -W type=type options on the command line. This may be useful if a table is shared among multiple models.

- **-X stderr**

  Redirect bdbpre trace information to stderr.

- **-X *filename***

  Redirect trace output to filename.

- **-Y**

  Additional tracing information.

See also

bdbpost

Note: If you pipe your bdbpre output directly to bdbpost without giving the –s option, messages on the screen will appear jumbled.

bdbpost

Name

bdbpost

Add data from a sequential dump to a new or existing database table.

Related session: Create Table from Sequential Dump (ttaad4227m000)

Synopsis

bdbpost [**–I** *input file*] [**–{E}** *output file*] [**–uUvVARTfilxnmkK**] [**–p** *package combination*] [**–e** *file*] [**–D** *seq_dir*] [**–t** *sep*] [**-r** *row/trans*] [**-L** *lang*] [**–c** *compnr*] [**-W** *dbcmoptions*] [**-X** *traceopts*] [**–C** *compnr range*] [*pattern*]

Description

The **bdbpost** tool reads data from standard input unless **–I** option is provided and creates a new database table if that table does not exist. If the **Append** option is selected, this option appends data to an existing table.

The **bdbpost** tool also compares current data dictionary information with the information in the dump. If the information does not match, the **bdbpost** tool returns an error. If the current data dictionary is not present, the tool creates a data dictionary based on the dump.

For each table, **bdbpost** prints information such as the table name, indexes, the number of records, and any errors.

Note: If you import a dump that contains Workflow tables, you must specify**–n**.

The **bdbpost** tool is used to import data from sequential dumps that are created with **bdbpre**.

The following example shows how to use a sequential dump created by **bdbpre**.

On system 1:

```
bdbpre –Ntimcs016 –C000-003 > timcs_dump
```

On system 2:

```
bdbpost < timcs_dump
```

The following example demonstrates the use of sequential dumps that are created with the **–x** or **–t** option of **bdbpre**. In this case the **–D** option is mandatory to retrieve the directory name in which .S or .F files are stored:

```
bdbpost –ddb2 –t"|" –D./seqdir
bdbpost –x –D./seqdir
```

- **–D *directory name***

  Specifies the directory in which the sequential files, which are files with extension .S or .F, are located. Use this option if you have used the **–x** or **–t** option of **bdbpre** to create the dump files.

- **-L *lang***

  In an environment where multiple data languages are used, it is possible to import a dump where a column is not an MLF (multi language field) into a table where that column is an MLF. The –L

option can be used to specify the data language that should be used to insert the data into the table. The data language is a code according to ISO639.2 and must be configured in the target environment.

- **-T**

  The default behavior of bdbpost is to preserve the timestamp values of the rcd_utc column (if present). This way, a straightforward export/import is neutral with regards to the data contained in the table. If the –T option is specified, all timestamp values of the rcd_utc column are updated to the time at which the import is done.

- **–t *separator character***

  Use this option if you want to import data from a sequential file that is created with the **–t** option of **bdbpre**. If you use the **–t** option, you must also specify a directory with the **–D** option.

Example

To load a sequential dump into the ttimcs016000 table, first move the sequential dump to the ttimcs directory with the name ttimcs016000.S. **bdbpost –Dttimcs –t"|"** searches for an .S file in ttimcs and, if that file is found, the corresponding tables are created or appended.

Note: With the **–D** option, all .S files in that directory are used to create and append the tables, therefore, you must make sure to remove unwanted .S files before you run **bdbpost.**

- **–x**

  Use this option to load a sequential dump with fixed-length records without any separators. Use this option to import data from a sequential file that is created with the **–x** option of **bdbpre**. If you use the **–x** option, you must also specify a directory with the **–D** option.

Example

Suppose you have used bdbpre to create sequential dumps in the directory dumps. **Bdbpost –D –x** searches for .F files in dumps and for each file found, the corresponding table is created or appended.

The following parameters can be used for **bdbpost**:

- **–u/U**

  Print usage information.

- **–v/V**

  Print information about the version of **bdbpost**.

- **–p *package combination***

  The name of the package combination to be used.

- **<pattern>**

  Pattern to specify tables that are filtered out of the dump. Wildcards such as * and ? are allowed.

- **–c**

  Company number for the tables to be created.

- **–C**

Range of company numbers on which to perform the **bdbpost** operation. This must be the last option specified in the command, other than the **<pattern>** option.

- **–x**

  Import files with fixed length records. Use this option to import files created with the **–x** option of **bdbpre**.

- **–D *directory name***

  Specifies the directory that contains the files that must be imported.

- **–e *file***

  File to store the names of unsuccessfully created tables.

- **-g**

  Skip import of logical linked tables.

  This parameter is only of use when the append option is chose [-A].

- **–k**

  The existing tables are dropped.

- **–K**

  The existing tables are dropped after a backup is made, if DBMS supports this.

- **–l**

  Display contents of the dump file.

- **–l *file***

  Redirects input from *file*.

- **–E *error***

  Redirects errors to *error*.

- **–m**

  Ignore domain constraints. Data will be imported even if the data does not fit the domain constraints. Be careful with this option because this involves importing data that does not match the application criteria.

- **–i**

  Ignore domain range error and skip record.

- **–n**

  Ignore referential integrity constraints. To be used if a range of tables is imported. Make sure to repair the reference counters afterwards using the Reorganize Tables (ttaad4225m000) session.

- **–A**

  Append to an existing table. If duplicate records exist, do not overwrite the records from the dump. Create the table if the table does not already exist.

  If –A option is not given the import will fail if the table already exists.

- **–R**

Append to an existing table or create a new one. If a record already exists, the record in the dump replaces the table. A summary is provided at the end. Note that only the existence of the primary key is checked.

If a primary key exists, the record is replaced. If the primary key does not exists but a secondary key exists, error 100, duplicate record, occurs.

- **–t *separator character***

  Specify the used separator. Use this option if the dump file was created with the –t option of **bdbpre**. With this option the **–D** option is required.

- **–f**

  Fast mode. First the rows are inserted and afterwards the indexes build resulting in a balanced Index tree in the database.

  The import itself is not faster when using **–f**.

  If you use the –f option, be aware of the following:

  - If you interrupt **bdbpost**, this can result in table inconsistency.
  - You cannot create an index in case of a duplicate conflict.
  - For large tables, adding indexes can take a long time, often more than 15 minutes.

- **–M**

  Can be specified with the options –x, -t or –I to specify that the input file consists of multiple files.

  If **bdbpre** was executed with this option the **bdbpost** must be run with **–M** as well. For more information, see also the **–M** option of **bdbpre**.

- **–r *rows/transaction***

  Defines after which number of inserted rows a commit is performed. The default value is 100. A number less than 100 is changed to 100.

- **-W all**

  Import all data, both checked-in and checked-out data if applicable, from the table dump.

- **-W ci**

  Import only checked-in data. Ignore checked-out data in the table dump.

- **-W nomodelcheck**

  Allow to import a table that is part of a model, where other tables that are part of this model are not present in the dump. Workflow information that was dumped in the table dump headers, and incomplete modules that were marked as incomplete, can be imported.

- **-W notablesharingcheck**

  Allow to import tables and companies, where not all the companies are present in the company set(s) that define a deployment. This applies only to models and deployments that use the ForeignCompnr in the Relation section of an object model.

- **-X stderr**

  Write trace data to stderr.

- **-X *filename***

Write trace data to filename.

- **-Y**
  Add additional trace information.

Examples

```
bdbpre –Nttadv000 –C000-010 > dump
bdbpost < dump
```

Creates all tables in a. dump

```
bdbpost –l < dump
```

Gives the names of tables in the dump.

```
bdbpost –C000-005 < dump
```

Creates tables only in the given company range.

```
bdbpre –Nticom000 –C000-010 > dump
bdbpost –C000-005 ticom* < dump
```

Creates tables only in the given company range and where the table name matches the ticom pattern.

```
bdbpost –R –C000-005 ttadv* < dump
```

Creates and appends tables only in the given company range and where the table name matches the ttadv pattern. If duplicate records exist, these records are replaced with records from the dump.

If you use the **–m** or **–n** option, the data in the database can violate the LN integrity constraints. Data can violate the LN domains or can violate LNLNLN referential integrity.

See also

bdbpre

bdbreconfig

Important: This executable is called by 4GL sessions. Do not run it directly (except for the change analysis feature, see the description of the -A option).

Name

bdbreconfig

Reconfigure database tables according to new data dictionary. Optionally analyses differences between database versions.

Synopsis

bdbreconfig [**-A** *pacc*] [**–C** *company numbers*] [**–c**] [**–E** *file*] [**-F** *file*] [**–I** *file*] [**–M** *size*] [**–m**] [**–N** *table* [*table* [...]] [**–p** *package combination*] [**–R** *file*] [**–r** *rows/trans*] [**–s**] [**–t** *directory*] [**–UuvVZ**] [**-J N**]

Description

The **bdbreconfig** tool converts tables in the database so that the physical tables in the database match the new metadata as defined in the definition data dictionary file. It can also analyze the differences between two database definitions and report the changes that are required to bring the database into a new state. You can use this to predict the required conversion time, spot potential performance bottlenecks, and gain insight into changes introduced by an upgrade procedure. For details, see the description of the -A (analyze mode) option.

For each table that must be converted, the **bdbreconfig** tool expects that the following two files exist:

- A data dictionary file of that table, for example, dtfgld106.
- A data dictionary file of that table with a .new extension, for example dtgld106.new.

The **bdbreconfig** tool compares the two data dictionary files to determine what changes must be made to the existing table. Based on this comparison, and based on the capabilities of the database, the **bdbreconfig** tool will try to find an optimum method to convert the table.

After successful completion, the table definition matches with that of the new version of the data dictionary file. Therefore, to complete the conversion, the existing data dictionary file must be replaced with the new version. The **bdbreconfig** tool will not perform this procedure. Therefore you must perform this procedure manually.

Depending on the type of changes, and depending on the capabilities of the database, **bdbreconfig** will use one of the following three methods to convert a table.

Method 1: Database reconfiguration

With this method, the table is converted in the following steps:

1. Indexes that must be changed or dropped are dropped.
2. The table is converted by issuing a single SQL ALTER TABLE command to the database.
3. Indexes that must be changed or created are created.

Method 2: Database reconfiguration with export

With this method, the table is converted in the following steps:

1. The table data is exported to a temporary file. The name of this file is R.*<table name><company number>*, for example, R.tfgld106590.
2. Indexes that must be changed or dropped are dropped.
3. The table is converted by issuing multiple SQL ALTER TABLE commands to the database.
4. Indexes that must be changed or created are created.

After successful completion, the temporary file will be deleted. The temporary file is used by **bdbreconfig** to restore the original table when reconfiguration fails.

Method 3: Reconfiguration with export/import

With this method, the table is converted by taking the following steps:

1. The table data is exported to a temporary file. The name of this file will be R.*<table name><company number>*, for example, R.tfgld106590.
2. The table is dropped.
3. The table is created according to the new data dictionary definition.
4. The data is imported from the temporary file into the new table.
5. The indexes are created.

After successful completion, the temporary file will be deleted.

Note: The **bdbreconfig** tool does not check referential integrity constraints. You must use the **refint** tool to repair the references after you reconfigure a table. Running the **refint** tool on all tables that are reconfigured can be time-consuming. To obtain a list of tables that actually need **refint** to repair references, you must run **bdbreconfig** with the **–F** *file* option.

To get a detailed report about the actions that are required to update the database, see the description of the -A (analyze mode) option.

The following options are available with **bdbreconfig**:

- **-A *package combination***

  This turns analysis mode on. The analysis mode of **bdbreconfig** helps in diagnosing performance bottlenecks when upgrading to a newer version of LN. It can also be used during development to gain insight in the resulting reconfiguration process and can help to prevent expensive conversions. This option is intended to be used from the command line.

  When running in analysis mode, ***no actual reconfiguration is performed, ever***. The analysis mode only analyses the differences and produces a report (on stderr, redirect using the -e command line option). This report lists all the alterations that must be made to the database and the other actions, such as export/import, that must be performed to upgrade the system. See also the description of the -c option (check-only); -A implies -c.

  The -A option accepts a package-combination argument and can be used on the command line in several ways to specify source and target database configurations:

1. Use -A once. The analysis is between the current database configuration (*source*) and the specified package combination (*target*).
2. Use -A twice. The analysis is between the package combinations you specify. The first package combination you specify is the *source* configuration, the second one is the *target.*.
3. Use -A once, combined with the -p option. The package combination specified behind -p is the *source*; the package combination specified behind -A is the *target*.

The resulting report shows the database actions that are required to upgrade from the source configuration to the target. The report shows this information:

- The changes, such as adding/removing columns, changing the characteristics of a column, and adding/removing indexes.
- The consequences of those changes.
- The exact SQL statements that are required by the database to actually apply the changes.

Many changes can be performed entirely by the database (Oracle, DB/2, and so on).

For example, extending the length of text string is a change that is very fast on Oracle, independent of the number of rows in the table. But some changes require a full table export/import, for example adding a character-constraint to a field, the constraint-check is performed by the bdbpost program when it imports the data.

For every detected change, a line is printed in the report that details the specific change. This is followed by an indication of the required reconfiguration method for that specific change. When a single table has multiple changes, the "heaviest" change decides which upgrade method is chosen for that table.

The report makes this visible. You can specify tables for analysis in the normal way, see the description of the -N and -I options. If you do not specify any tables at all, **bdbreconfig** attempts to find *all* tables that exist in the original package combination and reports on them all. The following code is an example invocation of a **bdbreconfig** analyze run:

```
bdbreconfig6.2 -m -c -A b61a3 -A b61a8 -C 0
…
bicit331000  No reconfiguration is required
bicit332000  No reconfiguration is required
bpmdm000000  Reconfiguration is required
             Column 'izbh' is new (ENUM). ALTER TABLE required.
             Column 'rstr' is new (ENUM). ALTER TABLE required.
             Column 'tefp' is new (ENUM). ALTER TABLE required.
             ORACLE: EXECUTE: ALTER TABLE details omitted

bpmdm001000  Reconfiguration with export/import is required.
            Column 'bano' is new (STRING(34)). ALTER TABLE required.

             Column 'cedt' is new (DATE). ALTER TABLE required.
             Column 'mail' (TSS_STRING(100)) changed length (from 50
 to 100). ALTER TABLE required.
             Column 'mail' changed from single byte to multibyte.
ALTER TABLE not possible.
```

This code shows a **bdbreconfig** invocation without explicit table names. Therefore a full report of all tables is created. In this case there are almost 4000 tables; the resulting output is over 15.000 lines. Many tables are unaltered (`no reconfiguration is required`).

The bpmdm000 table has some new columns, all of which can be added by issuing database commands to Oracle (the exact statement that is required is shown in the full report, omitted here for readability). The exact statements for DB/2, Oracle, and so on are printed in the report. The reconfiguration of table bpmdm000000 will be fast because the database can handle it.

The next table is bpmdm001000. The first line shows that a full export/import is required for this table. A few columns are new (which could be handled by the database), but the 'mail' column changed both length and type (single byte to multibyte). Therefore a data conversion is required (ALTER TABLE not possible), leading to a full export/import (shown on the first report line for the table).

To find potential performance bottlenecks, study the report looking for tables that require a full export/import and look for the reason. For example:

```
cisli107000  Reconfiguration with export/import is required.
             Column 'brid' (STRING(9)) has new adjustment 'LEFT' in
domain.
             Data conversion required.
             ALTER TABLE not possible due to domain constraints or
data conversions
```

This code shows that a string is now explicitly left-adjusted, which means that any leading spaces that may be in the database must be removed. Therefore, the entire table is exported and imported. Maybe the adjustment was an unintentional side-effect of assigning a domain to the column. Dropping the adjustment saves an export/import operation. Every time you see "Data conversion required", the change triggers the export/import operation. As another example:

```
cprpd210000  Reconfiguration with export/import is required.
             Column 'dsca' (TSS_STRING(30)) changed adjustment (from
 'NONE' to 'LEFT').
             Data conversion required.
             ALTER TABLE not possible due to domain constraints or
data conversions (see above)
```

The column has changed to a left-adjusted type, therefore leading spaces must be removed. This is a typical example of a change that *will* require an export/import.

- **–C *company numbers***

  The tables will be reconfigured in the companies that are specified by the –C option. You can specify the company numbers in the following two ways:

  - Specific company numbers, for example: –C 001 002.
  - Range of company numbers, for example: –C 001-010.

- **–c**

With this option, the **bdbreconfig** tool displays, for each table and company, if reconfiguration is required and displays which reconfiguration method the tool will use. The actual reconfiguration is not performed.

- **–E *file***

  Redirects errors to error file *file*.

- **–F *file***

  Creates a list of table names and company numbers for which you must run **refint** to fix the reference counters after **bdbreconfig** is finished. This list is stored in *file*. For older versions of the porting set, you had to run **refint** on all tables that were reconfigured. The **–F** option generates the minimum set of table names; that is: a table is only added to the list if the reconfiguration affects data that is actually used by reference counters.

  When you run **refint**, you must specify the resulting *file* using the **-s -l *file*** options. In this way, **refint** only fixes the tables listed in *file*.

  Using the **–F** option of **bdbreconfig** can significantly speed up the reconfiguration process.

- **–l *file***

  Reconfigure all tables whose names are listed in *file*. Each table name in *file* must be on a separate line. A table name can include a company number. For example, table tccom100 in company 570 is specified as tccom100570. If a table name does not include a company number, the table is reconfigured for the company numbers that are specified by the –C option. If the table does include a company number, this table is reconfigured for this specific company only and, for this table, the company numbers that are specified by the –C option are ignored.

- **-J N**

  Specifies the number of parallel tasks that **bdbreconfig** will execute. Every reconfiguration operation on a single table is a "task". Parallel execution can speed up the reconfiguration of many tables. The minimum valid value is 1 (sequential behavior), the default is 4, the maximum is 128. Be careful when selecting this value: too much parallelism can cause system overload and database connection problems.

  You can also set the parameter using the bdbreconfig_parallel resource value or using a BDBRECONFIG_PARALLEL environment variable.

  Analysis (-A) and check-only (-c) modes will also execute faster due to parallelism.

- **–M *size***

  Specifies the maximum size of the dump file. If the maximum file size is reached, a new file with a sequence number is opened and filled. The size can be specified as any number (bytes) or a number followed by K, M, or G, for kilobyte, megabyte, or gigabyte, respectively. The maximum size is 2 GB.

- **–m**

  Disable domain constraints. If you use this option, the **bdbreconfig** tool might be able to choose a more optimal reconfiguration method. However, after the table is reconfigured, the data in the table may violate LN domain constraints.

- **–N *table* [*table …*]**

A list of tables to be reconfigured. A table name can include a company number. For example, table tccom100 in company 570 is specified as tccom100570. If a table name does not include a company number, the table is reconfigured for the company numbers that are specified by the –C option. If the table does include a company number, this table is reconfigured for this specific company only and, for this table, the company numbers that are specified by the –C option are ignored.

- **–p *package combination***

  The name of the package combination to be used.

  When combined with -A, the -p combination is used as the source configuration.

- **–R file**

  Create a report file that describes, for each table and company, the result of the reconfiguration. This option is described in more detail in one of the following sections.

- **–r *rows/transactions***

  Defines after which number of inserted records, a commit is performed. The default value is 100. A number less than 100 is changed to 100.

- **–s**

  Suppresses error messages and other information.

- **–t *directory***

  The directory name for temporary dump files.

- **–U/-u**

  Usage information.

- **–V/-v**

  Version information.

- **–Z**

  Reorganize the table. If you specify this option, the tables are not converted but only recreated. With this option, you do not need a data dictionary file with the .new extension. If a data dictionary file with the .new extension exists, the file will be ignored. The table will be recreated by performing the following steps:

  1. The table data is exported to a temporary file.
  2. The table is dropped.
  3. The table is recreated.
  4. The data is imported from the temporary file.
  5. The indexes of the table are created.

The **bdbreconfig** tool reconfigures only one table of given company numbers at a time.

Example

```
bdbreconfig –Ntimcs016 –C001
```

Reconfigures timcs016 for company number 001.

```
bdbreconfig –Ntimcs016 –C000-010
```

Reconfigures timcs016 for a range of company numbers.

Note: Make a copy of the dump before you perform the following:

In case of interrupted reconfiguration while the R.table file still exists, the following command uses the R.ttadv100222 dump to rebuild the table:

```
bdbreconfig –N ttadv100222+
```

Example

```
bdbreconfig –Ntimcs016 –C000-003
```

Reconfigures timcs016 for some company numbers according to the new data dictionary, provided that dtimcs016 and dtimcs016.new are present.

If you specify the option –R, **bdbreconfig** creates a report that describes, for each table and company, the result of the reconfiguration. If you use the –R option in combination with the –c (check) option, this report describes the result of the check for each table.

Each line in the report has the following form:

```
<tablename><compnr><space><status code><space><error><space><message
text><newline>
```

The following table lists the possible status codes:

Status code	Description
R0	The table is reconfigured successfully. The Error field contains the value 0, or contains error code 506 if the table does not exist.
R1	The reconfiguration has failed. The Error field will contain the code of the error that caused the failure.
R2	The table is reconfigured successfully, but one or more indexes could not be created. The Error field will contain error code 114.
C0	No reconfiguration is required for this table. This only means that no action is required in the database for this table. If a new data dictionary file exists for this table, this data dictionary file can still differ from the current file and, therefore, must replace the existing file.
C1	An error occurred while checking. The Error field will contain the actual error.

Status code	Description
C2	Reconfiguration is required for this table and the database can handle the entire reconfiguration. The reconfiguration will be carried out using reconfiguration method 1.
C3	Reconfiguration is required for this table. The database can handle the entire reconfiguration. However, if this table is reconfigured, **bdbreconfig** must create an export of the table data before reconfiguring the table. The reconfiguration will be performed using reconfiguration method 2.
C4	Reconfiguration is required for this table. The database cannot handle the reconfiguration. The reconfiguration will be performed using reconfiguration method 3.

If **bdbreconfig** is run with the –c option, only the status codes that start with a C will be reported. Otherwise, only the status codes that start with an R will be reported.

The following example provides a sample report produced by **bdbreconfig** that was run without the –c option.

Example

```
dbtst100112 R0 0
dbtst100113 R1 2430 Reconfiguration failed (error 2430)
dbtst100114 R0 506
dbtst120112 R1 205 Insert failed (error 205)
dbtst120113 R0 506
dbtst120114 R0 506
dbtst160112 R2 114 Reconfiguration failed (error 114)
dbtst160113 R0 506
dbtst160114 R0 506
```

The following example provides a sample report produced by **bdbreconfig** that was run with the –c option:

Example

```
dbtst100112 R0 0
dbtst100113 R1 2430 Reconfiguration failed (error 2430)
dbtst100114 R0 506
dbtst120112 R1 205 Insert failed (error 205)
dbtst120113 R0 506
dbtst120114 R0 506
dbtst160112 R2 114 Reconfiguration failed (error 114)
dbtst160113 R0 506
dbtst160114 R0 506
```

Exit code

The exit code of **bdbreconfig** is either 0 or 1. If you do not use the –c option, the exit code is **1** if at least one of the tables could not be reconfigured. If all tables are reconfigured successfully, the exit code of **bdbreconfig** is 0.

If you use the c option, the exit code is **1** if the check did not succeed for at least one table. Otherwise, **bdbreconfig** will have an exit code of **0**.

bdbvalidate

Name

bdbvalidate

Validate database contents according to the data dictionary.

Synopsis

bdbvalidate [**–HuUvVsz** [**–N** *table* [*table* [...]] [**–E** *file*] [**–I** *file*] [**–R** *file*] [**–p** *package combination*] [**–C** *company numbers*]

Description

The **bdbvalidate** tool reads tables for given company numbers, verifies if the data is according to the domain constraints as defined in data dictionary, and writes a report to the file specified by the –R option.

The **bdbvalidate** tool will verify a number of internal consistency rules for multi language fields (MLFs).

The **bdbvalidate** tool prints information such as names, the number of records, and any errors. You can use the **–s** option to suppress the messages produced by bdbvalidate at run time.

The following options are available with bdbvalidate:

- **–U/u**
  Usage information.
- **-H**
  Check for high ASCII characters in single byte strings.
- **–V/v**

Version information.

- **–p *package combination***

  The name of the package combination to be used.

- **–s**

  This suppresses error messages and other information.

- **–z *number***

  This option limits the number of reported errors for each table.

- **–C**

  Company numbers for a given table in these formats:

  - Specific company numbers, for example, 001 002.
  - Range of company numbers, for example, 001-010.

- **–I *file***

  Validate all tables which names are listed in file. Each table name in file must be on a separate line. A table name may include a company number. For example, table tccom100 in company 570 is specified as tccom100570.

- **–N *table* [*table…*]**

  List of tables to be reconfigured.

- **–E *file***

  Redirects errors to error file.

refint

Important: This executable is called by 4GL sessions. Do not run it directly.

Name

refint

Check or repair the referential integrity in the database.

Synopsis

refint [**–uU**] [**–vV**] [**–I** |**–L** *file*] [**–c**] [**–r**] [**–s**] [**–p** *package combination*] [**–I** *infile*] [**–O** *outfile*] [**–E** *errfile*] [**–N** *table* [*table* [...]] [**–C** *compnr* [*compnr* [...]]

Description

The **refint** tool checks the integrity of references in the database. You can also use it to repair the integrity of a corrupted database. Integrity refers to the accuracy or validity of data.

You can use the following options:

- **–u/U**

  Print usage information.

- **–v/V**

  Print version information.

- **–p *package combination***

  The name of the package combination that is to be used.

- **–C *compnr***

  Specify the company number.

- **–c**

  Check validity of references with the **Report Only** option. Undefined references to the specified tables are logged in the ref_undefined file.

- **–r**

  Check validity of references and repair reference counters. From all specified tables, the reference counters are checked and changed if required. Handled undefined references to the specified tables are logged in the ref_undefined file.

- **–l/-L *file***

  Creates a file that contains all undefined references. If you use the option –l, the name of this file will be ref_undefined. With the option –L, you can specify your own file name.

- **–l *file***

  Check or repair all tables whose name appears in the *file* file. Each table name in *file* must be on a separator line. A table name might include a company number. For example, table tccom100 in company 570 is specified as tccom100570.

  The **-F** option of **bdbreconfig** can produce such files.

- **–N *table***

  The name of the table whose references must be checked or repaired. The table name might include a company number. For example, you can specify –N ttadv100 or –N ttadv100999.

- **–s**

  Handle specified tables only.

Examples

```
refint –Iref_tables –c–C100
refint –Ntimcs016 –l
```

bdbmlf

Important: This executable is called by 4GL sessions. Do not run it directly.

Name

bdbmlf

Add or remove translations for multi language fields (MLFs) in ‘shadow tables’.

Synopsis

bdbmlf [**–uUvVsr** [**–N** *table* [*table* [...]] [**–E** *file*] [**–I** *file*] [**–R** *file*] [**–p** *package combination*] [**–C** *company numbers*] [**-d**]

Description

Multi Language Fields (MLFs) are fields for which each record in the table has values in a number of data languages. When a data language is added, it is necessary to run the **bdbmlf** program in order to establish the fallback to the base language for the new data language in the database. When a data language is removed, it is necessary to run the **bdbmlf** program in order to remove values from the database.

Caution: The **bdbmlf** program is used by sessions in LN Tools. Do not use it from the command line, or the database may become inconsistent.

The **bdbmlf** tool reads tables for given company numbers, adds or removes values for certain data languages to the database, and writes a report to the file specified by the –R option.

The bdbmlf tool prints information such as names, the number of records, and any errors. To suppress the messages produced by **bdbmlf** at run time, use the **–s** option.

It is possible to inspect the database without making changes by omitting the –r option (repair). There is a ‘detail’ modus in which the program checks for various inconsistencies with regard to translations (option –d).

The following options are available with **bdbmlf**:

- **–U/u**
  Usage information.
- **–V/v**
  Version information.

- **–p *package combination***

  The name of the package combination to be used.

- **–s**

  This suppresses error messages and other information.

- **–C**

  Company numbers for a given table in these formats:

  - Specific company numbers, for example, 001 002.
  - Range of company numbers, for example, 001-010.

- **–I *file***

  Process all tables whose names are listed in *file*. Each table name in *file* must be on a separate line. A table name may include a company number. For example, table tccom100 in company 570 is specified as tccom100570.

- **–N *table* [*table…*]**

  List of tables to be processed.

- **–E *file***

  Redirects errors to error file.

- **-r**

  This option turns on the repair modus: the program actually fixes the problems that are reported.

- **-d**

  This option turns on the ‘detail’ modus. The program will check for missing translation records for all languages in the database.

- **EXIT STATUS**

  The exit status of the bdbmlf program is a bit pattern that can have a value that is a combination of the values below. Note that the values are in octal representation:

EXIT status values		
NAME	**Value**	**Description**
INIT_ERROR	001	Initialization error. If this occurs, the bdbmlf program halts.
ERROR	002	An error for one or more tables occurred.
RET_REPAIR	004	There are changes to do (if –r not specified: the changes are not executed).
RET_LOCKED	010	when executing repair actions, there were locking errors.

The bits in the bit pattern are set in various circumstances, specified in the following table (this table is not exhaustive):

Example error codes		
NAME	**ERROR**	**Report entry**
unknown user	INIT_ERROR	n/a
unknown PACC	INIT_ERROR	n/a
compnr range param wrong	INIT_ERROR	n/a
init bdb api problem	INIT_ERROR	n/a
no rows	-	0
Table does not exist	-	0 table does not exist
Logical table	-	0 ignoring logical table
Unknown table	ERROR	513 unknown table ‘ppmmmnnn’
dd corrupt	ERROR	512 dd corrupt
Invalid table name	ERROR	513 invalid table name
BDB errors	ERROR	nnn Message with details
Something (to) be repair(ed)	REPAIR	0 Message with details
Lock error during repair	LOCK|ERROR	Skip record due to LOCKED

Login Process

2

LN supports these mechanisms to communicate with the back-end:

- Rexec protocol
- ES Logic Service (Blogin daemon)
- SSPI (Windows only)
- Single-sign on (SSO)
- PAM

Rexec protocol

By using the rexec protocol the standard remote exec protocol, by default listening on port 512 is used. This protocol is a non-secure protocol.

Baan login protocol

The baan login protocol (Blogin) is a secure alternative for the rexec protocol. It requires starting as a separate daemon for UNIX (blogin6.2), or a service (rexecd.exe) on Windows (ES Logic Service).

The Blogin allows tracking of login attempts. For Windows this is registered in the EventViewer, for UNIX in log.blogin.

```
blogin6.2 -p 7150 -ssl security/ssl.properties
```

The option –p <port number> specifies the port number used by the Blogin daemon. When this option is omitted, the default port number is 7150.

The option –ssl <ssl properties file> specifies the ssl properties file to be used, relative to the BSE directory of the Baan Login daemon. When this option is omitted, the default ssl properties file is:

```
security/ssl.properties
```

Prefix the file name with a @-sign so it will not be interpreted relative to the BSE directory of the Baan Login daemon. For example use –ssl @local_file or –ssl @/etc/absolute_path. To start this daemon for problem tracing use:

```
>blogin6.2 –p 7150 –ssl security/ssl.properties –d > ${BSE}/log/blogin.
log 2>&1
```

The trace output will be sent to the file ${BSE}/log/blogin.log.

SSPI

For Windows: Solution 22893740 describes how auditing can be turned on to register logon of users.

SSO through Federation Services

A Single Sign On (SSO) solution removes authentication from the applicative code, and offers a globally secure software environment for users to provide their credentials once to access multiple applications.

Single Sign On by using Federation Services can be activated by these Enterprise Server sessions:

- SSO Parameters (ttams0100m000) session
- User Data (ttams1100s000) session

After running the session Convert changes to runtime DD (ttams2200m000) the following new files will appear in $BSE/lib/user :

- $BSE/lib/user/sso/s<SSO_USER>
- $BSE/lib/user/sso/<baanuser>

For more specific information, see the online help of these Enterprise Server sessions or the *Infor Enterprise Server - Single Sign On User Guide (U9559)*.

Other files on disk for SSO are:

- $BSE/lib/sso_config
  - UNIX
    - $BSE/security/sso_permissions.xml

    - $BSE/security/ssl.properties
  - Windows
    - $GLOBAL/security/sso_permissions.xml
    - $GLOBAL/security/ssl.properties

- **sso_config file**

  This file is configured through the SSO parameters session (ttams0100m000)

  This file contains the following SSO configuration parameters:

  - sso_location:<URL to CAS, which is the SSO server Infor uses> Only applicable for SSO through Infor Security. Note: Infor Security will be deprecated in the future by Infor.
  - generic_user:<Windows only - the name of the generic OS user name, used to start the Bshell>
  - gu_passwd:<Windows only - the crypted password for the generic OS user>

  This file describes which SSO user can impersonate a specific OS user. Protection is needed in case end-users try to change the configuration in the User Data in such a way that they can start a Bshell as for example OS user root.

- **sso_permissions.xml file**

  This file is maintained manually.

  A sample sso_permissions file will be placed in one of the following directories:

  - $BSE/security (on UNIX)
  - $GLOBAL\security (on Windows)

- **ssl.properties file**

  This file is maintained manually

  The property file ssl.properties contains the pathname and password of the keystore file used for the SSL communication with Web UI.

  An example of the ssl.properties file:

```
keystore: c:\Infor\ERPLN\commonx64\security\nlbaltoolsdev.p12
password: changeit
```

  A default keystore filename is:

```
c:\Infor\ERPLN\commonx64\security\keystore.p12.
```

  This default is used when the keystore line is omitted from the ssl.properties file. An alternative location for the ssl.properties file can be given via the following parameter when starting blogind:

```
-ssl <filename>
```

PAM

Pluggable authentication modules (PAMs) allow a UNIX computer to support multiple authentication technologies. Password Synchronization uses PAMs to provide UNIX-to-Windows password synchronization. The Password Synchronization PAM module (**pam_sso**) must be installed on each UNIX host where users can change their passwords.

The PAM library is a generalized API for authentication-related services which allows a system administrator to add new authentication methods simply by installing new PAM modules, and to modify authentication policies by editing configuration files.

If PAM is configured and enabled on the UNIX system, these protocols will use the PAM features that are available on the system:

- blogind
- rexec
- Password Aging

The authentication of users logging in through LN is handled using the normal system interfaces and PAM is used to handle the authentication. In case of the Password Aging the PAM libraries are used.

For more specific information about PAM refer to the Unix Manual Pages (man 5 pam.conf).

From LN perspective the use of PAM is transparent. No installation or configuration is needed on the LN side.

Environment variables for PAM

These environment variables can be used to trace PAM issues:

- PAM_SET_DEBUG:

  When set to “1”, trace messages are sent to stderr.

- PAM_TEST_LOG:

  If this environment variable is set, trace messages are sent to the file test.log in the directory/tmp.

These environment variables are available for badmin6.2:

- BADMIN_USE_PLAIN_PASSWORD:

  If this variable is set, a plain text password can be specified on the command line. This is useful for detecting badmin6.2 problems using the command line. This works for the options –chkpasswd and –chgpwd.

- CHECK_PASSWORD_DEBUG=2

  Test whether the password is changeable.

- CHECK_PASSWORD_DEBUG=3

  Test the password warning feature.

- CHECK_PASSWORD_DEBUG=4

Test the password expire feature. The warning in time is 7 days.

Using PAM has a limitation. The PAM library interface can not determine the exact number of days when a password will expire. It can only indicate that a password is about to expire.

You can place the resource named **pwd_default_warn** in the directory:

```
$BSE/lib/defaults/all
```

Set the warning indication to a certain value to let the resource supply a number of days. The default is 3 days.

Inter Process Communication

3

General

Enterprise Server consists of several processes with each their own specific task. If a process requires the services of another process, Enterprise Server can start this type of process and send requests.

This chapter describes how communication between these processes is set up, both local and remote communication are described. Remote communication refers to communication processes that do not reside on the same host.

Remote communication is required in a multi-application server setup. Processes on an application server initiate file server processes on the remote master application server to enable the processes to retrieve the required objects and configuration files.

InterProcessCommunication table

The ipc_boot process uses the ipc_info table as reference table for the communication. Any process that requires the service of another process will search the service based on the server name in the ipc_info file. The executable specified will be started and communication set up according to the specified protocol.

Note: The ipc_info table does not indicate if the requested server must be started on the local or a remote system. The requesting process must identify that system.

Example ipc_boot file

Server name	Protocol (s/p/d)	Executable server
oracle8	s	${BSE}/bin/ora8_srv6.2
informix	s	/usr1/bse/bin/inf_srv6.2
bshell	s	${BSE}/bin/bshell6.2

Server name	Protocol (s/p/d)	Executable server
fs6.2	p	${BSE_BIN}/fs6.2
db2v5	d	db2v5_srv6.2
db2v5	s	${BSE_BIN}/db2v5_srv6.2

- Server name: Used as identifier by the calling program.
- Protocol: Identifies the communication protocol
  - Socket (s)
  - Pipes (p)
  - Direct connection (d)
- Executable server: Identifies the executable to communicate with.

The ipc_info file resides in ${BSE}/lib.

Direct connections are only allowed for database drivers. A direct connection loads the database driver shared library (UNIX) or DLL (Windows) in the database client binary, e.g. the bshell. This is also known as ‘combo-driver’.

Direct connections are limited to one database server. By default, a database driver can handle up to 64 sessions simultaneously. If the number of sessions for a database server exceeds this limit, a second database server is started as a separate process; for that reason, a direct driver entry is always followed by a standalone-entry with the ‘p’ or ‘s’ protocol type (see the previous table).

Note: To overrule the default number of sessions per database driver, use the bdb_max_sessions driver resource.

It is not allowed to specify a path for a direct connection. A shared library/DLL associated with a direct connection is picked up from ${BSE}/shlib. It is required that the product version & PA-number of the client binary (e.g. bshell) exactly matches the product version & PA-number of the (database) shared library/DLL from ${BSE}/shlib. Mixing debug and release builds is not possible.

Communication protocols

Depending on the hardware configuration, you can choose the following protocols for the communication between the client and the server:

Socket protocol

You can use the socket protocol for both local and remote communication. All sockets are stream sockets (TCP).

Pipes protocol

You can only perform communication by means of unnamed pipes with local communication. Two pipes are always required, because pipes are unidirectional.

Direct connection

This is an specific implementation for shared library-based communication, used for database drivers only.

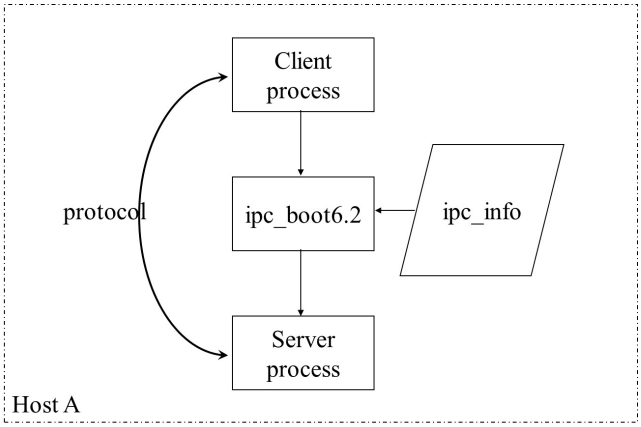
Inter process communication

This chapter describes how to set up process communication in Enterprise Server if an existing process starts another process. The initiating process is called the client process and the requested process is called the server process. This section describes both the set up for both local and remote communication.

The client process identifies if the server process must run locally or remotely.

Local processes

If you must start a server process on the same host, the client process starts an ipc_boot process, which provides the server name that must be started. Ipc_boot searches the ipc_info file ($BSE/lib/ipc_info) for the “server name” and pickup the related transport protocol and path of the executable. The ipc_boot process overwrites itself with the identified executable and the client server communication is arranged, leaving 2 processes in place.



Remote processes

The relevant Enterprise Server executables, such as bshell, database drivers, and audit server are equipped with an internal network layer. This network layer handles both local and remote communication.

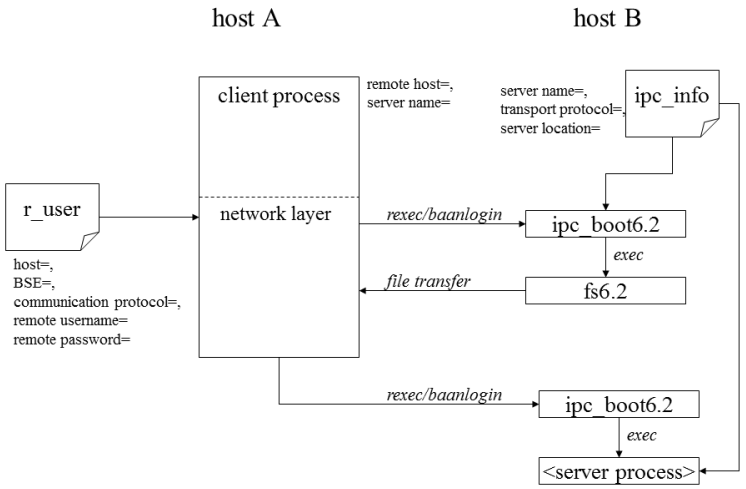
The client process requests the network layer to initiate a process. Based on the given parameters, the network layer identifies that a remote process is required.

Remote configuration information might be required before the user can start the expected server.

For example, if a remote database driver must be started, the remote tabledef ($BSE/lib/tabledef6.2) is first required to identify what type of db-driver the user must start. In that case, the file server must be running on the remote host.

To start the file server to retrieve remote files

The network layer opens the user's remote user file $BSE/lib/user/r_<user> and retrieves from that file the required information to start ipc_boot remotely. By default, you can start the remote ipc_boot by means of rexec. However, you can use baanlogin instead.



On the remote host, the ipc_boot process retrieves from the ipc_info file ($BSE/lib/ipc_info) the path of the fs executable, as well as the transport protocol. That information overwrites the ipc_boot process with an fs process. The fs is the Enterprise Server internal file server that enables remote file access.

The client process can now retrieve the required configuration file from the remote fs server to identify which server name you must start.

To start the remote server process

The network layer of the client process starts ipc_boot on the remote host by means of rexec or Baanlogin. Next, the network layer informs the ipc_boot process which server name the user is expected to start. The ipc_boot process searches the requested service name in the ipc_info file and extracts

the related transport protocol and path of the executable execute if the requested server process overwrites its own process space.

After the initialization, the following processes will be running: the client process on the local host and the server process on the remote host.

Distributed application servers

To distribute load over multiple application servers, Enterprise Server supports the AS and MAS concept, having one Master Application Server and one or more Application Servers. The *Installation and Configuration Guide for the Application Server (U8392 US)* describes the installation and configuration of the application server.

The purpose of the AS/MAS concept is to support an environment of distributed application servers, while limiting the impact on maintenance as much as possible. In principle, only the MAS must be updated in case of software updates. Application servers automatically retrieve the required software components and configuration files from the MAS during runtime. Only a portingset update must be installed on all application servers.

An application server has all the required processes running locally, such as the bshell, database drivers, printer drivers, and service processes (such as shared memory manager and shared memory timer). Application and configuration data (such as database configuration information, application objects, and user data) are retrieved from the master application server.

The application server is aware of the master application server through the BSEREM variable. You set this variable during the installation of the application server and it is stored in the $BSE/lib/bse_vars file. The BSEREM identifies the MAS host and BSE path.

If an Enterprise Server-related process on the application server requires configuration files, objects, or other file types, the network layer of that process first searches for these files in the regular way on the local system. If this file is not found locally, the network layer checks if a BSEREM variable is set, and, if so, uses that value to search for the file on the indicated remote host (the MAS). This requires the fs process to be running on the MAS. If the fs process is not running yet, the application server process starts the process as described in "To start the file server to retrieve remote files" on page 46.

Audit servers

In a distributed application server environment, all audit servers must run on the same application server. For information on how to start the audit server, refer to "Remote processes" on page 45.

The `$BSE/lib/audit_srv` file identifies the audit server host.

Remote user file

You can use the remote user file to identify required information to connect to a remote host.

A remote-user file can have multiple entries, identifying multiple remote Enterprise Server environments.

For each entry, the following information fields are available:

- **host**
  Remote host name.
- **[communication protocol [:port number] !]**
  optional
  Communication protocol: rexec | baan
  Port number: related port number of chosen protocol as set on the remote host.
- **BSE**
  BSE path of the remote environment.
- **[userid]**
  optional
  OS account to use for login at the remote host. If not set, the local account is used.
- **password**
  The password related to the previously mentioned or local account, stored in encrypted form.
- **[OS authorization or remote user]**
  yes | no
- **[Enterprise Server authorization of remote user]**
  yes | no

The remote-user file information is stored in the Systems (ttaad0550m000) session.

If the rexec protocol is selected, the value added in front of the BSE-path is “**rexec!**” If the BaanLogin protocol is selected, the value added will be “baan!” For example:

```
red baan!/usr3/Baan 3F'L|9)c)Z0Rw4|F>=LW+KtpMG_^T1]3
```

The default protocol here is also “rexec.” Therefore:

```
blue /usr3/BAAN40 3F'L|9)c)Z0Rw4|F>=LW+KtpMG_^T1]3
```

is exactly the same as:

```
blue exec!/usr3/BAAN40 3F'L|9)c)Z0Rw4|F>=LW+KtpMG_^T1]3
```

To use a port number that differs from the default, you must add the port number between the protocol and the exclamation point with a colon as the separator, for example:

```
red baan:8000!/usr3/Baan3F'L|9)c)Z0Rw4|F>=LW+KtpMG_^T1]3
```

The additional fields protocol and port are not yet supported from the session. Therefore, you must add these files manually to the r_users file, if required.

Native Language Support

4

General

Note: This chapter is only valid for installations on Unix, and mostly related to Infor Baan IVc running under the Convergence Porting Set, because Windows uses BWprint for printing.

Native Language Support controls the communication between the character set of the printer devices and that of bshell.

A bshell can use several character sets, including the eight-bit character set of ISO 8859-1. Many printers do not support this character set, therefore, Native Language Support techniques are implemented.

The character set of the terminal is uniform. However, various country-dependent keyboards can be used with the same terminal. Not all the characters of the terminal character set can be found on the keyboard. The remaining characters must be composed. The process to compose characters is described in "Composed characters" on page 51.

An output conversion table is used to convert the character set of the bshell to that of the printer. In this table, a character from ISO 8859-1 is converted to the corresponding value from the terminal character set. For a description of the conversion tables, refer to "Conversion tables" on page 52.

To maintain the conversion tables, you can use the NLS editor. This editor is described in "NLS editor" on page 53.

Conversion capabilities between the user interface, Web UI and WorkTop, and the bshell are not required because the prerequisite is that both be configured to run with the same character set.

WorkTop works in the character set configured in the Regional Options of the Windows system.

The bshell works in the character set defined in the User Data Template (ttams1110m000) session.

Composed characters

Characters that are not found on a keyboard must be composed by a unique combination of two or more existing characters. Before you enter the combination, you must press a compose key. The

compose key is defined in the terminal information file of the terminal. For more information, refer to "NLS-related files" on page 55.

The compose key can be linked to a special key on the keyboard, for example, a function key. To compose characters, you can also use the Compose character key, if present on the keyboard. This key has the same function as the compose key.

The manufacturer of the terminal has defined the composed characters. The user documentation of the terminal contains a list of these composed characters.

Conversion tables

You need conversion to make the character sets of bshell and the printer compatible: output conversion.

Output conversion table

The output conversion table converts the characters of the ISO character set to their corresponding values in the printer character set.

In this way, a character is printed as entered.

To record or modify an output conversion table, you can use the NLS editor, as described in "NLS editor" on page 53.

An example of an output conversion table is provided in the "Prestige Character Set (MT910 HP) printer output table" on page 60.

Example of a conversion

```
Printer type: Mannesmann mt910
Required character: ÿ = ISO8859-1 code 255
```

Assume that the bshell is running in character set ISO8859-1, the character ÿ is represented with code 255.

To print the entered (required) character on a Mannesmann mt910 printer, the printer output table must contain the following line:

```
\255 y\b\168
```

For this reason, the ÿ character is printed as follows:

```
[y] <Backspace>["]
y \b \168
```

NLS editor

Use the NLS editor to maintain input and output conversion tables. To start the editor, start **nlsedit6.2**. To maintain printer output tables, use the **–p** option, followed by the printer name.

After you start the editor, the upper part of the ISO character set (160-255) appears. If a character cannot be displayed on your screen, the reason can be because the character is not found in the terminal's character set.

To display feasible options, you can type **?**, or type **Q** to exit the editor.

Note: Standard delivered Enterprise Server communications no longer require input conversion; this conversion was used in the past for terminal conversion. Because the input conversion is required for other devices, the conversion is still supported.

Initial screen options

Feasible initial screen options include the following:

- **–d/h/o:**
  Set decimal/hex/octal display mode
- **–t:**
  Edit nls_in and nls_out tables
- **–r:**
  Toggle display of characters
- **–s:**
  Toggle character set

With the options **–D**, **–H**, and **–O**, you can change the display mode of the character values into decimal, hexadecimal, or octal.

You can use the **–T** option to edit the conversion tables. A second screen appears.

The initial screen shows the characters that the terminal displays as defaults. With the **–R** option, you can set this mode on or off.

The default characters that appear on the initial screen are characters of the upper part (160-255) of the ISO character set. With the **–S** option, you can switch between the lower part (32-127) and the upper part of this character set.

To display the initial screen options, you can type **?**. The window that appears also shows your terminal setup, NLS table names, and shell variable BSE.

Maintenance of NLS conversion tables

If you enter the **–T** option on the initial screen, you can maintain the input and output conversion tables. Five columns appear, as illustrated in the following example:

Input	int	s	#	Output
A`	192	À	0	\192
A'	193	Á	1	\193
A\^	194	Â	0	\194

The left column, **input**, is only required to create input conversion tables that are no longer required with Enterprise Server, but maintained for backward compatibility.

The second column, **int**, displays the values of the characters in the ISO character set.

The third column, **s**, shows the characters as they are represented after output conversion.

The fourth column, **#**, contains the character set number. Character sets are defined at the terminal or printer. The possible values are **0-9**, and the default is **0**.

The last column, **output**, contains the output sequence of the characters. You can type the output sequence in decimal, hexadecimal, or octal code.

The bottom of the screen shows the display mode in which the values of the characters appear. The last line indicates the marked character. The numeric representations and description are also included.

Feasible options to maintain conversion tables include the following:

- **–D/H/O:**
  Set decimal/hexadecimal/octal display mode
- **–N:**
  Toggle numeric/alpha mode compose sequence
- **–Down Arrow:**
  Next line
- **–Up Arrow:**
  Previous line
- **–Ctrl+N:**
  Next page

- **–Ctrl+P:**
  Previous page
- **–K:**
  Single key
- **–C:**
  Enter compose sequence
- **–S:**
  Enter output sequence
- **–W:**
  Write nls_in and nls_out file
- **–+/-:**
  Increment/decrement character set number

With the options **–D**, **–H**, and **–O**, you can change the display mode of the character values into decimal, hexadecimal, or octal.

The **–N** option converts alphanumeric characters from the input and output table to their numeric values, and vice versa.

To move the bar to the following or previous line, use the key down arrow or up arrow.

If you press Ctrl+N or Ctrl+P, you can move the bar 15 lines forward or backward.

To enter the input sequence of a single key, press **–K**. The editor asks you to press the required key. That character is included directly in the table. You do not have to press enter after you enter the character.

After you press **–C**, you can enter the compose sequence. You can enter up to 35 characters.

To type the output sequence, enter **–S**. You can enter up to 35 characters.

After any modification, to store the input and output tables, you can enter **–W**.

NLS-related files

To work with NLS, you must have the following files in your BSE environment:

- Terminal information file$BSE/lib/terminf/...
  Only required for input conversion.
- Printer information file$BSE/lib/printinf/...
- Input conversion table$BSE/lib/nlsinf/...
- Output conversion table$BSE/lib/nlsinf/...

You must also have the appropriate terminal setup, shell variable TERM, and font. For some additional features, you can use sort tables and shift tables.

Terminal information file

This section is relevant mostly for Infor Baan IVc. The terminal information files have been placed in subdirectories of the `$BSE/lib/terminf` directory. For example, terminal information files that start with the letter V are placed in subdirectory `v`.

You must add the following two lines to each terminal information file to indicate the conversion tables the terminal uses:

```
nls_in=<terminal type>.in
nls_out=<terminal type>.out
```

For example, for a vt200 terminal, the terminal information file vt200 can contain the following lines:

```
nls_in=vt200.in
nls_out=vt200.out
```

The compose key code is also defined in the terminal information file:

```
kcompose=<compose key code>
```

The code can be anything that does not conflict with other special keys. For example, suppose you want to define the compose key as Esc+C. Add the following line to the terminal information file:

```
kcompose=\ec
```

Printer information file

The printer information files have been placed in subdirectories of the $BSE/lib/printinf directory. For example, printer information files that start with the letter M are placed in subdirectory M.

You must add the following line to each printer information file to indicate the output conversion table the printer uses:

```
nls_out=<printer type>.out
```

For example, for an mt910 printer, the printer information file mt910 must contain the following line:

```
nls_out=mt910.out
```

Input and output conversion table

The input and output conversion tables are stored in the $BSE/lib/nlsinf directory. The possible table names in this directory include the following:

```
<terminal type>.in (input conversion terminal)
<terminal type>.out (output conversion terminal)
<printer type>.out (output conversion printer)
```

For each terminal, one input and one output conversion table is available. For each printer, only an output conversion table is available.

The format of every line in a conversion table is the following:

```
<string><tabs or spaces><string>
```

A string can consist of one or more of the following:

- ASCII codes: A-Z, a-z, and 0-9...
- Octal codes: \01, \012, \012...
- Decimal codes: \1, \12, \12...
- Hexadecimal codes: \0x1, 0x12...
- Control codes: ^A, ^B, ^C...
- Escape codes: \E, \e, \s, \n, \r, \f, \v, \b
- Special codes: \\, \^

For a list of control and escape codes, refer to "Overview of ESC/CTRL codes" on page 64. The first column contains the numeric representation of the ISO 8859-1 character set.

In an input table, the second column contains the compose sequence of the input characters. In an output table, the second column contains the numeric representation of the output characters. The columns are separated by spaces or tabs.

Sort tables

You can use a sort that differs from the default sort method. To perform this type of sort, you can use a sort table to specify the weight of characters.

The file with the sort table has the default name:

```
$BSE/lib/nlsinf/sort.tab
```

To make your own sort table, you can fill the SORT_TABLE environment variable with the name of your own sorting file. A sort table has two columns. The first column contains the character, while the second column contains the weight of that character.

For example, to sort the B before the A, create a sort table with the following contents:

```
\65 \66 or A B
\66 \65 B A
```

You can use escape and control codes, such as \012, \e, ^A. For more information, refer to "Input and output conversion table" on page 57.

Shift tables

You can use shift tables to specify lowercase and uppercase characters that belong together.

The file with the shift table has the following default name:

```
$BSE/lib/nlsinf/shift.tab
```

To make your own shift table, you can fill the SH_TABLE environment variable with the name of the file with your own shift table. A shift table has two columns. The first column contains the uppercase character, and the second column contains the lowercase character.

For example, to specify the lowercase and uppercase character C with cedilla, create a shift table with the following contents:

```
\199 \231
```

ISO 8859-1 character set (ASCII 0-127)

0	16	32 SP	48 0	64 @	80 P	96 `	112 p
1	17	33 !	49 1	65 A	81 Q	97 a	113 q
2	18	34	50	66	82	98	114

		“	2	B	R	b	r
3	19	35 #	51 3	67 C	83 S	99 c	115 s
4	20	36 $	52 4	68 D	84 T	100 d	116 t
5	21	37 %	53 5	69 E	85 U	101 e	117 u
6	22	38 &	54 6	70 F	86 V	102 f	118 v
7	23	39 ‘	55 7	71 G	87 W	103 g	119 w
8	24	40 (	56 8	72 H	88 X	104 h	120 x
9	25	41)	57 9	73 I	89 Y	105 I	121 y
10	26	42 *	58 :	74 J	90 Z	106 j	122 z
11	27	43 +	59 ;	75 K	91 [	107 k	123 {
12	28	44 ‘	60 <	76 L	92 \	108 l	124 |
13	29	45 -	61 =	77 M	93]	109 m	125 }
14	30	46 .	62 >	78 N	94 ^	110 n	126 -
15	31	47 /	63 ?	79 O	95 _	111 o	127
128	144	160 NBSP	176 °	192 À	208 Đ	224 à	240 ð
129	145	161 ¡	177 ±	193 Á	209 Ñ	225 á	241 ñ
130	146	162 ¢	178 ²	194 Â	210 Ò	226 â	242 ò
131	147	163 £	179 ³	195 Ã	211 Ó	227 ã	243 ó

132	148	164 ¤	180 '	196 Ä	212 Ô	228 ä	244 ô
133	149	165 ¥	181 µ	197 Å	213 Õ	229 å	245 õ
134	150	166 ¦	182 ¶	198 Æ	214 Ö	230 æ	246 ö
135	151	167 §	183 ·	199 Ç	215 ×	231 ç	247 ÷
136	152	168 ¨	184 ,	200 È	216 Ø	232 è	248 ø
137	153	169 ©	185 ¹	201 É	217 Ù	233 é	249 ù
138	154	170 ª	186 °	202 Ê	218 Ú	234 ê	250 ú
139	155	171 «	187 »	203 Ë	219 Û	235 ë	251 û
140	156	172 ¬	188 ¼	204 Ì	220 Ü	236 ì	252 ü
141	157	173 SHY	189 ½	205 Í	221 Ý	237 í	253 ý
142	158	174 ®	190 ¾	206 Î	222 Þ	238 î	254 þ
143	159	175 ¯	191 ¿	207 Ï	223 ß	239 ï	255 ÿ

Prestige Character Set (MT910 HP) printer output table

\160	\s
\161	\184
\162	c\b|
\163	\175
\164	\186

\165	\188
\166	|
\167	\189
\168	\171
\169	\s
\170	a\b_
\171	<<
\172	\176
\173	-
\174	\s
\175	\176
\176	\179
\177	\254
\178	\s
\179	\s
\180	\168
\181	u
\182	\s
\183	.
\184	,
\185	\s
\186	\250
\187	>>
\188	\247
\189	\248
\190	\s
\191	\185
\192	\161
\193	\224
\194	\162
\195	\225
\196	\216

\197	\208
\198	\211
\199	\180
\200	\163
\201	\220
\202	\164
\203	\165
\204	\230
\205	\229
\206	\166
\207	\167
\208	\227
\209	\182
\210	\232
\211	\231
\212	\223
\213	\233
\214	\218
\215	x
\216	\210
\217	\173
\218	\237
\219	\174
\220	\219
\221	\89
\222	\240
\223	\222
\224	\200
\225	\196
\226	\192
\227	\226
\228	\204

\229	\212
\230	\215
\231	\181
\232	\201
\233	\197
\234	\193
\235	\205
\236	\217
\237	\213
\238	\209
\239	\221
\240	\228
\241	\183
\242	\202
\243	\198
\244	\194
\245	\234
\246	\206
\247	/
\248	\214
\249	\203
\250	\199
\251	\195
\252	\207
\253	y\b\168
\254	\241
\255	\239

Overview of ESC/CTRL codes

ASCII mnemonic	Dec code	Hec code	Oct code	ESC code	CTRL code
SOH	0001	0x01	0001		^A
STX	0002	0x02	0002		^B
ETX	0003	0x03	0003		^C
EOT	0004	0x04	0004		^D
ENQ	0005	0x05	0005		^E
ACK	0006	0x06	0006		^F
BEL	0007	0x07	0007		^G
BS	0008	0x08	0010	\b	^H
HT	0009	0x09	0011		^I
LF	0010	0x0A	0012	\n	^J
VT	0011	0x0B	0013	\v	^K
FF	0012	0x0C	0014	\f	^L
CR	0013	0x0D	0015	\r	^M
SO	0014	0x0E	0016		^N
SI	0015	0x0F	0017		^O
DLE	0016	0x10	0020		^P
DC1 (Xon)	0017	0x11	0021		^Q
DC2	0018	0x12	0022		^R
DC3 (Xoff)	0019	0x13	0023		^S
DC4	0020	0x14	0024		^T
NAK	0021	0x15	0025		^U
SYN	0022	0x16	0026		^V
ETB	0023	0x17	0027		^W
CAN	0024	0x18	0030		^X
EM	0025	0x19	0031		^Y
SUB	0026	0x1A	0032		^Z
ESC	0027	0x1B	0033	\e	^[
FS	0028	0x1C	0034		^\
GS	0029	0x1D	0035		^]

ASCII mnemonic	Dec code	Hec code	Oct code	ESC code	CTRL code
RS	0030	0x1E	0036		^^
SPACE	0032	0x20	0040	\s	

Directory Structure

5

A direct relation exists between Enterprise Server Administration and the UNIX or Windows directory structure on the system. A large amount of data entered in Enterprise Server is stored in operating system files. The Infor LN administrator must know how the data is structured, and where the data is located.

This chapter includes:

- An overview of the directory structure
- The software subdirectories
- The dict directory
- The test directory

An overview of the directory structure

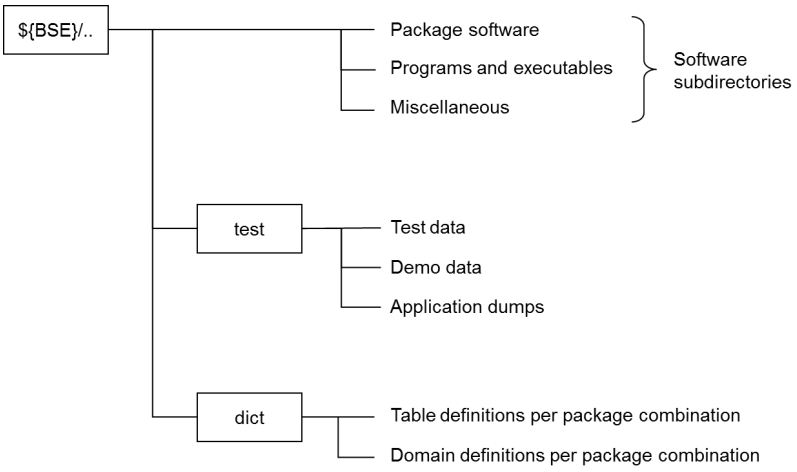
LN uses a Virtual Machine (VM) to make the software independent of the operating system. LN can, therefore, run on a UNIX operating system or a Windows operating system. For this reason, the notation of the directories in this chapter is subject to your operating system: UNIX operating systems use slashes (/) and Windows uses back slashes (\). For example, the directory notation on a UNIX system is ${BSE}/application/…and on a Windows operating system the notion for the same directory is ${BSE}\application\….

Note: Unless otherwise noted, the directories described in this chapter are used in both operating systems.

The LN application is stored in the ${BSE} directory. The main subdirectories in the ${BSE} are the following:

- The subdirectories that contain the software.
- The dict subdirectory, which contains the subdirectories for the table and domain definitions of all packages and package combinations except for Enterprise Server.
- The test subdirectory, which contains the test data and demonstration data, and dumps for the applications.

The following figure shows LN's main directories:



The location of the software environment on your operating system is defined by the LN software ${BSE} environment variable. You must define the ${BSE} variable for each LN user.

Software subdirectories

The LN software environment is stored in the ${BSE} directory. The software subdirectories in the ${BSE} directory are categorized as follows:

- Package software
- Executables and programs
- Miscellaneous

The following table shows the software subdirectories in the categories.

Category	Subdirectories
Package software	application tools
Executables and programs	bin etc shlib java
Miscellaneous	include6.2 log tmp secu api audit

Category	Subdirectories
	lib

Package software

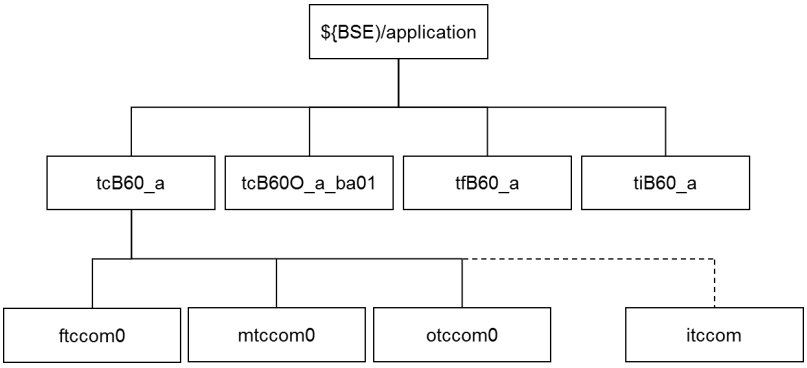
The package software category contains the software of the LN application and the Tools (tt) package.

${BSE}/application directory

The ${BSE}/application directory stores the run-time versions of the software components, such as forms, menus, report scripts, objects, and so on. For each package VRC, a separate subdirectory exists in the application directory, for example, ${BSE}/application/tcB60_ba01.

A package VRC defines a specific version of a package. For each package more than one version can exist, for example, the standard version and a customized version. The software of standard and customized versions is stored in dedicated directories identified by the package VRC.

The following figure shows an example of the ${BSE}/application directory:



The ${BSE}/application directory is the default directory in which the LN application's software components are stored. You can define another directory for the software components with the Directory of Software Components (ttadv1115m000) session. You can perform this function, for example, if you run out of disk space on your current file system.

${BSE}/tools directory

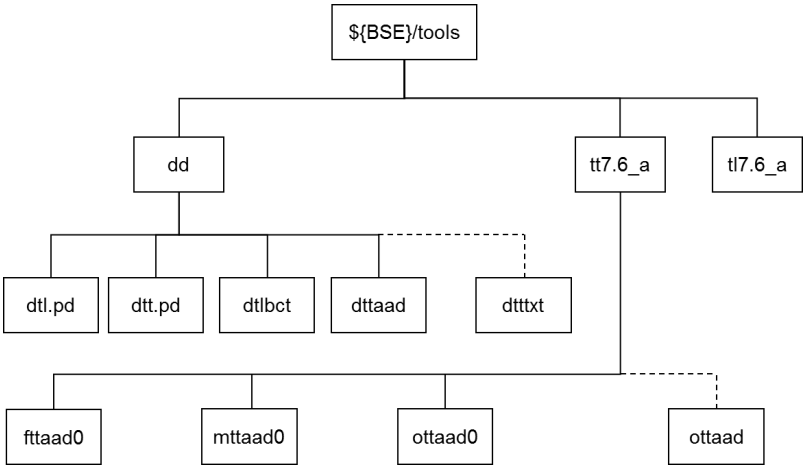
The Enterprise Server consists of the Tools (tt) and the Tools Add-on (tl) package. These two packages are technically different from the application packages. Therefore, the Enterprise Server software and the software's table and domain definitions are stored separately in the ${BSE}/tools directory.

The ${BSE}/tools directory contains:

- A dd (data dictionary) directory that contains the table and domain definitions of the packages that belong to Enterprise Server.

- A separate package VRC directory for each package that belongs to Enterprise Server, for example: tt7.6_a and tl7.6_a.

The following figure shows an example of the Tools directory:



The dd directory can be divided into two types of subdirectories:

- The d<*package*>.pd directories, which contain the domain definitions of the packages belonging to Enterprise Server, for example: dtt.pd and dtl.pd.
- The d<*package*><*module*> directories, which contain the table definitions for each module in the packages belonging to Enterprise Server, for example: dttaad, dttadv, dtlbct, and dtlcom.

The package VRC directories contain the software components of the packages that belong to Enterprise Server. Several subdirectories exist in which the various modules and software components are stored, for example:

- The tt7.6_a/ottaad subdirectory contains the program objects of the Application Administration (AAD) module of the Tools (tt) package.
- The tl7.6_a/ftltsm0 subdirectory contains the forms of the Table Sharing Modeler (TSM) module of the Tools Add-on (tl) package.

The package VRC directories in the ${BSE}/application and ${BSE}/tools directories have subdirectories for each type of software component, such as forms, reports, and so on. The following table describes how each of these subdirectories is coded:

Subdirectory	Component	Example
f<package><module>0	Forms	ftccom0
m<package><module>0	Menus	mtccom0
o<package><module>0	Report objects	otccom0
r<package><module>	Report scripts	rtccom
o<package><module>	Program objects	otccom
p<package><module>	Program scripts	ptccom
i<package><module>	Include files	itccom

Subdirectory	Component	Example
b<package><module>	Additional files: .GIF files for the graphical user interface and XML schema files for the arguments of business object methods.	btccom

The examples in this table represent components of LN Common (tc), which is an application package. The notation of the subdirectories in LN Tools (tt) is identical.

Note: All components in the subdirectories are language-independent: each package VRC directory contains only one set of components that is used for all languages.

For example, if a user starts a session, the corresponding label descriptions, for example, the field names on the session's form, are read at runtime and appear in the user's language.

The label descriptions are stored in the ${BSE}/lib/labels directory for all package VRCs and for all installed languages. Refer to "Language translation support" in the Enterprise Server Web Help for details.

Executables and programs

The programs and executables that LN uses are located in the ${BSE}/bin directory.

${BSE}/bin directory

The ${BSE}/bin directory contains the programs of the operating system that LN uses. Examples of these programs include the following:

- The ASCII display server (ba), which is still used, for example, for job management on a UNIX system, but not on a Windows platform
- The shared memory manager (shmmanager)
- The Virtual Machine (VM) (bshell)
- Several database drivers

Note: On a Microsoft Windows operating system, the programs in the ${BSE}/bin directory have the .exe extension instead of the 6.2 extension.

${BSE}/etc directory

The ${BSE}/etc directory contains the start and stop scripts for the LN environment (UNIX only). Examples of these programs include:

- **rc.start**: Starts up the LN environment and starts the subscripts to initialize the shared memory and the printer daemon. The rc.start script is automatically started from the UNIX startup files upon system initiation.

- **rc.stop**: Shuts down the LN environment. Actions such as stopping and starting shared memory, printer daemon, and database are performed. You can start the rc.stop script if the system is shut down with the Shut Down program.
- **rc.startjob**: Starts a job
- **rc.startjobdm**: Starts the job daemon. The job daemon can place jobs in a queue, which starts the jobs automatically.
- **rc.stopjobdm**: Stops the job daemon

Note: The ${BSE}/etc directory only exists on a UNIX operating system. For more information on the etc directory, refer to the UNIX Installation Manuals.

Miscellaneous

The miscellaneous software that LN uses is stored in the following directories.

${BSE}/include6.2 directory

The ${BSE}/include6.2 directory is used for Infor Baan 5.0 and Infor LN support. This directory contains several files for standard functions and definitions that the Virtual Machine (VM) uses during the compilation of program scripts and report scripts.

${BSE}/include6.1 directory

The ${BSE}/include6.1 directory is used for Infor Baan IVc support. This directory contains several files for standard functions and definitions that the Virtual Machine (VM) uses during the compilation of program scripts and report scripts.

${BSE}/log directory

The ${BSE}/log directory contains error messages, as well as some additional information, which is stored in a log file. If an error occurs while the application is in use, an error message appears on the screen. Log files have the *log.<program name>* format and are used to determine the cause of an error message.

Note: The size of the log files in ${BSE}/log is adjustable. By default, the size of the log files is 512 KB. When a log file exceeds that size, the bshell automatically renames the file to *olg.<program name>* and starts writing a new log file from scratch.

To adjust the size of the log files, set the ‘log_size’ resource. The default value is: 512.

To set this resource, for example, add the following line to the $BSE/lib/defaults/all file:

```
log_size:1024
```

${BSE}/tmp directory

The ${BSE}/temp directory stores the application's temporary files. For example, LN enables you to store the temporary file for print requests in the ${BSE}/tmp directory to repeat print requests. If a print request must be repeated, the temporary file is retrieved and started. Most of the files are located in the temporary directory (tmp). Some of the files in this directory can be lock files and are used to prevent some processes from starting up twice.

${BSE}/secu directory

The ${BSE}/secu directory contains the files that are related to protected software. LN software can be protected against unauthorized use and the protected LN software can only be used after being validated and patched.

${BSE}/api directory

The ${BSE}/api directory contains some examples for customers.

${BSE}/audit directory

The ${BSE}/audit directory contains the audit information, such as table transactions. LN provides an audit management capability, which enables you to log actions on LN tables. The audit functionality must be configured, using the sessions of the Audit Management (AUD) module of the tt package. These sessions store the configuration data in the current session.

For each audited table, the following two types of audit files exist:

- Sequence files: End with a sequence number and contain the audit information
- Information files: Have an .INF extension and contain information about the sequence files that pertain to the same table

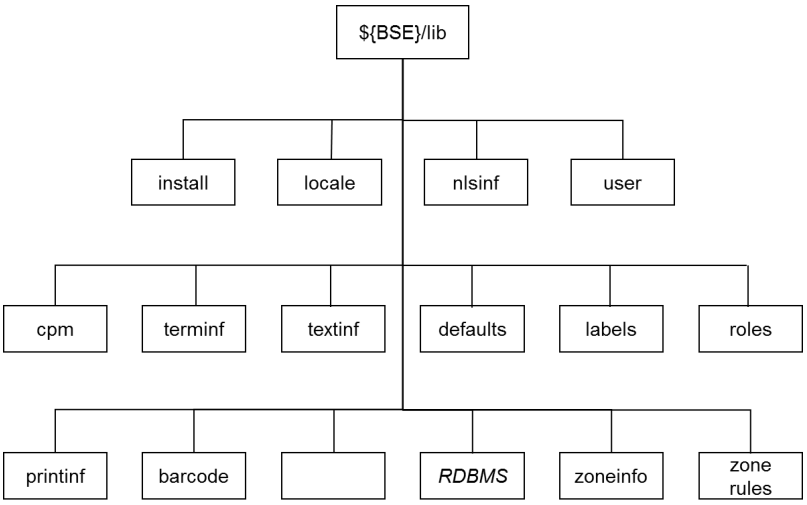
By default, audit-related data is assigned to the $BSE/audit directory. You can use the Audit Trail Paths (ttaud3136m000) session to change the directory that must contain the audit data.

For more information, refer to "Audit Management" on page 133.

${BSE}/lib directory

The ${BSE}/lib directory contains the subdirectories that contain the settings and files with the database driver information.

The following figure is an example of the structure in the ${BSE}/lib directory:



You can find the following subdirectories in the ${BSE}/lib directory:

- **barcode**: Contains the scripts that are required to print bar codes
- **cpm**: Cognos ini files.
- **defaults**: Contains files to set resources. For details, refer to "LN Environment Variables and Resources" on page 175.
- **install**: Contains the files, and not merely log files, that are used to install LN. You cannot remove or clean up this directory.
- **labels**: Contains the label resource files from which the label descriptions are read at runtime
- **locale**: Contains the LN superset locales. A locale contains information about character-set definitions, native language support, and the minimum and maximum values of the forms and database fields.
- **models**: Contains runtime files for Document Authorization, also known as Database Change Management (DBCM).
- **nlsinf**: Contains the files for various terminal types, which enables the Virtual Machine (VM) to support several languages and their specific characters
- **RDBMS**: Contains the programs for the database you use. For example, if you use an Oracle database, the notation of the RDBMS directory is ${BSE}lib/ora.
- **roles**: Contains the authorizations and database permissions of the LN users
- **printinf**: Contains the printer drivers with detailed printer definitions for various types of printers. You can use a driver each time an LN report is sent to a printer.
- **terminf**: Contains the terminal definition files (drivers) for various types of terminals
- **textinf**: Contains error messages for the display servers
- **user**: Contains the u<*User*> files for every LN user. These files store the basic user data, such as the Startup menu, the package combination, and the default company. Some other files are optional, for example, the remote user file r<*User*>.
- **zoneinfo**: Contains files to support dates/times in various time zones
- **zonerules**: Contains files to support dates/times in various time zones

The ${BSE}/lib directory also contains the following files:

- **auditdef6.2 / auditdef6.1**: Contains the directory paths where the sequence files are stored on the system. The auditdef6.1 is used for Infor Baan IVc. The auditdef6.2 is used for Infor Baan 5.0 and Infor LN.
- **audit_spec**: Specifies how the audit trail is organized and handles permission issues
- **audit_cols**: Defines which tables and fields are audited, and when the tables are audited
- **audit_hosts**: Defines on which system the audit servers run
- **compnr6.2 / compnr6.1**: Contains information on the mapping of logical and physical companies/tables. The compnr6.1 is used for Infor Baan IVc. The compnr6.2 is used for Infor Baan 5.0 and Infor LN.
- **Datecurr**: Contains the date format and currency format
- **dtopt.<*Language Code*>**: Contains information that the Virtual Machine uses to display the menus in the user interface
- **fd6.2.<*Package Combination*>**: Contains the location of the software components for a particular package combination
- **ipc_info**: Contains the technical information the display driver uses, and the database drivers
- **srdd_tab6.2 / srdd_tab6.1**: Contains the names of the software components that are loaded into shared memory during the start-up process. The srdd_tab6.1 is used for Infor Baan IVc. The srdd_tab6.2 is used for Infor Baan 5.0 and Infor LN.
- **tabledef6.2 / tabledef6.1**: Contains the database types that are used for the various tables. If one or more tables are located on a remote system, the name of the system also appears in this file. The tabledef6.1 is used for Infor Baan IVc. The tabledef6.2 is used for Infor Baan 5.0 and Infor LN.
- **tss***: These files are related to the Triton Super Set (TSS). TSS is a collection of character sets that you can use in Infor LN or Baan applications. TSS accommodates all multibyte character sets in a single package. TSS includes both ASCII character sets and multibyte character sets, such as Japanese, Chinese, and Hebrew.

Note: You must not edit the files under the ${BSE}/lib directory manually.

- Some files are delivered as standard files and must only be modified by Infor
- Some files can be regenerated through a conversion to the runtime data dictionary. For example, to change the contents of a user file in ${BSE}/lib/user, perform the desired changes in the User Data (ttams1100s000) session and subsequently convert the changes to runtime through the Convert Changes to Runtime DD (ttams2200m000) session.

dict directory

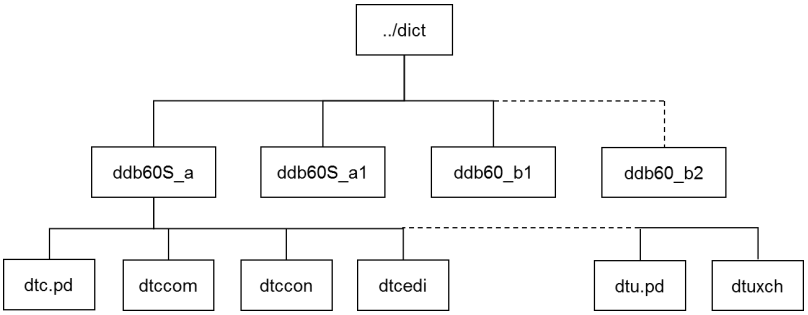
The dict directory contains the subdirectories for the table definitions and domain definitions of each package combination. The table definitions describe the structure of the LN tables. The names and domains of the table fields are defined in the table definitions.

A table field is always linked to a domain. A domain definition defines the data type for a table field and contains additional information, for example, about adjustment and conversion to uppercase or lowercase. A domain can be linked to more than one table field.

A package combination is a combination of several packages. The packages in the package combination do not have to be identical. For example, a package combination can have two packages with the

standard version of the software and three packages with customized versions of the software. However, a package combination can contain only one version of each package.

The following figure shows an example of the structure in the dict directory:



The dict/ddb60S_a directory in the previous figure contains the following subdirectory types:

- The d<*package*>.pd directory, which contains the domain definitions of the package d<package>.pd. For example, dtc.pd contains the domain definitions of LN Common (tc).
- The d<*package*><*module*>, which contains the table definitions of the module in the package. For example, dtccom contains the table definitions of the Common Data (com) module in LN Common (tc).

Test directory

The test directory contains the test data and demonstration data for the applications. The demonstration data is stored in the demo companies.

Information Files

6

Terminal information file

A terminal information file stores information, including the type of terminal, terminal operations, how to access keys on the keyboard, color support, cursor movements, and whether code features are blinking, bold, reversed, and so on.

Terminal information files are stored in the $BSE/lib/terminf directory.

With Enterprise Server, the usage of these files is minimal and adjustments are not required so layout is not described here.

Printer information files

The directory where printer information files are stored is $BSE/lib/printinf.

Create a subdirectory in the printinf directory. The name of the subdirectory must be the first letter of the future printer information file.

The printer information file is stored in this subdirectory. For example, the printer information file for a Mannesmann mt910 is stored in the $BSE/lib/printinf/m/ directory.

The name of the file itself is an abbreviation of the printer name, for example, mt910.

You can include another printer information file at any position in the current file with the **Include** command, for example, include=mt910.

The variables are filled in accordance with the specifications (escape sequences) of the printer as listed in the Printer Manual. You can omit some of these variables, because these variables are not required for the majority of the applications. These variables are marked with an asterisk (*).

A variable definition consists of a name, a =, an escape sequence that contains special characters and a comma. A new line character closes the lines.

If a printer does not offer an escape sequence for boldface and underlined text, and the specific variables are left empty, the filter program simulates these functions by backspace and carriage return. The positions are then covered twice. To switch this simulation off, you can enter the value \000 for the definition of bold on/off or underline on/off, or simply enter no value, for example, pbold=,.

The escape sequences can contain special characters and can be divided into escape codes and control codes, as described in the following table:

Escape codes	Control codes
\b backspace	^A to ^Z
\E or \e escape	^[
\f form feed	^\
\n newline	^]
\r carriage return	^^
\t tab	^_
\s space	
\nn decimal value	nn=1-255
\0nn octal value	nn=1-377
\xnn hexadecimal value	nn=1-FF (for example \x1F)

You can use an octal value instead of an alphabetical representation. For example, “escape” then becomes \033.

The control codes are converted to the ASCII codes 1 to 31. For example, ^B equals the ASCII character with decimal value 2 (STX).

A printer information file can have the following Boolean entries:

- rsf_pbold, reset font after pbold
- rsf_prev, reset font after prev
- rsf_punder, reset font after punder
- rsf_pobold, reset font after pobold
- rsf_porev, reset font after porev
- rsf_pounder, reset font after pounder

A printer information file can have the following variables:

- **barcode_dir=**
  Barcode directory, relative to $BSE/lib/barcode.
- **bin1=**
  Select first paper bin.
- **bin2=**
  Select second paper bin.
- **bin3=**
  Select third paper bin.
- **bin4=**

Select fourth paper bin.

- **hpos=**

  Set horizontal cursor position of the printer.

- **initpage=**

  String sent before each page.

- **initpr=**

  Initialization string that precedes the output to the printer. This variable must be filled with the codes for large typeface.

- **initprog=X**

  The output of program X (full path name) is sent to the printer.

- **initpr2=**

  The second initialization string sent after initprog.

- **landscape=**

  Select landscape printing.

- **large=**

  Large typeface (10 cpi) control. Some printers require a code to change the typeface size.

- **nls_out=**

  NLS output table name.

- **middle=**

  Medium typeface (12cpi) control. Some printers require a code to change the typeface size.

- You can use the following codes to set the print color:
  - **p_black=**
  - **p_red=**
  - **p_green=**
  - **p_yellow=**
  - **p_blue=**
  - **p_magenta=**
  - **p_cyan=**
  - **p_white=**

- **pbold=**

  Print boldface characters. A boldface character can print wider than normal. This does not imply any proportional spacing.

Printing graphic characters, with ASCII code over 127 decimals, yields additional line feeds on most Mannesmann printers. If you do not use this variable, the bshell uses the carriage return character to reprint each line to simulate boldface printing.

- **pcbl=**

  Print +. If this variable cannot be filled, for example, because the printer cannot print the character, a + is used.

- **pcbr=**

  Print +. If this variable cannot be filled, for example, because the printer cannot print the character, a + is used.

- **pctl=**

  Print +. If this variable cannot be filled, for example, because the printer cannot print the character, a + is used.

- **pctr=**

  Print +. If this variable cannot be filled, for example, because the printer cannot print this character, a + is used.

- **pdbl_wide=**

  Double-wide mode on.

- **pdt=**

  Print –. If this variable cannot be filled, for example, because the printer cannot print the character, a + is used.

- **pfont1=, to pfont16=**

  Select font 1 to font 16. These fonts are user definable.

- **phb=**

  Print –. If this variable cannot be filled, for example, because the printer cannot print the character, a – is used.

- **pitalic=**

  Italic mode on.

- **pkr=**

  Print +. If this variable cannot be filled, for example, because the printer cannot print the character, a + is used.

- **plt=**

  Print |. If this variable cannot be filled, for example, because the printer cannot print the character, a + is used.

- **pnlq=**

  NLQ mode on.

- **pobold=**

  Switch off boldface printing without affecting character size.

- **podbl_wide=**
  Double-wide mode off.
- **pofont1=, to pofont16=**
  Deselect font 1 to font 16. These fonts are user-definable.
- **poitalic=**
  Italic mode off.
- **ponlq=**
  NLQ mode off.
- **porev=**
  Reverse mode off.
- **portrait=**
  Select portrait printing.
- **posubscript=**
  Subscript mode off.
- **posuperscript=**
  Superscript mode off.
- **pounder=**
  Underlined printing off.
- **prev=**
  Reverse mode: white on black.
- **prt=**
  Print |. If this variable cannot be filled, for example, because the printer cannot print this character, a + is used.
- **psubscript=**
  Subscript mode on.
- **psuperscript=**
  Superscript mode on.
- **punder=**
  Print underlined characters. If this variable cannot be filled, the bshell prints underscore characters on the next line to simulate underlining.
- **put=**
  Print –. If this variable cannot be filled, for example, because the printer cannot print the character, a + is used.
- **pvb=**
  Print vertical bar. If this variable cannot be filled, for example, because the printer cannot print the character, a | (pipe) is used.

- **q1=, to q13=**

  You can use these thirteen variables with strings that specify whether a character must be printed in double width, italicized, and so on. The variables are userdefinable. These codes have been replaced and will be removed in a future release.

- **resetpr=**

  Resets the printer. The string is sent to the printer after all output. Fill this variable with a form feed (\014) followed by the large typeface codes.

- **resetprog=X**

  The output of program X, full path name is sent after the reset string.

- **resetpr2=**

  The second reset string, sent after resetprog.

- **set0=, to set9=**

  Set a character set of the printer. The default set is set0.

- **small=**

  Small typeface (16.66-17 cpi) control. Some printers require a code to change typeface size.

- **usr1=, to usr16=**

  Define user entry1 to user entry16.

Bar codes

The report writer interfaces with the printer driver using bar-code scripts. A barcode script saves the cursor position, prints a bar code with a given height, and returns to the saved cursor position. These scripts must be available in the barcode_dir directory. This directory is relative to $BSE/lib/barcode, therefore, if barcode_dir = hp_barcode, the bar-code directory is $BSE/lib/barcode/hp_barcode. Bar-code scripts are prefixed by the word type and suffixed by a two-digit number, for example:

```
type01.
```

The following example shows a bar-code script, which saves the cursor position and restores the cursor after the bar code is printed. You can only implement bar codes if this type of scheme is permitted.

```
#!/bin/sh
#
# Sample driver for HP Laserjet 4
# with "Bar Codes & More Font Cartridge"
#
# Prints EAN/UPC with the code under it.
# $1 means the bar code
# $2 means the height of the bar code (number of lines)
#
```

```
code=$1
height=$2
Push()
{
# push cursor position (max 20x)
echo "\033&f0S\c"
}
Pop()
{
# pop cursor position
echo "\033&f1S\c"
}
NextRow()
{
# move to next row, relative
echo "\033&a+1R\c"
}
# - save cursor position - filter assumes same pos. after bar code
print
Push
str1=`echo $code | awk '{print substr($1, 1, 5)}'`
str2=`echo $code | awk '{print substr($1, 6, 5)}'`
# - select EAN/UPC 13 mil font
echo "\033(8Y\033(s1p12v0s3b0T\c"
while [$height -gt 1]
do
Push
echo "($str1-$str2(\c"
Pop
NextRow
height=`expr $height - 1`
done
# - last (empty) line of bar code
Push
echo "(- (\c"
Pop
# - select Courier 12cpi.
echo "\033(10U\033(s0p12.00h10.0v0s0b3T\c"
# - move x+25 dots
echo "\033*p+25X\c"
# - print code (text)
echo "$str1 $str2\c"
# - restore cursor position for filter
Pop
exit 0
```

Example of printer information file for the mt910 printer

This example shows what a printer information file can consist of. You can add comments in the same manner as in a terminal information file:

```
#mt910
small=\E{s16.66H\EL08
large=\E{s10H
middle=\E{s12H
initpr=\E{s12H\EF66
resetpr=\E{s12H\EF66\014
pctl=+
pctr=+
pcbl=+
pcbr=+
put=+
prt=+
plt=+
pdt=+
phb=-
pvb=|
pkr=+
pbold=\E"g1B
pobold=\E"g0B
prev=\E"g1K
porev=\E"g0K
punder=\EI
pounder=\EJ
pitalic=\E{s1S
poitalic=\E{s0S
landscape=\E"g1R
portrait=\E"g0R
nls_out=mt910.out,
```

Executable Programs

7

Database management

This chapter describes the executable programs used for database management that are located in the $BSE/bin directory.

The programs are sorted by type of database server. This chapter also provides references for programs described in other documents.

General

- **audit_srv**

  The name of the server that handles logging of actions on the database. For more information, refer to "Audit Management" on page 133.

- **bdbpre**

  Program to convert database tables to a sequential dump. For more information, refer to "Database Tools" on page 11.

- **bdbpost**

  Program to create database tables from a sequential dump. For more information, refer to "Database Tools" on page 11.

- **bdbreconfig**

  Program to reconfigure database tables. For more information, refer to "Database Tools" on page 11.

- **refint**

  Function to check the referential integrity of database tables. For more information, refer to "Database Tools" on page 11.

- **gcommand**

  Database test program used internally to solve problems. This program is not intended for end users.

- **qptool**

Database test program used internally to solve problems. This program is not intended for end users.

- **bsql**
  Database test program used internally to solve problems. This program is not intended for end users.
- **blogind (Unix)/rexecd (Windows)**
  A program that uses secure passwords to connect the client to the server.

Oracle

- **ora8_srv6.2**
  The driver for Oracle Version 8.
- **ora8_maint6.2**
  The program used to maintain Oracle version 8.

Informix

- **inf_srv6.2**
  The Informix driver itself.
- **inf_maint6.2**
  The Informix version of the program used to maintain the database.

DB2

- **db2v5_srv6.2**
  The driver executable program that functions with DB2.
- **db2v5_maint6.2**
  The maintenance executable program to be used with the DBA module, which works with DB2.

Logic server (bshell)

Bshell

The bshell is a virtual machine between the user interface, database drivers, applications, and the operating system. This program makes applications, including Enterprise Server Tools, independent of the operating system. The bshell functions as a shell between the application and the operating system.

If you start the user interface, bshell connects with the bshell on a machine on the network, which is the usual way to start the bshell. Note that you can run the bshell and user interface on separate machines. The following settings are required:

- Add a line to your local or remote $BSE/lib/ipc_info file similar to the following:

```
bshell s ${BSE}/bin/bshell6.2
```

- When you run on a remote host fill the **r** <*user*> file
- In the User Data Template (ttams1110m000) session, you can specify where the application logic server is located. You must then use the User Data (ttaad2500m000) session to connect the template to the user.

Environment variables and resources

The DS_AS environment variable overrules these settings of the user file (example: DS_AS=host!bshell).

The resource session_timeout can be used to let the bshell terminate itself after a predefined period of inactivity. The defined period is in minutes. A value of 0 means that session_time functionality is not activated, this is the default.

Synopsis

Bshell [options] [program [program arguments]]

Options for bshell are described in the following sections:

- To debug the bshell during run time
  - Baan CPU
  - Scheduler
  - File I/O
  - Miscellaneous

    - Message and log extract
- Memory usage
- Miscellaneous

To pass options to the bshell, enter the options in the command line of the BW Configurator, for example:

```
-set CORE=1 –dbgcpu –dbgfun ttaad2100m000
```

Pass "-set CORE=1 –dbgcpu –dbgfun" options to the bshell and run the Application Configuration (ttaad2100m000) session

Bshell logs all output to **$BSE_TMP/bshell.*PID***. This log file is removed when the bshell exits normally. You can instruct the bshell to keep the log file with the **–keeplog** option.

To debug the bshell during run time

To debug the bshell while the Enterprise Server application runs, open the **Option Dialog** dialog box, which appears as an icon while you run Enterprise Server, and click **Debug Bshell**. The **Run Time Debugging of Bshell** dialog box appears. You can also choose these options from the command line.

Baan CPU

Debug functions

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Debug Functions**, or, from the command line, run **–dbgfun**.

Use debug version of the CPU

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Debug CPU**, or, from the command line, run **–dbgcpu**.

Show Bshell CPU instructions

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Debug Instructions**, or, from the command line, run **–dbginstr**.

Dump 3GL stack traces on function entry

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Dump Stack Traces**, or, from the command line, run **–dbgstack**.

Debug get.var and put.var functions

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Getvat/Putvar debug**, or, from the command line, run **–dbggpvar**.

Show program flow

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Program Flow**, or, from the command line, run **–dbgflow**.

Trace specific bshell functions

On the command line, you can select a specific set of bshell functions to trace. This allows for a finer level of trace without the overhead of **–dbgfun –dbgcpu**. For example, specify **–dbgtrace "seq.open,seq.close"** to trace all the bshell calls to the **seq.open** and **seq.close** functions. You can add '*' at the end. For example, specify **–dbgtrace "xml*,seq.*"** to trace all functions that start with '**xml**' and with '**seq.**'. If you specify **–dbgfun** and **–dbgcpu**, this trace is ignored.

The **–dbgtrace** works without the **–dbgcpu**.

If **–tracelevel** *level* is added, function arguments are also displayed in the trace.

Scheduler

Debug scheduler

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Debug Scheduler**, or, from the command line, run **–dbgsched**.

Show process actions, for example, activate, sleep, and kill

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Show Process Actions**, or, from the command line, run **–dbgmulact**.

File I/O

Show all files that are currently opened by the Bshell

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Show Opened Files**, or, from the command line, run **–dbgfile**.

Debug file access

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Debug File Access**, or, from the command line, run **–dbgfdev**.

Miscellaneous

Show object information

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Show Object Information**, or, from the command line, run **–dbgobj**.

Show whether domains, data definitions and objects are loaded from disk or shared memory

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Show SRDD Use**, or, from the command line, run **–dbgsrdduse**.

Show various TSS debugging information

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Debug TSS**, or, from the command line, run **–dbgtss**.

Show data input options (not for fields)

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Debug data.input()**, or, from the command line, run **–dbgdata**.

Stop debugger, if possible, when a message is sent to the message window

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Debug Messages**, or, from the command line, run **–dbgmesg**.

Show loaded resources

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Debug Resources**, or, from the command line, run **–dbgres**.

Do not remove the logfile after ending the Bshell

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Keep Log File**, or, from the command line, run **–keeplog**.

Add time stamps to bshell log output

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Add Time Stamps**, or, from the command line, run **–logtime**.

Database-related

Show the Baan database activities initiated from the bshell

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Show BDB Actions**, or, from the command line, run **–dbgbdbact**.

Show actions related to enums

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Print Enums**, or, from the command line, run **–dbgenums**.

Show locking errors

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Show Locking Errors**, or, from the command line, run **–dbglck**.

Show database server type

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Show BDB Server Type**, or, from the command line, run **–dbgsrv**.

BDB/SQL tracing

To trace Enterprise Server database and SQL actions, open the **Option Dialog** dialog box and click **Debug Bshell**. The **Run Time Debugging of Bshell** dialog box appears. You can also choose these options from the command line.

BDB Debug Flags

Show the drivers and parameters currently in use

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **BDB/SQL Tracing** tab, select **Driver Type**, or, from the command line, run **BDB_DEBUG=01**.

Show database actions such as Insert, Update, Delete, Commit, and Abort

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **BDB/SQL Tracing** tab, select **Database Actions**, or, from the command line, run **BDB_DEBUG=02**.

Show information on currently set locks

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **BDB/SQL Tracing** tab, select **Delayed Locks**, or, from the command line, run **BDB_DEBUG=04**.

Show references between tables

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **BDB/SQL Tracing** tab, select **References**, or, from the command line, run **BDB_DEBUG=010**.

Show all tables using native storage format

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **BDB/SQL Tracing** tab, select **Multibyte Storage**, or, from the command line, run **BDB_DEBUG=040**.

Show the permissions and roles allowed for each user

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **BDB/SQL Tracing** tab, select **Permissions/Roles**, or, from the command line, run **BDB_DEBUG=0100**.

BDB_DEBUG Value

Shows the current setting that can be used in a **–set BDB_DEBUG=<value>** from the command line.

TT SQL TRACE Flags

Show the full text of a query with an ID number

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **BDB/SQL Tracing** tab, select **Show Query with ID**, or, from the command line, run **TT_SQL_TRACE=040**.

Show how long a query has been running

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **BDB/SQL Tracing** tab, select **Query Execution Times**, or, from the command line, run **TT_SQL_TRACE=0200**.

Show main SQL functions such as SQLExec, SQLParse, SQLFetch, and SQLBind

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **BDB/SQL Tracing** tab, select **Internal SQL Functions**, or, from the command line, run **TT_SQL_TRACE=02000**.

Show the best possible design for indexing, joins, and so on

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **BDB/SQL Tracing** tab, select **Query Evaluation Plan**, or, from the command line, run **TT_SQL_TRACE=04000**.

Show all full table scans

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **BDB/SQL Tracing** tab, select **Show Full Table Scans**, or, from the command line, run **TT_SQL_TRACE=020000**.

Show low-level communication between client and driver

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **BDB/SQL Tracing** tab, select **Show BDB Communication**, or, from the command line, run **TT_SQL_TRACE=040000**.

Show current time to level of milliseconds: YYYYMMDDhhmmss.mmm

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **BDB/SQL Tracing** tab, select **Add Time Stamps (SQL)**, or, from the command line, run **TT_SQL_TRACE=0400000**.

TT_SQL_TRACE value

Shows the current setting that can be used in a **–set TT_SQL_TRACE=<value>** from the command line.

Memory usage

Show currently running processes in bshell

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Dump Process List**. This option is not available from the command line.

Show total memory usage

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Total Memory**, or, from the command line, run **–dbgmemtot**.

Show free memory list

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Free Memory**. Or, from the command line, run **–dbgmemfree**.

Show used memory list

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Memory Used**. Or, from the command line, run **–dbgmemused**.

Show memory usage per block

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Memory Block List**, or, from the command line, run **–dbgmemblk**.

Show all memory statistics

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **All Memory Info**. Or, from the command line, run **–dbgmem**.

Add remark to log file (enter the remark in the field and then click the button)

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Write Remark to Log**. This option is not available from the command line.

Display shared-memory information

In the **Option Dialog** dialog box, select **Debug Bshell**. On the **Bshell Debug Levels** tab, select **Display Shared Mem**. This option is not available from the command line.

Miscellaneous

- **–vV**: Version information
- **–r**: Show resources
- **–deftext**: Show bshell texts
- **–mdebug**: Display bshell messages sent to display server
- **–dbgref**: Show reference paths
- **–dbgrefer**: Show references
- **–dbgbdbact**: Show database actions
- **–dbgenums**: Show loading of enums
- **–dbgpty**: Debug pseudo terminals (pty)
- –dbgorb: Debug ORB integration (where available)
- **–set var=val**: Set environment variable **var** to **val**
- **–logfile <file>**: Log stdout/stderr output in file
- **–appendlog**: Append to Logfile. This option is only useful with the **–logfile** option.
- **–nolog**: stdout and stderr go to the controlling terminal
- **–delay sec**: Delay for *sec* seconds before continuing

bshcmd

You can use this command to change the log facilities of the Logic Server (bshell) or kill one or more bshell processes. You perform the actions with this command while the bshell is running.

Synopsis

bshcmd [options] <bshell_pid>

Possible options

- **–v**: Print version information.
- **–p**: Show process list.

- **–m**: Show memory usage.
- **–d <*dbglvl*>**: Set DEBUG_LEVEL to (octal) <dbglvl>.
  The following <dbglvl> values are available:
  - 0000000001: Show data input actions.
  - 0000000002: Show object information.
  - 0000000004: Show reference paths.
  - 0000000020: Debug functions.
  - 0000000040: Database server information.
  - 0000000100: Show delayed locks.
  - 0000000200: Show process actions, such as sleep, kill, and so on.
  - 0000000400: Database reference information.
  - 0000001000: Show database actions.
  - 0000002000: Debug file access.
  - 0000004000: Show loaded resources.
  - 0000010000: Show loading of **enums**.
  - 0000020000: Show Bshell CPU instructions.
  - 0000100000: Show whether domains, data definitions and objects are loaded from disk or shared memory.
  - 0000200000: Debug **get.var** & **put.var** functions.
  - 0000400000: Debug scheduler.
  - 0001000000: Debug pseudo terminals.
  - 0002000000: Show opened sequential files.
  - 0004000000: Debug TSS functions.
  - 00020000000: Debut ORB integration.
  - 00040000000: Show stack traces.
  - 00100000000: Debug messages.
  - 00200000000: Show program flow.
- **–k *pid***: Kill bshell process ID *pid*.
- **–e**: Kill all bshell processes.
- **–M "*message*"**: Send message to bshell.
- **–W *sec***: Wait until the previously issued command is executed. After this, the previous command is overwritten.
- **–w *sec***: Wait *sec* seconds for bshell to execute command.
- **–u *sec***: Send SIGUSR1 to bshell (wakeup). This option must only be used in combination with the **–w** option. Waits *sec* seconds to see if you run the command.
- **–s**: Show entire contents of log file, if accessible. You must only use this option in combination with the **–p** and **–w** options.
- **-L**: Print log file name of bshell. You must only use this option in combination with the **–p** and **–w** options.

- **-T *cmdstr***: Modify BDB_DEBUG or TT_SQL_TRACE (bshell) and DBSLOG, TT_SGL_TRACE, {DBMS}STAT (drivers) tracing variables. *cmdstr* can contain multiple commands of the form:
  - *trace variable***=***value*: Set variable to value.
  - *trace variable***+***value*: Add bits to variable.
  - *trace variable***-***value*: Remove bits from variable.

Notes

- All output is stored in the bshell's log file, which by default is: **$BSE_TMP/bshell.*pid***.
- Only one command can be active at a time. New commands overwrite previous ones.
- Check the return value to see if a command has been processed.
- The bshell_pid process number is a member of the output of the UNIX **ps** command.

Examples

```
bshcmd –s –p –w 10 <bshell_pid>
```

Show the process list, and wait 10 seconds for a response. If no actions are carried out within 10 seconds, no output is given.

```
bshcmd –d 02000 <bshell_pid>
```

Set DEBUG_LEVEL to 02000.

```
bshcmd –M "Hello" –u 10 –w 10 <bshell_pid>
```

Send a message to a specific bshell.

```
bshcmd –k <pid> <bshell_pid>
```

Kill the bshell process **<**pid**>**. The **<**pid**>** process number can be accessed from the shell program (ttstpshell) with the ps command.

```
bshcmd –T “TT_SQL_TRACE+02000” <bshell_pid>
```

Set TT_SQL_TRACE to log interface calls.

badmin

Used to perform administration of the bshell. The usage is as follows: badmin [-UuVv] [-qo outfile] [-qe errfile] –chkuser **<**user**>** | –chkgroup –ostype

Available options are:

- **–U** or **–u**: Print usage.
- **–V** or **–v**: Print release number.
- **–qo outfile**: Redirect standard output to file outfile.
- **–qe errfile**: Redirect error output to file errfile.
- **–chkuser <*user*>**: Returns a 0 if user exists, otherwise, returns a 1.
- **–chkgroup <*group*>**: Returns a 0 if group exists, otherwise, returns a 1.
- **–chkpwd <*user*>**: Returns a 0 if successful, otherwise, returns a 1.
- **–getpwd <*user*>**: Returns a 0 if successful, otherwise, returns a 1.
- **–ostype <*ostype*>**: Returns 0 if the operating system is ostype, otherwise, returns a 1. Ostype can be either NT or UNIX.
- **–P**: Tag for aged password notification, which is only effective with other flags.

Installation

- **cmt6.2**

  Unix only. Script for component merge tool. Used to migrate the sources from user customized applications to new LN versions.
- **sh_server**

  Shell server used to execute system-dependent commands.
- **bsp.setperm6.2**

  Unix only. Script used to assign the correct permissions to all the Enterprise Server software.
- **binperm6.2.**

  Unix only. Script used to assign the correct permissions to binaries in the $BSE/bin directory after a new porting system is set up.

Development

- **bic**

  Program compiler for Baan C compiler.
- **repgen**

  Report generator.

- **std_gen**
  4GL-program preprocessor.
- **bic_info**
  Shows information for a specified object.

Printer management

- **filter6.2**
  UNIX only: Program to translate Enterprise Server native codes to printer codes that the pdaemon6.2 program uses.
- **lp6.2**
  UNIX only: System spooler interface for the pdaemon6.2 program.
- **pdaemon6.2**
  UNIX only: Daemon process to print requests from the environment. The user **root** must start this program.

The options that you can use to call the printer daemon are the following:

- **–v**: Print version and porting information of the daemon.
- **–k**: Kill the running printer daemon.
- **–f**: Start daemon as foreground process (for debugging).
- **–d *n***: Print debugging information to stderr. The greater the value of n, the more information is output.
- **–r**: Remove the lock file and start daemon. Use this option in rc.start.
- **–h**: Print the previous options.

Resources

- **maxproc**
  Maximum number of background (filter) jobs. The default is 20.
- **maxtries**
  Maximum number of minutes to retry opening the database files. The default is 30 minutes.
- **sleeptime**
  Sleep time between two polls. The default is 10 seconds.
- **shell_cmd**
  Shell that runs the lp6.2 system interface. The default is /bin/sh.

- **strict**

  The interval in seconds that the printer daemon command (up/down) is checked for all devices. If this resource is not set, the printer daemon command of the device in question is checked as soon as a print job is found.

  Change default values in the $BSE/lib/defaults/pdaemon6.2 file.

Note: For best results, you must group physical printers into logical printers. The daemon recognizes whether a printer is busy and chooses another physical printer device. The daemon also calculates the size of the various queues for physical printers, so the daemon chooses the queue with the fewest number of bytes.

Shared memory

- **shmmanager**

  Manager program for shared memory. For more information, refer to "Shared Memory Management" on page 115.

- **shmtimer**

  Daemon process that stores and updates the current time in shared memory. See "Shared Memory Management" on page 115.

- **srdd_init6.2**

  Program to load the data dictionary for the bshell from shared memory.

Network

- **fs**

  Program to handle remote file I/O. With this program, you can access files on another system. On the other system, you must configure the fs program in the ipc_info file. See "Audit Management" on page 133.

- **ipc_boot**

  Program to start remote processes. See "Audit Management" on page 133.

TSS

- **tsscomp**

  Infor internal binary, not to be used by customers.

- **tsscvt**

  TSS conversion filter. For more information, refer to "Multibyte Management" on page 123.

- **tssinfo**

  Provides information about the current TSS settings. For more information, refer to "Multibyte Management" on page 123.

Miscellaneous

- **binput**

  Unix only. Program to read input. You can use this program in shell scripts.

- **bput**

  Unix only. Program to set or print terminal settings.

- **compress**

  Program to compress data. Equivalent to the UNIX **compress** command.

- **diff**

  Program used to compare two files. Equivalent to the UNIX **diff** command.

- **encrypt**

  Program to encrypt passwords that you can use to fill the *r<user>* file.

- **nlsedit**

  Editor to maintain input and output conversion tables for native language support. See "Native Language Support" on page 51.

- **popup**

  Unix only. Program to handle pop-up screens. You can use this program in shell scripts.

- **sort**

  Sort program, equivalent of the UNIX **sort** command. To specify a path where temporary files are stored during the sort process, use the BSE_SORT environment variable.

- **sum**

  Sum program that you can use to calculate and patch executable programs.

Errors

8

General

Three categories of errors are distinguishable:

- Error numbers 1-99 are UNIX-generated.
- Error numbers 100-899 are database errors.
- Error numbers 900-999 are network errors.

Error numbers between 34 and 100 are system-dependent.

UNIX errors

1 EPERM Not owner

This error indicates an attempt to modify a file that cannot be modified, except by the file's owner or a super user. This error also appears if ordinary users attempt actions permitted only to the super user.

2 ENOENT No such file or directory

This error occurs if a specified file name must exist but does not, or if one of the directories in a path name does not exist.

3 ESRCH No such process

This error signifies that no process can be found that corresponds to the process specified.

4 EINTR Interrupted system call

This error signifies that an asynchronous signal, such as interrupt or quit, which the user has chosen to catch occurred during a system call. If the system resumes execution after processing the signal, it will appear as if the interrupted system call returned this error code.

5 EIO I/O error

This error signifies that a physical I/O error has occurred. In some cases, this error can point to the call subsequent to the call to which the error actually applies.

6 ENXIO No such device or address

This error signifies that I/O on a special file refers to a sub-device that either does not exist or that is beyond the limits of the device. This error can also occur if, for example, a tape drive is not online or no disk pack is loaded on a drive.

7 E2BIG Arg list too long

This error occurs if an argument list longer than 5120 bytes is presented to a member of the UNIX exec family system calls.

8 ENOEXEC Exec format error

This error signifies that a request has been made to execute a file, which, although the file has the appropriate permissions, does not start with a valid magic number. A magic number is the first two bytes in a file, which are used to determine the type of the file. See a.out(5).

9 EBADF Bad file number

This error signifies that either a file descriptor does not refer to an open file, or that a write request has been made to a file that is read-only, or that a read request has been made to a file that is write-only.

10 ECHILD No child processes

This error signifies that a process that has no child processes has executed a wait. A child process is a process spawned by another process, which is called the parent process.

11 EAGAIN No more processes

This error alerts you that a fork failed, either because the process table of the system is full or because the user is not permitted to create any more processes.

12 ENOMEM Not enough space

This error signifies that, during an **exec** or **sbrk**, a program has requested more space than the system is able to supply. This error is not a temporary condition. The maximum space size is a system parameter. The error can also occur when the arrangement of text, data, and stack segments requires too many segmentation registers, or if not enough swap space is available during a fork.

13 EACCES Permission denied

This error signifies that an attempt was made to access a file or process in a manner forbidden by the protection system.

14 EFAULT Bad address

This error signifies that the system encountered a hardware fault when attempting to use an argument of a system call.

15 ENOTBLK Block device required

This error signifies that a non-block file was specified where a block device was required, for example, in mount.

16 EBUSY Device busy

This error signifies that an attempt was made to mount a device that was already mounted or to dismount a device on which an active file exists, such as an open file, current directory, mounted-on file, or active text segment. This error also occurs if you attempt to enable accounting that is already enabled.

17 EEXIST File exists

This error alerts you that an existing file was specified in an inappropriate context, such as in a link.

18 EXDEV Cross-device link

This error signifies that an attempt was made to link to a file on another device.

19 ENODEV No such device

This error signifies that an attempt was made to apply an inappropriate system call to a device, for example, a read on a write-only device.

20 ENOTDIR Not a directory

This error signifies that a non-directory was specified where a directory is required, for example, in a path prefix or as an argument to chdir(S).

21 EISDIR Is a directory

This error signifies that an attempt was made to write to a directory.

22 EINVAL Invalid argument

This error signifies that an invalid argument was used, for example, if you do the following:

- Dismount a non-mounted device.
- Specify an undefined signal in **signal** or **kill**.
- Read or write a file for which **lseek** has generated a negative pointer.

This error is also used by the math functions described in the (S) entries of the UNIX manual.

23 ENFILE File table overflow

This error signifies that the systems table of open files is full, and no more open commands can currently be accepted.

24 EMFILE Too many open files

This error signifies that too many file descriptors are open.

25 ENOTTY Not a typewriter

This error signifies that the selected device does not have the properties of a terminal.

26 ETXTBSY Text file busy

This error signifies that an attempt was made to execute a pure-procedure program that is currently open for writing or reading. This error can also signify that an attempt was made to open for writing a pure-procedure program that is being executed.

27 EFBIG File too large

This error signifies that the size of a file exceeded the maximum, or the configured ULIMIT. See ulimit(S).

28 ENOSPC No space left on device

This error alerts you that, when an ordinary file is written, no free space is left on the device.

29 ESPIPE Illegal seek

This error signifies that an **lseek** was issued to a pipe.

30 EROFS Read-only file system

This error signifies that an attempt was made to modify a file or directory on a read-only device.

31 EMLINK Too many links

This error signifies that an attempt was made to link more than the maximum number of links to a file. The maximum number of links to one file is 1,000.

32 EPIPE Broken pipe

This error signifies that a write was made on a pipe for which no process is available to read the data. This condition usually generates a signal. The error is returned if you ignore the signal.

33 EDOM Math arg out of domain of func

This error means that the argument of a function in the math package is out of the domain of the function.

34 ERANGE Math result not representable

This error means that the value of a function in the math package cannot be represented within machine precision.

Database errors

100 EDUPL

This error means that a duplicate value exists.

101 ENOTOPEN

This error means that the table is not open.

102 EBADARG

This error means that an illegal argument has been specified.

103 EBADKEY

This error signifies that an illegal key description has been specified. Use the **bdbpre** and **bdbpost** tools.

106 ENOTEXCL

This error alerts you that the table is not exclusively locked action. You can either wait until the lock on table is released, or you can remove the lock yourself.

107 ELOCKED

This error signifies that the record you are trying to retrieve is locked. You can either wait until the lock is released or remove the lock yourself.

108 EKEXISTS

This error informs you that the key already exists.

109 EPRIMKEY

This error alerts you that you are trying to perform an illegal action on a primary key. For more information, refer to the log file.

110 EENDFILE

This error signifies that you have reached the end of the file.

111 ENOREC

This error alerts you that no record was found that matches the query criteria.

112 ENOCURR

This error signifies that no current record exists.

113 EFLOCKED

This error signifies that the table is locked. You can either wait until the lock is released or you can remove the lock.

114 EFNAME File name too long

This error signifies that you are using a file name that is too long. To check the maximum length for a file name, refer to your system requirements.

116 EBADMEM

This error signifies that the system cannot allocate memory because the system is out of memory. One possible solution is to try to restart the bshell.

117 EBADCOLL

This error signifies that a problem has occurred with the collating or sorting order.

123 ENOSHMEM

This error alerts you that that no shared memory is initialized. To initialize shared memory, you can, for example, use the rc.start script.

129 EUSER

This error alerts you that too many sessions have been started.

136 ENOSPACE

This error alerts you that no space is available in shared memory. You can either initialize the shared memory or change the shared memory parameters.

140 ETRANSON

This error signifies that this operation is invalid when the transaction is on.

141 ETRANSOFF

This error alerts you that this operation is invalid when the transaction is off.

142 EADMON

This error alerts you that an administration process is running.

146 ENOSNAPSHOT

This error signifies that the system cannot take a snapshot of the database, most likely because another user locked the database. You can either wait until the lock is released, or you can remove the lock.

148 EOFRANGE

This error signifies that an error occurred in the data-type range check.

201 EROWCHANGED

This error alerts you that the record was changed after a delayed lock.

202 EDBLOCKED

This error alerts you that the database is locked. You can either wait until the lock is released or you can remove the lock yourself.

203 ETRANSACTIONON

This error signifies that this action is not permitted within a transaction.

204 EISREADONLY

This error signifies that this transaction is read-only.

205 ENOTINRANGE

This error signifies that the field value is out of range and does not agree with the domain definition.

206 ENOTLOCKED

This error specifies that the record is not locked.

207 EAUDIT

This error signifies that an error occurred in the audit trailer.

208 EPERMISSION

This error alerts you that the action you just attempted is not permitted at this time.

209 EMIRROR

This error signifies that an error occurred in the mirroring of the database. The tables are inconsistent. You can use **bdbpre** and **bdbpost** to copy the tables correctly.

210 EMLOCKED

This error alerts you that either the record is locked in the mirrored database, that the tables are inconsistent, or that the mirroring definition in tabledef6.2 is not compatible.

213 ETRANSACTIONOPEN

This error alerts you that the transaction is started, but not updated. This error is an internal bshell error.

214 EUNALLOWEDCOMPNR

This error alerts you that an operation for mapping company numbers is not permitted. If the logical company is not equal to the physical company, you cannot perform a drop/clear table operation.

215 EILLEGAL

This error indicates an illegal state that must never occur.

220 EDBCMMODELCORRUPT

This error indicates that the current DBCM model is corrupt. For more information, see the log messages.

221 EDBCMACTIONNOTALLOWED

This error indicates that an action is not allowed in DBCM context. For example, if a table is under DBCM, an update on the primary key is not allowed. Other actions that are not allowed are updates that would make a record migrate from one to another object.

222 EDBCMOBJECTCORRUPT

This error indicates that a corrupt object is detected. This typically occurs if no root record of the object can be determined.

251 EAUDSETUP

This error indicates that the audit server is set up incorrectly. For more information, refer to the Log.audit file.

252 EAUDCORRUPT

This error alerts you that an audit file is corrupt. For more information, see the Log.audit file.

253 EAUDLOCKED

This error indicates that another user has locked the audit file. For more information, see the Log.audit file.

254 EAUDABORT

This error indicates that a commit transaction has failed in the audit-server action. For more information, see the Log.audit file.

301 ESQLQUERY

This error is a general SQL error code. This error occurs if a problem with the SQL syntax arises.

302 ESQLSYNTAX

This error indicates that the SQL syntax is incorrect.

303 ESQLREFER

This error signifies that a reference in the query cannot be found.

304 ESQLUNDEFINED

This error occurs if an error occurred but no error code can be set.

305 ESQLWRONGROW

This error signifies that an incorrect record was returned. This implies that either the table index is corrupt or that the RDBMS has sorting order that differs from the Enterprise Server software.

501 EMEMORY

This error is an internal memory error.

502 EBDBON

This error indicates that the user is already logged on.

503 EBADADRS

This error signifies that an illegal address has been used.

504 EBADFLD

This error indicates that a column is undefined.

505 ENOSERVER

This error signifies either that no server is specified in tabledef6.2, or that the server cannot be started. For more information, refer to the log file.

506 ENOTABLE

This error indicates that the table does not exist.

507 ETABLEEXIST

This error signifies that the table that you are trying to create already exists.

508 EBDBNOTON

This error indicates that you are not logged on to a database.

509 EBADCURSOR

This error indicates that you have a bad memory cursor or that you have specified a bad table pointer.

510 EDBNOTON

This error indicates that the database is not on. To correct the problem, simply start the database.

511 EWRONGVERSION

This error signifies that the version of client differs from the version of the server.

512 EDDCORRUPT

This error indicates that the data dictionary is corrupt. To repair this error, use **bdbpre** and **bdbpost** tools.

513 ENODD

This error indicates that the data dictionary was not found.

514 ESECURITY (Oracle)

This error is a security error and often indicates that you do not have the proper user or group permission.

515 ELICENSEERROR

This error is a license error that usually indicates an unpatched binary.

516 EUPDSEGM

This error occurs during the making or filling of rollback segments and usually indicates that the disk is full.

517 EDELAYED

This error is a general error that indicates delayed locking.

518 ENOSESSION

This error alerts you that an invalid session code has been specified.

519 ENOCOMPNR

This error signifies that either no company number or an illegal company number has been specified. A valid company number is a number between 0 and 999.

520 EBUFUPD

This error occurs when flushing of buffered updates fails. The flushing can fail due to a lock or referential integrity constraint.

521 ENOSHM

This error indicates that shared memory has not been loaded. For more information about how to start shared memory, refer to "Shared Memory Management" on page 115.

522 EBDBDBCONNECTIONLOST

This error alerts you that the connection between the driver and database has been lost.

523 E_BDB_FULL

The user’s request cannot be carried out because the required space is not available in the database. The database administrator must either extend or reorganize allocated space manually.

600 EREFERENCE

This error is a general reference error. For more information, see the log file.

601 EREFLOCKED

This error indicates that the reference table is locked. You can either wait until the lock is released or remove the lock yourself.

602 EUNDEFREF

This error indicates that the reference is not defined. The problem is likely located in the run time data dictionary. For more information, see the log file.

604 EREFUPDATE

This error indicates that a reference could not be updated.

605 EREFEXISTS

This error signifies that the record cannot be deleted while the reference exists. For more information, see the log file.

606 EREFNOTEXISTS

This error indicates that the reference does not exist.

607 ENOREFTBL

This error alerts you that the reference table was not found. The data dictionary might be incorrect. For more information, see the log file.

608 ENOREFCNT

This error signifies that no reference counter fields are present.

609 EUPDREFCNT

This error can occur when you update the reference counter.

700 ESETLOCALE

This error can occur when you set the locale. For more information, see the log file.

If an error code greater than 1000 appears, the error can be retrieved as follows, depending on the database driver you use:

Informix/Oracle: error – 1000 gives DB error

Example UNIX:

```
Error no: 11400
Error: 11400 - 1000 = 10400
UNIX error = 0 (last two digits)
```

Example Oracle:

```
Error no: 1979
Error: 1979 - 1000  = 979 Not a GROUP BY
```

Note: If a fatal error occurs, more information is stored in the log files in the $BSE/log directory. For example, if **bdbpost** causes an error, the error is reported in the Log.bdbpost file.

Shared Memory Management

9

General

Shared memory is a part of the internal memory intended for common usage. All users can read from and write to shared memory. In Enterprise Server Tools, the shared memory contains part of the data dictionary.

A prerequisite for shared memory is that the hardware must support the shared memory and be sufficient internal memory must be available.

This chapter describes the way in which you can use shared memory for Enterprise Server Tools.

To use the shared memory manager

The shared memory manager is located in the $BSE/bin directory and can be started as follows:

```
shmmanager [-iksavr]
```

The options are:

- **–i**: Initialize shared memory.
- **–k**: Delete created shared-memory blocks. After you delete shared memory, you must re-initialize the memory. You can only use this option if all processes that use shared memory have finished. Otherwise, an error message is displayed.
- **–s**: Display technical information on the contents of shared memory. This includes the following:
  - Common bshell pointers
  - Description of segment table
- **–a**: Display of allocation data, such as addresses, segments, size, and so on.
- **–v**: Display machine data and porting data.
- **–r**: Reset shared memory. Removes all data from shared memory, except the first segment. You do not have to reinstall shared memory afterwards. The result of this option is the same as when

you use the **–k** option followed by the **–i** option. You can only use this option if all processes that use shared memory have finished. If not, the system displays a message:

```
# of attaches = <number>
Cannot remove... # of attaches not equal to zero
```

The following section, "Initialization of shared memory" on page 116, discusses the **–a**, **–i**, and **–v** options.

Initialization of shared memory

After the kernel selects a start address, the kernel does not always leave enough memory for the bshell. You must, therefore, specify a virtual address to determine where shared memory is allocated in internal memory.

You can determine an adequate address using the shared memory manager, **shmmanager**. This section includes the instructions for the initialization of shared memory. Following the initialization is a description of error messages that you can encounter during initialization. The section "Kernel requirements" on page 121, contains the minimum values of the kernel parameters.

To initialize shared memory

Use the **–i** option to initialize shared memory, for example:

```
# shmmanager6.2 –i
BUFSZ 524288, MAXATTCH 10, START 0xa40000, STEP 0x80000
Start /usr/bse/bin/shmtimer6.2:
Shmtimer started: pid = 22743, time = 790343035
(Tue Jan 17 12:43:55 1995)
Starting successful
```

Note: During initialization, the BSE shell variable must be the same as for the bshell users. Rather than initializing shared memory manually each time, you can simply include the installation in your system’s start-up procedure. Make sure the BSE variable is set correctly before startup.

To start shmtimer

The UNIX **time()** function is a system call often called in database servers. On multiprocessor systems, use of the **time()** function can be quite heavy. Even though a process can run successively on different processors, this must be invisible to the process. The process requires that each processor return the

same time, using **time()**. Therefore, the processors must synchronize their internal clocks each and every time the **time()** function is called.

The **shmtimer** program prevents this overhead. **shmtimer** is a daemon process that writes the current time in shared memory. The time is updated every second, which reduces the number of calls of the **time()** function to a maximum of one per second.

When you initialize shared memory (**shmmanager –i**), the **shmtimer** program starts. If you remove shared memory (**shmmanager –k**), the **shmtimer** stops first. The following option is available in **shmtimer**:

- **–i**: Initialize shmtimer

  A **shmtimer** program is started, and runs in the background.

- **–k**: Kill shmtimer

  The running **shmtimer** program is killed. The allocated shared memory is cleared, but not removed.

- **–s**: Show status as stored in shared memory:
  - The current time
  - The process ID of the running timer
- **–u**: Show information about shmtimer
- **–v**: show version and porting data of shmtimer

If **shmtimer** stops, the shared memory is cleared, as described for the **–k** option. From then on, the system calls the UNIX **time()** function, rather than reading the time from shared memory.

If **shmtimer** is killed, **shmtimer** cannot clear its shared memory. Because **shmtimer** is killed, the time in shared memory is never updated, but the time is still read from shared memory. In that case, **shmtimer –s** returns a warning. You can start a new **shmtimer** with **shmtimer –i**.

Error messages during installation

This section describes the error messages in the order in which the errors can occur.

No more space

```
# shmmanager6.2 –a
No. of kbytes to be malloc: 4096
no more space
abort – core dumped
```

This error occurs because some computer systems place a limit on the amount of virtual memory used a process uses. As a result, the shared memory manager cannot allocate 4,096 KB.

Solution: Adjust the kernel parameters. For more information, refer to "Kernel requirements" on page 121.

Cannot create shared-memory segments of 512K

```
# shmmanager6.2 –a
No. of kbytes to be malloc: 4096
No. of shm segments to be created: 10
shmget errno 22
etc......
Errno 22 (EINVAL): The kernel cannot create any shared-memory segments
 of 512 K (SHMMAX).
```

Solution: Adjust the kernel parameters. For more information, refer to "Kernel requirements" on page 121.

Not enough virtual memory

```
# shmmanager6.2 –a
No. of kilobytes to be malloc: 4096
No. of shm segments to be created: 10
shmat errno XX id 46
addr[0]: 0xa40000 id[0]: 40
addr[1]: 0xac0000 id[1]: 41
addr[2]: 0xb40000 id[2]: 42
addr[3]: 0xbc0000 id[3]: 43
addr[4]: 0xc40000 id[4]: 44
addr[5]: 0xcc0000 id[5]: 45
addr[6]: 0xffffffff id[6]: 46 (first invalid segment)
addr[7]: 0 id[7]: 6747 (junk)
addr[8]: 0 id[8]: 9644 (junk)
addr[9]: 0 id[9]: 20492 (junk)
step 0x80000
step 0x80000
step 0x80000
step 0x80000
step 0x80000
step 0x80000
step 0xff53ffff
step 0x1
step 0
ret 0
```

Errno XX can be:

- 12 (ENOMEM): The user process does not have sufficient virtual memory.
- 24 (EMFILE): (In this example) the kernel cannot create more than 6 shared-memory segments per process (SHMSEG).

Solution: Adjust the kernel parameters. For more information, refer to "Kernel requirements" on page 121.

Syntax srdd_tab

The Srdd_tab6.2 file can be divided into several sections. Each section is indicated by the package combination for which components must be loaded into shared memory, for example:

```
package=31Sa
```

This parameter means that the components below this line belong to package combination 31Sa.

After this line, the domains are specified first, for example:

- **dti.pd**
- **dtt.pd**

The following data is the table definitions, for example:

- **dttaad000**
- **dttaad001**

Subsequently, you must place the objects, for example:

- **ottstpconv**
- **ottstpdisplay**

As a result, the syntax must be the following:

- **package={pc}**
- **for domains: <d{p}.pd>**
- **for data definitions: <d{p}{t}>**
- **for objects and reports: <o{p}{m}{o}>**

The abbreviations are:

- **{pc} package combination**
- **{p} package**
- **{m} module**
- **{t} table**
- **{o} object (program/report)**

You can also read parts of a data dictionary on a remote system into shared memory. To do this, write the host name of the remote system at the beginning of a line, for example:

```
${BSE}/standard6.2/ddbaan/dti/dti.pd
ibm1!/usr3/bse/standard6.2/ddbaan/dtd/dtd.pd
${BSE}/standard6.2/ddbaan/dttadv/dttadv100
${BSE}/standard6.2/ddbaan/dtppdm/dtppdm100
/usr2/bse/standard6.2/ttB40_a_CP/ottadv/oadv1100
```

If the srdd_tab6.2 file contains a redirection to a remote system, a remote user file must be available for the user who executes srdd_init6.2.

To fill shared memory

You can load parts of the data dictionary into shared memory, such as data definitions, objects, reports, and domains.

To specify which parts must be loaded into the shared memory, use the following sessions:

- Shared Memory Data (ttaad4156m000):

  To load program and report objects.

- Package Combinations (ttaad1120s000):

  To load table definitions and domains.

After a conversion to runtime, the data specified in these sessions is stored in the $BSE/lib/srdd_tab6.2 file.

For details on the procedure to fill the shared memory, refer to *Infor Enterprise Server Administration Guide (U8854 US)*.

The parts of the data dictionary loaded into shared memory have an entry that consists of the inode number, date, time, and name. If the entry of a part in shared memory does not match the entry for that part on hard disk, the parts on hard disk are loaded locally, but not into shared memory. The old part remains in shared memory until you execute shmmanager **–k** to delete shared memory blocks. You must then use shmmanager **–i** to reinstall shared memory.

Use the **srdd_init** program to load the parts of the data dictionary into shared memory. The available options are:

- **–l**: Print what actions the program has performed, including error messages, if any. Use only in combination with the **–i** option.
- **–i**: Initialize the parts that must be loaded in shared memory.
- **–p**: Print a list of loaded components.
- **–v**: Version information of the program.

Use the shmmanager **–s** command or the ipcs **–m** UNIX command to see which segments are used and how much space is left to load other parts of the data dictionary.

Error messages and other messages are written to stderr. You can then enter the following command to write these messages to a file:

srdd_init –i 2> *file name*

The ${BSE}/etc directory contains the Rc.start file. This program is executed at system startup. Shared memory is installed, filled, and initialized for a BSE environment specified in this file. The **shmmanager**, and **srdd_init** programs are used.

Kernel requirements

For tips and hints on how to set the kernel parameters, refer to the kernel Tuning Guides. The available guide is the *Infor LN - Performance, Tracing and Tuning Guide (U9357 US)*.

Multibyte Management

10

General

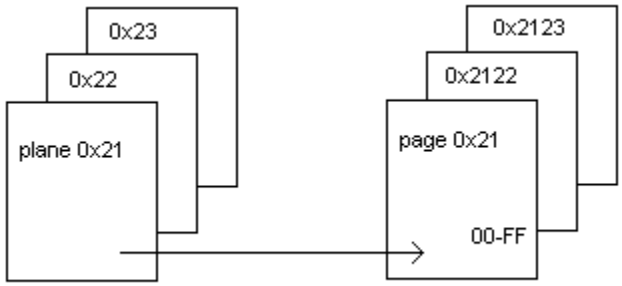
TSS is the LN solution to support languages that require a character set of at least 16 bits, like the languages of the Asian countries. A 32-bit wide character space is implemented in Enterprise Server to support all possible character sets.

Be aware that a character is not necessarily the same as a byte and can take up more than one screen position. A character in the bshell occupies one or two screen positions at a size of 4 bytes.

TSS

TSS is a collection of character sets that you can use in Enterprise Server applications. TSS accommodates all multibyte character sets in a single package. TSS includes both ASCII character sets and multibyte character sets, such as Japanese and Chinese. To distinguish between, for example, Japanese and Chinese, TSS consists of planes, each of which holds a 64 KB-size character set, or 256 pages of 256 characters, as shown in the following figure.

Numbers are according to the hexadecimal system, for example, the value of 0x21 is 33 in the decimal notation.



The characters 0x00 – 0x20 and 0x7F – 0x9F cannot be included in the sets, because these ranges contain control and escape codes, which leaves the following ranges:

```
0x212121 – 0x7EFFFF
0xA12121 – 0xFFFFFF
```

To distinguish TSS characters, the character contains the prefix 0x9B. As a result, the complete TSS ranges are the following:

```
0x9B212121 – 0x9B7EFFFF
0x9BA12121 – 0x9BFFFFFF
```

Code Features (CF) and Line Drawing Characters (LDC) have been implemented in the code range of ISO-8859/1 as follows:

- Line Drawing Characters | 0x80 – 0x8A
- Code Features | 0x8B – 0x9A
- TSS Lead byte | 0x9B
- Reserved | 0x9C – 0x9F

These code ranges do not conflict with the currently implemented EUC character sets, but can cause problems when you import these codes. In addition, these codes overlap with SHIFT-JIS trailing bytes, which causes import conversion errors.

Therefore, the code ranges are translated into ASCII characters (0x000x7F) during the export phase and are converted back into the correct LDC and CF characters during the import phase.

The currently assigned planes are the following:

Character set	Page	Range from	Range to	Remarks
KANJIEUC/SHIFTJIS	0x21	212121	217e7e	JISX0208-1983
KANJIEUC/SHIFTJIS	0x23	2321a1	2321df	JISX0201-1976
GB2312-80	0x25	25a1a1	25ffff	GB2312-80
BIG5	0x27	27a140	27f9d5	

Unicode support

Unicode is a unification of all (or at least: almost all) currently available character sets. See http://www.unicode.org.

TSS is the Baan proprietary character set, and contains all the native character sets supported by the Enterprise Server porting set.

For many years, Unicode and TSS were independent.

At a certain moment, mapping between the two became available in the Enterprise Server porting set. TSS was used internally, and at the borders of the system conversion from and to Unicode became possible. Examples are: communication with BI, with Web UI, and exchange of XML documents. The conversion was done by means of external tables, one pair of 128 kbyte tables for each supported native character set.

The next change in the relationship between Unicode and TSS was to embed the complete Unicode character set in TSS. In this way it became possible to always use TSS internally, and at the same time support all Unicode characters, independent of supported native character sets. Conversion between TSS and Unicode at the borders of the system became a matter of some algorithmic bit shuffling and no longer needed conversion tables.

This chapter describes several aspects of the embedding of Unicode in TSS. First, the most important differences between using the old and using the new TSS are discussed. Then, the most important differences between using old TSS in the old situation and using the same old TSS in the new situation are discussed. Finally, some things which are not changed at all are explicitly mentioned.

Legacy TSS versus UTF-T

The old TSS is called legacy TSS. The embedding of Unicode in TSS is called UTF-T, analogously to the names UTF-8, UTF-16, and UTF-32 for the three standard Unicode Encoding Forms.

TSS consists of two types of characters: single byte and multi byte. The single byte characters use hexadecimal values 0 … FF, except 9B. Multi byte TSS characters use a sequence of 4 bytes, the first of which has hexadecimal value 9B. The following table describes how all supported native character sets are mapped into the available TSS space.

TSS range (hexadecimal)	Meaning
00 - 7F	ASCII character.
80 - 8A	Line drawing character.
8B - 9A	Code feature.
9B	Lead byte for 4-byte TSS characters
9C-9E	Reserved for future use.
9F	Used to represent the Euro Symbol in a Cyrillic context. In ISO8859-5 there is no room available in the normally used range A0 - FF.
A0 - FF	Is ambiguous. It corresponds to a 'high ASCII' character in one of the ISO8859-n character sets. Can be converted (without actual conversion) to the correct ISO8859-n character set, or (often with some offset) to the corresponding Windows Code Page.
9B 21 pp qq	Japanese (Kanji). Can be converted to Kanji EUC, Shift JIS, or Windows Code Page 932

TSS range (hexadecimal)	Meaning
9B 23 21 pp	Single width Japanese. Can be converted to Kanji EUC, Shift JIS, or Windows Code Page 932
9B 25 pp qq	Simplified Chinese. Can be converted to GB2312-80 or Windows Code Page 936
9B 27 pp qq	Traditional Chinese. Can be converted to Big 5 or Windows Code Page 950
9B 31 pp qq	Korean (Wansung). Can be converted to Wansung or Code Page 949.
9B 32 pp qq	Korean (Johab). Can be converted to Johab or Code Page 1361.
9B 9C 9D nn with: 40 < nn < BF	Is ambiguous. It corresponds to Microsoft extension character nn + 0x40 in one of the Windows Code Pages corresponding to an ISO8859-n character set. Can be converted to the correct Windows Code Page. The value nn + 0x40 is in the high ASCII range 0x80 - 0xFF, so nn is in the range 0x40 - 0xBF. Only 55 positions are really used, and only 3 of them are really ambiguous.

The ambiguity shown above for TSS characters in the ranges A0 – FF and 9B 9C 9D 40 – 9B 9C 9D BF is resolved by interpreting these characters in the context of the character set of the current locale.

The embedding of Unicode in TSS uses the single byte ASCII range and a major part of the remaining TSS space, as shown in the following table. Notice that this embedding does not increase the existing ambiguity described above.

TSS range (hexadecimal)	Meaning
00 - 7F	ASCII character. Corresponding to the first 128 Unicode characters U+0000 - U+007F. Can be converted (without actual conversion) to single byte UTF-8 or to single word UTF-16.
9B pp qq rr with: BC < pp < BF, 80 < qq < FF, 80 < rr < FF	UTF-T corresponding to the first 216 Unicode characters U+0000 - U+FFFF, the so called Basic Multilingual Plane (BMP), except for the first 128 Unicode characters U+0000 - U+007F (corresponding to the ASCII character set, and mapped to single byte TSS). Can be converted algorithmically (bit shuffling) to 2-byte or 3-byte UTF-8 or single word UTF-16.
9B pp qq rr with: C0 < pp < FF, 80 < qq < FF, 80 < rr < FF	UTF-T corresponding to the 220 so called Supplementary Unicode characters U+010000 - U+10FFFF. Can be converted algorithmically (bit shuffling) to 4-byte UTF-8 or double word UTF-16.

When converting from TSS to some other character set, the Enterprise Server porting set can interpret both legacy TSS and UTF-T. The other way around, when converting from some other character set to TSS, it depends on the so called TssMode whether legacy TSS of UTF-T is produced. The TssMode is determined by the content of the file $BSE/lib/tss_mbstore6.2 file. If the first line of this file consists

of exactly the text “UTF-T”, then the mode is called ‘UTF-T mode’ and the conversion produces UTF-T. Otherwise the mode is called ‘legacy mode’ and the conversion produces legacy TSS.

In the table below this and further differences between UTF-T mode and legacy mode are indicated. These differences are so essential that it is not allowed to switch the mode at arbitrary moments. Switching from legacy mode to UTF-T mode is allowed at the price of a complete database conversion. Switching back from UTF-T mode to legacy mode is not allowed.

Legacy mode	UTF-T mode
Conversion from any character set to TSS produces legacy TSS	Conversion from any character set to TSS produces UTF-T
Conversion from any Unicode encoding (UTF-8 or UTF-16) to TSS uses conversion tables and fails for characters which do not exist in the character set of the current locale.	Conversion from any Unicode encoding (UTF-8 or UTF-16) to TSS does not need conversion tables and will not fail.
Multi Byte Data in the database is stored in the native character set of the database	Multi Byte Data in the database is stored in Unicode (probably UTF-16, possibly UTF-8) for Multi Byte, in a native locale for Single Byte
Each user must use a language and corresponding locale of which the character set corresponds to the character set used in the database	Each user can choose a language and locale, independent of the character set used in the database.
Single Byte Data in the database is sorted using binary sort, e.g. A Z a z À Ý à ý	Data in the database is sorted according to the Unicode Collation Algorithm, e.g. a A à À ý Ý z Z
For single byte character sets, the non-ASCII characters are mapped to single byte TSS (with some exceptions which are mapped to 9B 9C 9D nn)	For single byte character sets, the non-ASCII characters are mapped to 4-byte UTF-T

Legacy TSS in the old and in the new situation

When UTF-T was introduced, legacy TSS remained supported. However, the implementation of the support was drastically changed, together with the implementation of the new UTF-T support. In the table below the differences between the old and the new support of legacy TSS are indicated

Legacy TSS in the old situation	Legacy TSS in the new situation
Conversion from TSS to any other character set recognizes only TSS corresponding to the character set of the current locale. E.g. in a Simplified Chinese locale only multi byte TSS in the range 9B 25 pp qq is recognized, and multi byte TSS in the range 9B 27 pp qq is not recognized.	Conversion from TSS to any other character set recognizes all TSS characters. Ambiguous TSS characters are interpreted according to the character set of the current locale (if that is a single byte character set) or according to the ISO 8859-1 character set (otherwise).
Conversion from any Unicode encoding to any other character set always contains a conversion	No external Unicode tables are needed. The amount and size of the internal tables is less than

Legacy TSS in the old situation	Legacy TSS in the new situation
from single word UTF-16 to the character set of the current locale, using an external table of 128 kbyte in the file $BSE/lib/Unicode/<charset>.U2N, loaded in shared memory	the amount and size of the previously used external tables. The internal tables are integrated as read-only data into the executables of the porting set.
Conversion from any other character set to a Unicode encoding always contains a conversion from the character set of the current locale to single word UTF-16, using an external table of 128 kbyte in the file $BSE/lib/Unicode/<charset>.N2U, loaded in shared memory	See above.
The supported character sets are defined in some (password protected) sessions, dumped in file $BSE/lib/tss6.2 and compiled to runtime file $BSE/lib/tss_c6.2. This file is used for conversion between TSS and the native character set of the current locale.	No external file is needed for conversion between TSS and any native character set. The conversion is almost completely algorithmic. Only for single byte Windows Code Pages small tables are needed, integrated as read-only data into the executables of the porting set.
The file $BSE/lib/tss_c6.2 contains also references to $BSE/lib/locale/<locale name>, used for character width determination.	No external file is needed for character width determination.

Explicitly mentioned unchanged things

Apart from the changes mentioned in previous sections, all the rest remains unchanged. Some of the unchanged things are explicitly mentioned in the table below.

Unchanged things
Each user has a current language and a current locale.
In the 3GL/4GL programming language, multi byte strings are assumed to contain text encoded in TSS, be it legacy TSS or UTF-T

Font selection in BW when using UTF-T/Unicode

Whether a given character is displayed correctly does not only depend on the correct character encoding but also depends on the selected font. One benefit of Unicode is the ability to represent many languages and scripts in a single string. This is also a problem, since very few fonts support more than a couple of scripts.

Fonts that have glyphs for all supported scripts and for all Unicode characters are very rare. The font "Arial Unicode MS" is one of the most complete proportional fonts for Windows, and yet it does not

contain all the glyphs associated with all Unicode code points. The “Arial Unicode MS” proportional font contains support for all LN Locales except for the Vietnamese Locale. This font is delivered with the Microsoft Office 2000 product and later versions. This font is particularly useful when the LN backend is running in Unicode mode and contains textual data from a variety of scripts.

For viewing and printing LN reports, a fixed width font is used. There is however, no fixed width font available which contains glyphs for a wide range of Unicode characters. Therefore viewing and printing LN reports which contain a mix of characters from various scripts might be problematic.

Note: With Infor BW no specific font files will be delivered.

The font selection mechanism used in Infor BW and BWPRINT is based upon the following properties:

- Typeface name (the name of the font)
- The backend locale of the LN user. This locale is set in the User Data Template (ttams1110m000) session. It is advised to set this locale to match with the primary language of the user (so in case of Traditional Chinese, Language code o, set the locale to BIG5_XXX).

When the font indicated by the typeface name does not correspond with the backend locale of the user, Infor BW and BWPRINT will select an alternative font which does support the backend locale. (so backend locale has precedence over typeface name).

Infor BW uses the following rules for selecting the typeface name of a font for a specific UI object:

- For most UI controls, BW will use the font which is linked to menus in the Windows display settings. This font can also be changed through the BW Configuration dialog, Tab Font, property: "Standard Baan Windows Font".
- The font for the title bar of all main windows will follow the active title bar setting in the Windows display settings.
- The font for the items in the tree control follows the Windows setting for “icon”.
- The font used for the BW message box follows the Windows setting for “Message Box”.
- The font used for displaying tooltip and status bar messages in BW follows the Windows setting for “tooltip”.

For character windows and the LN text editor, two fixed width fonts are configured in the BW configuration dialog, Tab Font. One for 80 columns windows and one for wider windows. The BW character window is mostly used when previewing a report (send report to display).

To use multibyte character sets

To print multibyte characters

To print multibyte characters, make sure the following conditions are met:

- A device created in the Device Data (ttaad3500m000) session must be linked to a TSS locale that contains the definition of the character set in question. If the proper locale is not specified, the printer daemon cannot print the report.
- The printer must be able to print the characters from the TSS locale. Check your printer manuals.

Utilities

This section describes the available programs that belong to Enterprise Server. These programs are present in the $BSE/bin directory.

tsscvt

Name

tsscvt

TSS conversion filter.

Synopsis

tsscvt [–vV] [–dow] [–l *locale*]

Description

This program converts multibyte characters strings into TSS strings or vice versa. The input/output character set can be set using the **–l** option, but is read from the user file (locale resource) if not given.

If conversion from/to UTF-8 is needed specify –l utf-8.

Use the **–o** option to convert from TSS to the native character set.

The available options are the following:

- **-vV**: Show version and porting information.
- **-d**: Show some additional debug information.
- **-w**: Do not accept any conversion warnings, but instead accept exit(1).

Return values

- 0 = success
- > 0 = failure

tssinfo

Name

tssinfo

This program offers information about the current settings such as your TSS locale, NLS locale, character set information, and so on.

Synopsis

tssinfo [-vV] [-sf] [-l *locale*]

Description

The available options are:

- **-vV**: Show version and porting information.
- **-s**: Show character set definition of TSS locale.
- **-f**: Show font assignments.
- **-l**: Show information about the given locale. This information is read from the user file if not given.

Example

```
Command: tssinfo -sf
 User               : bsp
 TSS locale         : KANJIEUC_AIX32
 TSS character set  : KANJIEUC
 NLS locale         : ja_JP
 Factor             : 1
Multibyte strings:
 Database min.      : 0x1
 Database max.      : 0x9b217e7e
 Forms min.         : 0x20
 Forms max.         : 0x9b217e7e
Single byte strings:
 Database min.      : 0x1
 Database max.      : 0x7f
 Forms min.         : 0x20
 Forms max.         : 0x7a
```

Character set definition for KANJIEUC

NATIVE		TSS	
From	**To**	**From**	**To**
1	7f	1	7f
8ea1	8edf	9b2321a1	9b2321df

NATIVE		TSS	
A1a1	Fefe	9b212121	9b217e7e

Audit Management

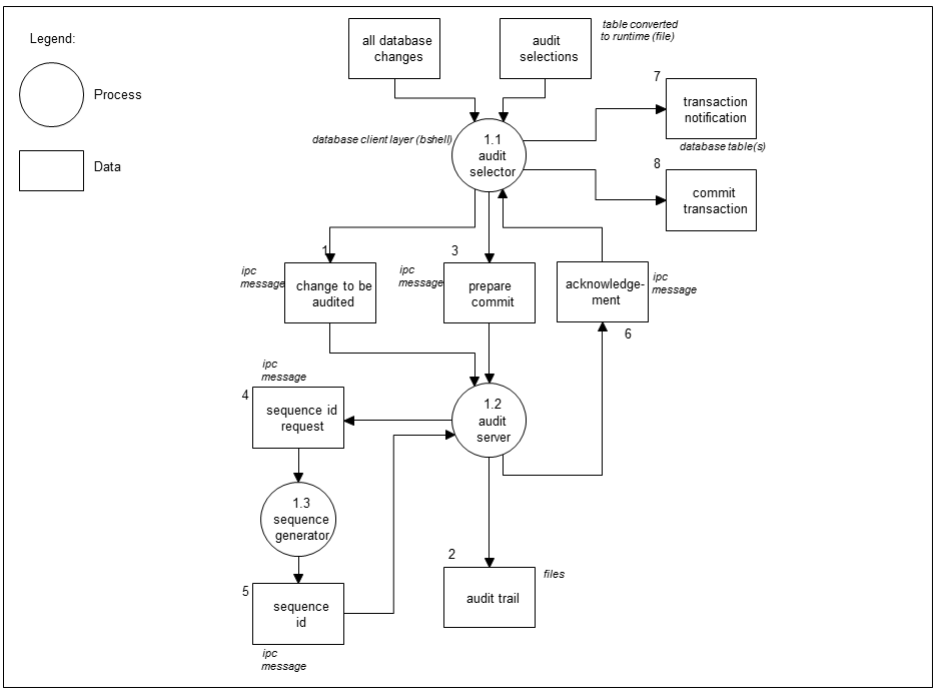
11

General

This chapter introduces the audit management facility provided with Enterprise Server. This chapter also describes the audit server, the audit management utility, and where audit data is stored, and describes the structure of the audit files and how to give users permissions to print, maintain, or clean audit data.

Architecture

The following diagram shows the audit selector and audit server in a process model:



Auditing process

The process can be described as follows:

The audit selector, which is the database client layer of the bshell, checks whether each change on the source database meets the audit selections. If so, the audit selector sends a change to be audited (ipc message) to the audit server.

The audit server writes each change it receives to the audit trail (files). The audit server writes all changes for a transaction and then prepares the commit. The audit server requests a transaction ID from the sequence generator. Because the audit server requests a sequence ID, all audit servers must be running on the same application server.

After receiving the transaction ID, the audit server returns an acknowledgement to the audit selector, including the transaction ID and information on where the data is stored in the audit trail.

The audit selector waits until the audit server has logged all changes and acknowledged the prepare commit. Next, the audit server creates a transaction notification (*database insert*). The user transaction and the transaction notification are committed within the same transaction scope.

A transaction notification consists of the following:

- The transaction ID
- User name
- Session name
- The commit time of the transaction
- A list of tables updated by the transaction
- For each table: An index to where the first log data of the first database action on that table is stored.

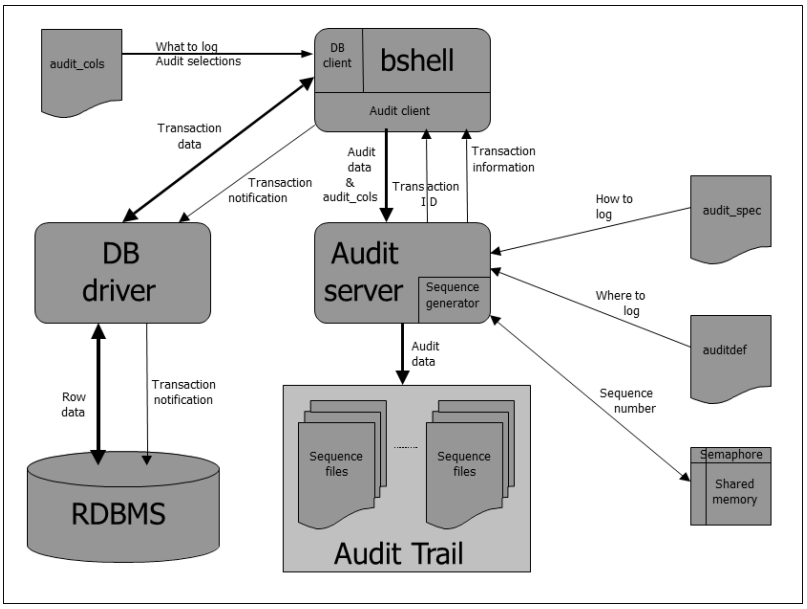
The transaction notifications are physically stored in a relational database table.

By default, audit notifications are stored in the Tools company. If the transaction involved updates to more than one table or company, the audit selector creates more than one transaction notification row.

The following figure provides a technical overview of the various components.

Note: Prior releases used the tabledef6.2 file, the table data dictionary files and the audit_set file to configure the audit setup. This setup is still supported by the Enterprise Server 7 release for backward compatibility. For a description of this former setup, refer to the *Infor Baan 5.2a Tools Technical Manual.*

To use this former setup, the directory $BSE/lib/ will not contain the audit_cols file:



The DB-client part of the bshell reads the audit_cols file with audit selections. The DB-client part sends the contents of the audit_cols file to the audit server. The audit server reads the auditdef and audit_spec file.

The audit_cols file specifies whether a column of a table must be audited. The audit_spec file specifies how the audit trail is organized and handles permission issues. The auditdef file specifies where the audit server must store the audit information.

The bshell sends the database actions or transaction data to the database driver. If a database action meets the audit selection criteria, the action is also sent to the audit server.

At commit time, the audit server writes the audit-data to the audit trail and returns the transaction information and a transaction ID to the audit client.

To create this transaction ID, the sequence generator creates a sequence number. Then the bshell writes a transaction notification within the user transaction scope.

Storage of audit-data

For each table in each company number, the audit server uses a set of files to store audit-data. To read audit files, you can use the 4GL functions or the dictionary programs. This section describes the file types that are used.

Sequence files

The audit-data is stored in a set of files called sequence files. Multiple sequence files can exist for each table and for each company number.

In addition to storing audit-data and transaction information, a sequence file also stores information about audited columns of the table and the audit-data dictionary.

The name of the sequence file is built using the table name and company number. The format of the sequence file name, where the prefix **a** indicates an audit file, just as **t** indicates a table itself, is as follows:

```
a<mmmnnn><company number>.<sequence file number>
```

- mmm represents the module code
- nnn represents the file number

For example, for table ttadv999 and company 000, the sequence files are:

```
aadv999000.000
aadv999000.001
aadv999000.002
aadv999000.003
..........
```

For detailed information on the sequence file, refer to the section "Format of audit files" on page 153.

Information file

One information file exists for each table and for each company number, which does the following:

- Stores information about all sequence files used for this table/company number combination.
- Contains controlling information used to create and maintain sequence files.

The name of the information file is based on the table name and the company number. The format of the information file name is as follows, where mmm represents the module code and nnn represents the file number:

```
a<mmmnnn><company number>.inf
```

For example, for table ttadv999 and company 0, the information file is aadv999000.inf.

Note: One set of files exists for each table name/company number combination. As a result, any associated information, sequence file, or operation is identified by the table name/company number combination. For the sake of readability, this document uses the word table to refer to a table name/company number combination.

Location of audit files

The $BSE/lib/auditdef6.X file is used to specify the location of the sequence and information files for the tables. Under the directory specified in auditdef6.X, a subdirectory is created for each module. Audit files for all tables in this module, for all company numbers, are stored in this directory, unless another directory is specified for other company numbers. The format of the directory name is the following:

```
a<Package Code><Module Code>
```

The following example shows a sample entry in auditdef6.X:

```
ttadv:*:/usr/adv/AUDIT
*:*:/usr1/AUDIT
```

In this example, the audit files for all ttadv tables are stored in the /usr/adv/AUDIT/attadv directory. For all other tables, respective directories are created under /usr1/AUDIT.

For example, **atfacr** is created for the LN Accounts Receivable (ACR) module, **atiitm** for the LN Item Control (ITM) module, and so on.

An entry such as the following can be present:

```
ttadv:*:/usr/adv/AUDIT/#
```

In this case, all ttadv tables are stored under specific respective company numbers. For example, for company number 000, the files are stored in the /usr/adv/AUDIT/000/attadv directory.

Other parameters of audit files

The location of the audit files is controlled as described in the previous section, "Location of audit files" on page 137.

The user can control other parameters, as defined in a file named $BSE/lib/audit_spec. For more information on this file, and specifications of parameters in this file, refer to "Format of audit files" on page 153.

Information file

The tables for which audit trail is enabled can contain sensitive data such as company financial information. As a result, the audit files also contain sensitive data. Audit management contains a security mechanism by which access to the audit-data can be controlled according to user requirement.

A security level is defined for each table/company number combination. Depending on the security level, users either can or cannot access or modify the audit data. The audit server reads the security level from the audit_spec file and stores the security level in the information file of the table.

The central audit management utility, as described in the following section, "Audit management" on page 138, uses this security level to control audit-data access and modification.

Sequence file

The start sequence number for a table is fixed and always 000. The end sequence number for a table is specified in the audit_spec file. For example, if the end sequence number for the ppbdb000 table in company 000 is 999, the table has the following sequence files:

```
abdb000000.000
abdb000000.001
....
abdb000000.998
abdb000000.999
```

For the same table, if the end sequence number is 50, the files are the following:

```
abdb000100.000
abdb000100.001
abdb000100.002
....
abdb000100.049
abdb000100.050
```

In principle, all audit-data for a table/company number combination can be written to a single sequence file. However, because this file becomes too large to manage properly, the audit-data is split across multiple sequence files.

The maximum size of a sequence file for a table is defined in the Audit_spec file. If the limit is exceeded, the audit server tries to use the next sequence file.

Audit management

The audit management version 6.2 incorporates the following features in Enterprise Server audit management utility:

- Audit-data is stored in a set of files, instead of in a single file. Multiple files enable better management than a single large file.
- Each transaction is identified by a transaction header, which logs useful information about the transaction, such as the corresponding user and time. In addition, the data is organized for each transaction. Each transaction has a header followed by all audit-data, which enables you to easily track the changes that users have made.
- When the audit-data is being written, information of audited columns in tables is also stored in audit files. If the table data dictionary is later changed, you can still interpret old audit-data by using the stored column information.

- Table level commands affecting table data such as create table, clear table, and drop table, are also logged by the audit server.
- Audit management such as printing audit files and display audit sequences is done by using the sessions that belong to the Audit Management business object.
- The transaction notification server sends a message to the audit file, which indicates when a transaction has successfully been completed.

Table data dictionary as seen by audit server

By means of the Audit_cols file, the audit trail can be enabled for any column of any table. The audit trail enables a user to audit a single column or all the columns of the table.

The audit server is concerned only with audited columns of a table. The audit server logs the information only for the audited columns. If a column is not audited, no information is logged for that column. If the table has no audited columns, no information is logged for the table. The audit_cols file specifies the columns of the tables for which audit is enabled.

For the rest of this chapter, all columns are assumed to be audited, unless otherwise stated.

The audit-data dictionary (the information of audited columns) is stored in the sequence header. This information is stored in a particular order of table columns. First, information for all audited primary key columns is stored in the order in which they form the primary key. Next, information of remaining audited columns is stored in the order in which the columns occur in the table.

For example, suppose a table has columns c1, c2, c3, c4, c5, c6 with primary key (c4, c2) and all columns except c5 are audited. The audit-data dictionary contains information for the columns in this order: c4, c2, c1, c3, c6. For each column, the audit-data dictionary stores the column name without the table name prefix, field type, size, and depth. The format of a column entry in the audit-data dictionary is as follows:

Name	Type	Depth	Size

Name

Stores column name without table name prefix.

Type

Single byte that indicates the field type. The defines in BDB layer for field type are used, because 4GL also uses the same values. For example, for the CHAR field, the BDB type is BDB_CHAR, and the 4GL counterpart DB.CHAR has the same define as BDB_CHAR.

Depth

Single byte that indicates the depth of the field.

Size

2 bytes indicate the field size. For array fields, this size is the total size of all fields in array.

Commands logged by audit server

The audit server tracks all commands that affect the table data, that is, **Insert**, **Update**, and **Delete** commands. It also tracks certain table level commands such as **Create Table**, **Drop Table**, and **Clear Table**, which affect all rows in a table.

For each of these commands, the audit server places particular information in the audit files. This information can be considered as Audit Row in the audit file.

The audit server does not track select commands, with or without lock, or other table-level commands such as **Change Order**, **Count Rows**, **Create Index**, or **Drop Index**, because these commands do not change the table data in any way.

Data stored by audit server

Application data

In accordance with the application user requirement, the audit server reserves an extra four bytes in each audit row, which application programs can use. This 4byte space is called application data. The audit server does not interpret the application data in any way.

Table operations

For table operations such as create table, drop table, and clear table, the audit server stores only a one-byte indicator. If a non-empty table is dropped or cleared, a delete row operation is created for all rows in the table.

Row operations

For row operations, the audit server keeps the values of audited fields. In addition to field values, the audit server also stores a one-character code that indicates the type of row operation.

Insert actions

The new values of all audited fields to be inserted are stored in the audit file.

Delete actions

The field values of the record to be deleted are stored in the audit file.

Update actions

You can set up a configuration to store only the primary key columns and those values that have been changed, or include columns of a record that have not been changed. You use the Audit_cols file to specify whether a column must be logged only when the column's value changes, or if any other column in the table changes.

Format of audit row

The format of the audit row in the sequence file is as follows:

Length	Appl Data	Action	Data...

Length

A two-byte field that indicates the length of this audit row, excluding this field itself.

Appl Data

A four-byte field reserved for application program usage. When writing a row, the audit server initializes this field to zeroes.

Action

A one-byte character code that indicates the action carried out on the table. The following codes are available:

- **-C**: Create table
- **-R**: Drop table
- **-L**: Clear table
- **-I**: Insert row
- **-D**: Delete row
- **-U**: Update row

Data

The previous fields fulfill the audit requirement for table-level actions. No **Data** field is included for table-level actions.

For row-level actions, the important values of audited fields are stored as data, as described in the following sections.

Insert/Delete actions

Values of all audited fields are stored for insert and delete actions. These values are stored in the same field order as that of the audit-data dictionary of a sequence header. If the audit-data dictionary of a table has columns in the order c4, c2, c1, c3, c6, the values are stored in the same order.

Update actions

For update actions, the field number precedes the field values in the audit row. Field numbers used here are related to the sequence of fields in the audit-data dictionary and not to the original table data dictionary. Suppose a table has columns c1, c2, c3, c4, c5, and c6, with primary key (c4, c2), and all except c5 are audited. The audit-data dictionary contains information for these columns in order c4, c2, c1, c3, and c6.

Therefore, the field number 0 refers to field c4 and not to field c1. Similarly, field 3 refers to c1 and not to c3. For update actions, an audit row looks as follows:

Length	Appl Data	U	Multiple Field Entries

A single field entry is as follows:

Field Number	Changed?	Old Value	New Value

Field Number

A two-byte entry that indicates the field in the audit-data dictionary.

Changed

A single-byte flag that enables you to indicate whether or not the field is updated. The field value for primary key fields must be stored, whether the fields are changed or not. If you change the field value, the corresponding flag is set to **Y**. Otherwise, the flag is set to **N**. The values for all other audited fields are stored in the audit row only if the fields are changed, therefore, for these fields, the flag is always set to **Y**.

Old Value

The old value of the field that is being updated. For a primary key column that is not changed, this value is the only value stored.

New Value

If an audited field value is changed, this represents the new value. For primary key audited fields that are not changed, this field is not present in the audit row.

Using this logic, you can do the following:

The primary key field (audited) is not changed. In this case, the field entry is:

Field number	N	Old Value

The primary key field or any other audited fields are changed. In this case, the field entry is as follows:

Field number	Y	Old Value	New Value

Here, the new value follows the old value.

Examples of the audit row for various operations

Assume you have a table with the columns c1, c2, c3, c4, c5, and c6 with primary key (c4, c2). If all columns except c5 are audited, the audit-data dictionary contains information for these columns in order c4, c2, c1, c3, and c6. The field number 0 then refers to field c4, field number 1 is c2, and so on.

Create table

Length	Appl Data	'C'

Here, the length is the same for all tables: 5.

Drop table

Length	Appl Data	'R'

Here, the length is the same for all tables: 5.

Clear table

Length	Appl Data	'L'

Here, the length is the same for all tables: 5.

Insert row

Length	Appl Data	‘I’	c4	c2	c1	c3	c6

Here, c4, c2, and so on, indicate the values of the corresponding fields. Each field entry size is the size of the field itself.

Delete row

Length	Appl Data	‘D’	c4	c2	c1	c3	c6

Update row

Assume that c2 (primary key column) and c6 are changed.

Length	Appl	'U'	0	N	c4.o	1	Y	c2.o	c2.n	4	Y	c6.o	c6.n

Here, the entry for field 0 (c4) is present, although the entry is not changed, because the field is a primary key field. The field's only value stored is c4.o, which is the old value. Because c2 (field 1) and c6 (field 4) are changed, their old and new values are stored as .o and .n, respectively.

Limit on the size of audit data

All audit data generated in a transaction for a table is always placed in a single sequence file. The transaction data is never split up across sequence files. If the current sequence does not have enough space to write the transaction data, that is, if the data size exceeds the maximum sequence size, the audit server tries to write the transaction data in the next sequence file. If the transaction data cannot be accommodated in the entire sequence file, the audit server cancels the transaction. If this occurs, the application programmer must either reduce the scope of the transaction or increase the maximum file size for the sequence file.

Audit server

Audit file security

The resource audit_mask can be used to restrict the permissions on the audit directories and audit files. By setting the resource audit_mask:007 or adding AUDIT_MASK=007 in the environment of the audit server, all audit files get permission 0660 and all audit directories get permission 0770.

The audit-mask can be set via bse_vars.

Long transactions and overflow file

The audit server buffers audit-data in its memory buffers until the end of the transaction. For example, if during a transaction, 100 rows are inserted in a table and then a commit is carried out, all the audit rows for the insert are buffered by the audit server. This data is written to the audit files only at the time of commit.

The audit server uses individual memory buffers for each table. The size of the buffers is set to enable them to hold large amounts of data. However, the memory buffers might not be able to contain the results of a very large transaction. In this case, the audit server uses an overflow file to buffer the remaining data.

The overflow file is created in $BSE/tmp. One overflow file exists for each session and for each server. The overflow file's name consists of the session ID and process ID (pid) of the audit server, as follows:

```
ao.<Session Id>.<Process Id of the Audit Server>
```

A single overflow file is used to buffer data for all the tables in the session for which the memory buffers were insufficient. At the end of the transaction, if any table has data in the overflow file, the overflow file is read and placed in the corresponding sequence file. After all data has been read and copied from the overflow file, the overflow file is deleted.

If a transaction is canceled, the overflow file is deleted.

Note: The memory buffers of the audit server can hold a large amount of data. If the audit server still must use the overflow file, the transaction is too large. This can be a flaw in application design or coding. Avoid such transactions whenever possible.

Sequence termination

The audit server uses a set of sequence files to store audit-data for a table. The audit server switches from one sequence file to the next under the following circumstances.

Sequence file maximum size reached

Audit-data generated in a transaction for a table is written to a single sequence file.

If the size of the transaction data exceeds the maximum sequence-file size limit for the current sequence, the audit server marks the current sequence file as terminated, and writes the transaction data to the next sequence file.

Before the audit server switches to the next sequence, the audit server ensures that the maximum possible space is available. If the transaction data for the table still cannot be accommodated, the data cannot be placed in any sequence file. This problem is an error condition and the audit server cancels the transaction. Note that in this case, the current sequence is not terminated. If this error occurs, you can either increase the maximum size of the sequence file or reduce the scope of the transaction.

Table audit-data dictionary changed

If the audit server starts to use a sequence, the audit server writes the audit-data dictionary to the sequence headers of the sequence file and the information file. Audit-data in the sequence file depends on the audit-data dictionary of that sequence. For example, for an insert/delete action, all audited field values are placed in the sequence in which the values occur in the audit-data dictionary. For updates, the field values are preceded by field numbers, which are offsets in the audit-data dictionary. 4GL interface audit-data reading also depends on the audit-data dictionary of the sequence.

If the audit-data dictionary changes after the system writes some audit-data to a sequence file, you cannot continue to use the same sequence. In this case, the audit server stops using the current sequence and marks the sequence closed while the audit-data dictionary has been changed. The audit server then starts to use the next sequence.

Note that, here, the switch is made only if the audit-data dictionary changes. This differs from changing the table data dictionary. Changes to the table-data dictionary do not necessarily imply that the audit-data dictionary is also changed.

The following example illustrates events that lead to changes in the audit-data dictionary.

Suppose you have a table with columns c1, c2, c3, c4, c5, and c6 with primary key (c4, c2) in which all except c5 are audited. The audit-data dictionary contains information for these columns in order c4, c2, c1, c3, and c6.

Consider the following changes:

Number of audited columns changed

If the audit trail status for any field is changed, the audit-data dictionary changes. For example, if audit trail is enabled for c5, the audit-data dictionary must have column information in the order c4, c2, c1, c3, c5, and c6, which differs from the audit-data dictionary of the current sequence, so a sequence change occurs. Similarly, if you disable auditing for c1, the new audit-data dictionary is c4, c2, c3 and c6, which differs from the current audit-data dictionary. As a result, a sequence change occurs.

Order of audited primary key columns changed

Initially the primary key was (c4, c2). If the primary key's definition is changed to (c2, c4), the new audit-data dictionary is c2, c4, c1, c3, and c6. Because this differs from the audit-data dictionary of the current sequence, the audit server closes the current sequence and uses the next sequence.

Order of other audited columns changed

The audit-data dictionary changes if the order changes in which the columns (other than the primary key columns) occur in the table definition. For example, if the table is now defined as c1, c2, c4, c5, c6, c3, the new audit-data dictionary is c4, c2, c1, c6, c3. In this case the sequence order is changed.

Number of audited primary key columns changed

You can change the number of audited primary key columns without affecting any of these conditions. For example, if the primary key definition is changed to (c4, c2, c1), the new audit-data dictionary is the same as the current audit-data dictionary. But, because c1 is now a primary key field, the c1 value must be stored in an update action, even if the value is not changed. In previous updates in the current sequence, this step was not required. To make the sequence file consistent, the audit server switches to the next sequence.

Sequence terminated by user

If current security permits, application users can terminate the current sequence using the central audit management utility (see the Audit Management Business Object). Application programs can also terminate the sequence. In this case, the terminating program sets the sequence end date and time and the sequence status.

Reusing sequence files

Suppose you have a table with start sequence 0 and end sequence 49 for a total of 50 audit files. In the first run, each sequence file is created. After all 50 sequences are created and used, the audit server will try to use sequence 0 again. Rather than overwrite the file, the audit server then sends an error message that says that the file must be deleted before the server can reuse the files. You must run the Purge Audit Files (ttaad4161m000) session to delete the file before the audit server can continue.

Always use the Purge Audit Files (ttaad4261m000) session to delete the sequence files rather than use commands such as **rm** or **del**, because the status of the sequence file is maintained in the sequence header in the information file. If this session deletes the sequence file, the session also sets the status of this sequence in the information file as deleted. If the file is removed by any other means, the status is not set and this results in inconsistency.

To lock a file

When you write the data for a transaction, the audit server can close the current sequence and start using the next sequence. As a result, the audit server can change the current sequence field in the information file header. The audit server also changes the sequence status and sequence end time for the current sequence, and changes the status and start date/time of the next sequence.

Even if the audit server does not switch sequences, the server must change the current offset in the information header and the number of transactions in the sequence header. For this reason, the audit server always changes the information header and the sequence header. In addition, because the sequence header is maintained in the sequence file and information file, both of these must be changed.

At the start of a transaction termination command for each table, and before writing buffered data to the audit file, the audit server locks the information file header and sequence header in the information file and in the sequence file. The audit server puts write locks on, so that no other read or write lock can succeed. After the data for all tables is written, the locks on all tables are released. If an error occurs on any table, the status of transactions for tables already written is changed to Abort and the locks on all tables are released.

Limit on open files

When writing audit-data for a table, the audit server must open the information file and one sequence file for that table. After these files are opened, the files are kept open for subsequent use, even after the end of a transaction. These open files can be used in all subsequent transactions involving this table.

Having so many files open, however, implies that the server can reach the operating systems limit of maximum open files per process. For example, if a transaction involves 15 tables, 30 files are opened at the end of that transaction. If the next transaction has 10 different tables, 20 more files are opened. At the end of the second transaction, 50 files are open, which exceeds the maximum limit.

When this limit is reached, the server closes two open files: the information file and the sequence file for a table. The server uses the least recently used (LRU) criteria to choose the table for which the files are closed.

The audit server uses a data structure called a handler to keep track of the file descriptors for the information and sequence files on each table. When these files are opened, the file descriptors are stored in the handler.

The handlers of all tables are kept in a linked list in an LRU manner. For example, when files are first opened, the handler is allocated and is put in most recently used (MRU) position. As other tables are opened, this handler moves toward the LRU position. Later on, if this table is involved in another transaction, the same handler is reused and is again moved to the MRU position. In addition, when a file is open, all data is written to the file before the data goes to the next table. As a result, until all data is written for a table in the current transaction, the handler for that table is guaranteed to stay in the MRU position and in no way can the handler reach the LRU position.

If the audit server reaches the open file limit when opening a new information file, sequence file, or overflow file, the server picks up the LRU handler in the handler list and closes the handler's information file and sequence file. The assumption is that because the table has not recently been used, this table is the best choice as a victim for closing files.

To open large number of tables in a single session

A session can update a large number of tables. On systems that have a small maximum number of open files, this can create a problem. Suppose a session with a limit of 30 open files starts to update 50 tables.

After opening the information and sequence files for the first 30 tables, the system reaches the maximum limit of open files. If files for table31 are opened, the LRU logic is activated and files for table1 are closed. But table1 is itself in transaction and its files are locked. When the audit server closes the files, the locks are released and other processes can alter the files. Now, the same audit server might again have to operate on the audit files of table1.

For this reason, the audit files for table1 must be reopened. However, if the files were changed in the meantime, the status changes to ABORT_DONE. In this case, the audit server returns an error and cancels the transaction.

Multisession support

The audit server supports multiple sessions, which means that the same audit server is used for multiple sessions within the client.

The audit-data dictionary of a table is kept globally. The same table/company number in multiple sessions shares a single audit-data dictionary.

The handlers, one for each table/company number combination, are kept in global lists. If multiple sessions operate on the same table/company number combination, the sessions share a single handler.

File security for various audit management files

The audit_spec file must have read permission for all users, and that write permission is granted only to the administrator user. Otherwise, users other than the administrator can change the security parameter in the file, which would provide these users with illegal access.

Audit server debugging options

For debugging purposes, a code is placed in the audit server at appropriate stages to print useful information. In normal usage, this printing is not enabled. To enable this printing, you can set an AUDITLOG environment variable. You can also set this variable to assorted values to print various types of information. The values must be set as octals.

- 0001: General information.
- 0002: Print audit rows generated.
- 0004: Print audit-data dictionary for table.
- 0010: Print data structures such as information header, sequence header, and transaction header.

The information is placed in a log file named **audit.log.*pid*** of the server. This file is created in the current directory of the audit server.

To access transaction data

Previous versions of the audit trail files required that transaction data be accessed in sequence. This process was extremely time consuming if the last transaction had to be retrieved from a sequence file.

To speed things up, the audit server now saves the audit file offset internally. The audit server combines the offsets for each audit file, the table name, and the company number, and sends the offsets in a message back to the bshell at the moment of the commit.

The information the audit server sends to the bshell contains the following data:

- Tablename T_1

- Company number C_1
- Sequence number of audit file S_1
- Offset of transaction X_1 in sequence file
- Tablename T_2
- Company number C_2
- Sequence number of audit file S_2
- Offset of transaction X_2 in sequence file

The following table shows an example:

Tablename	Company number	Audit file sequence number	Offset in audit file
ttadv100	000	001	2353
tccom010	812	065	1406
tfgld045	812	123	56028

After the application competes the database update, the data is committed. The bshell requests a transaction ID from the audit server. The audit server generates a sequence number and sends the number back to the bshell together with the transaction information message. The bshell uses this information to create a transaction notification, and sends a commit to the database.

To create a transaction notification

The two-phase commit protocol used in previous versions has been replaced by the transaction notification. This eliminates the in-doubt transactions that can occur if the transaction is prepared but the data is not yet committed or aborted.

The bshell writes this transaction notification into a notification table in the database, within the user transaction. If the transaction commits, the bshell writes the transaction notification, and if the transaction aborts, the bshell does not write the notification. The bshell creates a transaction notification for each transaction that is logged in the audit trail.

The transaction notification contains the transaction ID and information on where to find the transaction information in the audit sequence file. These transaction notifications can be used as a trigger for any process that needs to react to database changes such as a replication process, and enables fast access to the audit trail data.

Audit errors

The audit server uses the following error messages. If you receive one of these error messages, refer to $BSE/log/log.audit for more information on the cause of the error.

```
E_AUD_SETUP 251 The audit setup is not correct.
```

This error can occur if the specification in the $BSE/lib/audit_spec file is not in the proper format.

```
TOSEQ 6 SECURITY 28 MAXSEQSIZE 25K or
```

No table/company number combination.

```
aa:00a:TOSEQ 6 SECURITY 28 MAXSEQSIZE 25K
```

Incorrect company number.

```
*:*:TOSEQ 6 SECURITY 28 MAXSEQSIZE 5K
```

The maximum sequence size limit is less than the specified limits. For details about the format of this file, refer to "Format of audit files" on page 153. This error can also mean that the current sequence file was illegally removed, or that the sequence files were removed without using the 4GL interface. Audit files used during the first run can also cause this error.

```
E_AUD_CORRUPT 252 An audit file is corrupt.
```

Some corruption of the information file has occurred. A mismatch exists in sequence headers in the information file and the sequence file.

```
E_AUD_LOCKED 253 Another user has locked the audit file.
```

Multiple users are trying to use the same audit file at the same time. The audit file is locked before the file can be modified and, as a result, the next user receives an error message.

```
E_AUD_ABORT 254 Commit transaction has failed in audit server.
```

The audit file is changed when the audit-data is written to the audit file. When too many files are opened and the maximum limit is reached, the LRU file is closed. This error can refer to the file used in the transaction. When the file is reopened, the starting date time of the sequence header is checked with memory data structure. If the times do not match, the E_AUD_ABORT error is returned.

Authorizations

To grant audit security permissions to users, the administrator must use the Audit Authorizations (ttaad4562m000) session. These permissions determine whether the user can use the sessions in the Audit Management business object.

Security permissions assigned to users are checked against previously existing records to check for conflicts.

You can assign security permissions for all packages, modules, table numbers, and companies. You can provide some permissions for all packages, and provide no permissions to a range of packages to disable these permissions.

You can use the **Specified** command to provide particular permissions to a specified package. Using **Specified**, the highest priority is given to the security permissions assigned for that particular package, module, table, and company.

You cannot assign security permissions for a range within a range. Suppose you assign some permissions using the **All** command, then disable permissions for a range between tt—vv. You cannot then enable permissions within this range or an overlapping range. However, you can use the **Specified** command to enable the permissions for a specific package.

The conflicting records are not stored in the table.

The security permissions for users are stored in a table ttaad462, which you can access using the sessions in the Audit Management business object.

The security checking is carried out at two layers: one at user level and another for reading audit information. The user can have permissions at the user level but not for reading the audit information. At audit level, the security permissions are stored in the audit information header of the audit information file.

User level permissions	Audit level permissions
None	Print
All	Clean
Print	Maintain
Clean	Application-defined
Maintain	

At user level, the administrator user can grant permissions in various combinations. The administrator user assigns permissions to print and clean audit information through the Audit Authorizations (ttaad4562m000) session on the Enterprise Server Audit Management menu. The only session the administrator user can execute without being assigned any security permissions is the maintain session.

Example

None	No permissions
All	Print, clean, and maintain permissions

Print	Only print permissions
Print/Clean	Print and clean permissions
Print/Clean/Maintain	Same as all permissions
Application defined	Access to some sessions restricted
Maintain/Application	Only maintain permission, no print and clean-defined permissions

Note: None and All are mutually exclusive.

At audit level, the permissions can be granted in combinations except application-defined. Only maintain can be combined with application-defined.

The user cannot print or clean up any audit information. To print, clean, or maintain the audit information from the audit files, the permissions must exist at both levels.

- Permission at user level: Print or print/clean
- Permission at audit level: Application-defined

The user cannot print or clean up any audit information in the audit files using the sessions in the Audit Management Business Object. The user can, however, run the Audit Information File (ttaad4160s000) session, because the user has maintain permission only at audit-file level.

- Permission at user level: Maintain or print/maintain or clean/maintain
- Permission at audit level: Maintain/application-defined

If the user is not the administrator and only application-defined exists at the audit file level, no session of the Audit Management business object can be executed. The administrator user can use the maintain session even if no permissions are assigned to the administrator user. However, the administrator user cannot use print and clean sessions unless security permissions are assigned to the administrator.

If user **bsp** has print permission at user level, a further check is made to see the security permissions at audit level.

User **bsp** can only continue if print permission is available at the audit level. Otherwise, the session is canceled and an appropriate message appears.

Format of audit files

audit_cols

The audit_cols file specifies the table columns that are to be audited. You can specify whether a column a) must be logged only when the column’s value has changed or b) must be logged when any other column of the table has changed.

The audit_cols file contains entries of the following format:

```
<Table Name>:<Company Number>:<A/C>[:<Column name list>]
```

Each entry specifies one or more columns that are to be audited.

The following sections list the meaning of each field.

Table name

This specifies the table names to which this entry applies. The following formats are possible:

- *: Means all tables in all packages.
- Package: Means all tables in this package, for example, tf, td, and so on
- Package and module code: All tables in the module, for example, tccom or ttadv
- Package, module, and table number: For example, ttadv999 or tccom010

Company number

This specifies the company numbers to which this entry applies. The following formats are possible:

- *: All companies.
- 100: Company 100 only.
- 100,000: Company 100 and 000.

A/C

This specifies when the column must be logged:

- A (Always): The column is logged when any audited column in the table has changed.
- C (when Changed): The column is logged only if the column's value has changed.

Column name list

This list is a comma-separated list of the column names to which this entry applies. If the **<*column name list*>** is omitted, this entry applies to all columns of the table. Otherwise, this entry applies to a column only if the column's name is in the **<*column name list*>**.

Only if the **<*table name*>** specifies a full table name, including package, module and table number, a **<*column name list*>** can be supplied.

To determine whether a column of a table in a company is audited, the column is matched against every entry. An entry matches if and only if:

- The table name of the column matches with **<*table name*>**.
- The company matches with **<*company number*>**.
- The name of the column is in **<*column name list*>** or **<*column name list*>** is omitted.

Note: Multiple matching entries can exist for a column. In case of conflicts, for example, if one entry specifies A for a particular column and another entry specifies C, then A takes precedence over C. In

other words, if some entry specifies that a column will be logged always, then no other entry can change this specification to “when changed”.

Primary key columns

Primary key columns are handled differently from other columns. If an entry specifies C (log when changed) for a primary key column then the column is always logged, even if the column’s value is unchanged.

Example

```
dbtst120:*:A:bonus
dbtst120:*:C:salary
dbtst100:000:A
dbtst100:*:C
```

The following statements hold for this example:

- dbtst120.salary is audited only if the column itself has changed.
- dbtst120.bonus is audited only if dbtst120.salary or dbtst120.bonus has changed.
- No other column, including primary key columns, of dbtst120 is audited.
- All columns of dbtst100 in company 000 are audited if any column of dbtst100 has changed.
- Only the changed columns and the primary keys columns of dbtst100 in company 100 are audited if any column has changed.

audit_hosts

The audit_hosts file specifies at a company level the hostname of the audit server for that company. The format of an entry in the audit_hosts file is:

```
<Company Number>:<Hostname>
```

Company Number

The company number can be specified as follows:

- *: All companies.
- 100: Company 100 only.
- 100,000: Company 100 and 000.

Hostname

The hostname, for example, *eddie*, eddie.infor.com or *10.31.112.63*.

Example

```
812:eddie.infor.com
```

If, for a specific company, no applicable entry exists, all data that belongs to that company is audited on the local host.

auditdef6.2

This file is used to specify the directories in which audit files are created. One such file exists for each BSE environment, and is located in $BSE/lib.

This file has the following structure:

```
<Table Name>:<Company Number>:<Directory Name>
```

The following section lists the meaning of each field.

Table Name

Indicates the table and can be in the following formats:

- *: Means all tables in all packages
- Package: Means all tables in this package, for example, tf, td, and so on
- Package and module code: All tables in the module, for example, tccom and ttadv
- Package, module, and table number: For example, ttadv999 and tccom010

Company Number

You can choose tables based on company number, for example:

- *: All companies.
- 100: Company 100 only.
- 100,000: Company 100 and 000.

Directory

Directory in which the audit files are stored. Under this directory, another directory is created, called: ***a*Package + Module Code**

audit_spec file

The audit_spec file is used to specify audit file parameters. One such file exists for each BSE environment, and is located in the $BSE/lib directory.

The format of the file is the following:

```
Table Name:Company Number:Audit Specifications
```

The following list provides the meaning of each field:

Table Name

Indicates the table. The table name can have the following formats:

- *: All tables in all packages
- Package All tables in this package, for example tf, td, and so on
- Package and module code All tables in the module, for example, tccom or ttadv
- Package, module and table number For example, ttadv999 or tccom010

Company

This can be in one of following formats.

- *: All companies
- 100: Single company, for example, company 100 only
- 100,000: Comma-separated list of companies, for example, company 100 and 000

Specifications

The following specifications can be put in the file:

- To sequence:

  This sequence is specified by the keyword **TOSEQ** or **toseq**, followed by a sequence number. Note that the keyword and the sequence number must be separated by a space, for example:

  ```
  TOSEQ 999 or toseq 999
  ```

  If the end sequence is not specified for a table/company combination, the default is 999.

- Maximum sequence file size:

  This size is specified by the keyword **MAXSEQSIZE** or **MAXSEQSIZE** followed by a size specification. Note that the keyword and the size specification must be separated by a space. The size can be specified in:

  - Bytes: For example, **MAXSEQSIZE 10000** (10,000 bytes).
  - Kilobytes: For example, **MAXSEQSIZE 10K** (10KB or 10*1024 bytes).
  - Megabytes, for example, **MAXSEQSIZE 2M** (2MB or 2*1024*1024 bytes).

Note that for kilobyte and megabyte specifications, the characters K and M must be uppercase only and there must not be a space between the number and the specification character.

If the maximum sequence size is not specified for a table/company combination, the default is 512 KB.

Security assigned to current table/company number audit files. The security is specified as keyword **SECURITY** or **security** followed by a security number. Note that the keyword and the security specifications must be separated by a space, for example:

```
SECURITY 4 or security 4
```

The following table lists the various security levels and the relevant codes:

Security	Value
Print	4
Clean	8
Maintain	16
Application defined	32

The combination of these is described in "Authorizations" on page 152.

To specify clean and maintain, security is specified as security 24 (the sum of 8 + 16).

You cannot combine an application-defined with print and clean. You can only combine application-defined with maintain permissions. Note that you can place these specifications in any order.

Note: The audit_spec file must have a default entry covering all tables/company numbers. You can specify this with an asterisk (*) for table and company number fields. This must be the last line of the file.

A sample audit_spec file can look like the following:

```
ttadv999:000:TOSEQ 99 MAXSEQSIZE 2M
tt:000: toseq 49 maxseqsize 1M
tf:000:toseq 49 maxseqsize 20K
*:*: TOSEQ 499 MAXSEQSIZE 100K
```

Note: The audit_spec file must have read permission for all users and write permission for the administrator user only. Otherwise, users other than the administrator can change the security parameter in the file, which would provide these users with illegal access.

Information file

One information file exists for each table for each company number. To read the information file, you can use the 4GL functions or the dictionary programs.

The information file contains the controlling information for the sequence files, called the information header, followed by the sequence headers of all the sequence files created.

The format of the information file name, where *mmm* means module code and *nnn* means table number, is as follows:

a *mmmnnn company number*.inf

For example, the name for table ttadv999, company 0, is aadv999000.inf.

The structure of the information file is a header called information header, followed by multiple sequence headers, one for each sequence created.

Information header

Field	Length (bytes)	Description
Version Information	4 Str	Used by the audit server to store the version information. This must never be modified by any application by any means.
Table Name	8 Str	Table for which audit information is stored.
Company Number	2 Int	Company number of the table.
Status	1 Int	The status specifies whether or not sequence reuse is permitted.
From Sequence Number	2 Int	Audit-data will be stored in sequence files specified from this value.
To Sequence Number	2 Int	The last sequence file after which sequence reuse will be started or the user must take appropriate the action such as deleting the existing file before reuse.
End Sequence Number	2 Int	Specifies the last sequence written in the info file. This field is for internal use by the audit server only. The user must never change this by any means. Otherwise, the audit server will not work correctly.
Max. Size of 1 Audit File in Bytes	4 Int	Specifies the maximum file size of a sequence file.
Security Level	2 Int	Determines the users who have access to the information in the audit files. The security permissions regarding access to audit information are stored here.
Current Sequence Number	2 Int	Determines the current sequence number that will be used to write the audit-data in the sequence file.
Offset in Current Sequence File.	4 Int	Specifies the offset in the current sequence file where the audit-data will be written.
Reserved Space	8	This space is not used at present and is reserved for future use. Application programs must never use this space as this space may be used by a future release of audit server.

Field	Length (bytes)	Description
Length of Following Sequence Header	2 Int	The length of the first sequence header in the info file.

Sequence header

Each sequence file begins with a sequence header. The information file contains copies of sequence headers for all sequences created.

Field	Length (bytes)	Description
Sequence Number	2 Int	Specifies the sequence number of the sequence file.
Creation Date	8 Str	Specifies the creation date, in UTC, of the sequence file in YYYYMMDD format.
Creation Time	6 Str	Specifies the creation time, in UTC, of the sequence file in HHMMSS format.
Termination Date	8 Str	Specifies the termination date, in UTC, of the sequence file in YYYYMMDD format.
Termination Time	6 Str	Specifies the termination time, in UTC, of the sequence file in HHMMSS format.
Status	2 Int	The status provides information to the user on whether the sequence file was user-terminated or was closed because of DD change or because a transaction data could not be accommodated in the sequence file. The possible values are: • 0001: Terminated because of size. • 0002: Table DD changed. • 0004: User forced. • 0010: Sequence and info header mismatch. • 0020: To be set only in info header. Set when the sequence is removed. • 0040: Change in sequence file version. • 0100: Version number present in sequence.
Number of Transaction in the Sequence	2 Int	Specifies the data of the number of transactions present in the sequence file.
Application Info - 1	4	These bytes are reserved for storing data specific to the application. If the application must store any information in the audit files, these fields must be used to store that information.
Application Info - 2	4	
Application Info - 3	4	
Application Info - 4	4	

Field	Length (bytes)	Description
Number of Fields Being Audited	2 Int	Total number of fields being audited.
Number of Fields in Primary Index	1 Int	Field information for all fields being audited is written below. First, the fields that are part of the primary index are written in the same sequence in which the fields appear in the primary index. Next, all remaining fields are written in the order in which they occur in table DD. For example, if fields f1, f2, f3, and f4 are being audited and the primary index is on f4, f2, then field information will be written first for f4, and then for f2. Next the information will be written for all remaining fields. This field indicates the number of fields in the primary key. For example, for the previous example, the following field value must be 2 and the field layout will be f4, f2, f1, f3.
<Field Data 1>		
…		
<Field Data n>		n = number of fields being audited
Sequence Version	4 Str	Change note: In Infor Baan IVc and earlier releases, the sequence version is not available. In that case, these four bytes are reserved space.
Audit_cols configuration ID	4	Identifies the audit_cols configuration that was used to create this sequence file.
Length of Following Sequence Header	2 Int	The length of the next sequence header in the info file.

Field data

Field	Length (bytes)	Description
Field Name	8 Str	Name of the audited field
Field Type	1 Int	Type of the field
Field Depth	1 Int	Array depth of field
Field Length	2 Int	Length of the field

Sequence file

Multiple sequence files are used to store audit-data for each table/company number combination. You can read the sequence files by using the 4GL functions or the dictionary programs. The format of the sequence file name, where mmm refers to the module code and nnn means table number, is as follows:

a *mmmnnn company number.sequence file number*

For example, these sequence files can exist for table ttadv999, company number 0:

```
aadv999000.000
aadv999000.001
aadv999000.002
aadv999000.003
....
```

The sequence file contains the table name and company number, followed by the sequence header, followed by transaction data of multiple transactions. Transaction data is followed by a transaction header and contains multiple audit rows generated for the transaction.

Sequence file layout

Field	Length (bytes)	Description
Table name	8 Str	Name of the table
Company number	2 Int	Company number of the table
Length of sequence header	2 Int	The length of the next sequence header.
<sequence header>		
<transaction header 1>		
<transaction data 1>		
…		
<transaction header n>		n = number of transactions in the sequence file (specified in the transaction header)
<transaction data n>		

Transaction header format

You must log both the operating system user ID and the Enterprise Server user name, because a user can change the USER environment variable and use different Enterprise Server user names. In addition, different Enterprise Server users can have different table privileges in SQL databases. Programs can find the operating system logon name by using the operating system user ID **print**.

Example

```
Operating system user ID        4
Baan User Name                  8
Date (YYYYMMDD)                 8
Time (HHMMSS)                   6
Session Name                   15
```

The transaction status can have one of the following values:

- COMMIT_DONE
- ABORT_DONE
- GATI_UTC_FLDS

From Infor Baan IVc onward, the status can also have the following flags. You can find the values for these flags in the audit include file.

Note: The audit server no longer writes the PREPARE_DONE status.

```
Transaction Status                            1
Number of audit rows in the transaction       2
```

In addition, as of Infor Baan IVc, the following fields are added at this position:

```
{
GATI 0                                       4
GATI 1                                       4
Commit Date (YYYYMMDD)                       8
Commit Time (HHMMSS)                         6
Reserved space                               8
}
Total bytes of data of this transaction      4
```

You must log both the operating system user ID and the Enterprise Server user name because a user can change the USER environment variable and use different Enterprise Server user names. In addition, different Enterprise Server users can have different table privileges in SQL databases. Programs can use the operating system user ID **print** to find the operating system logon name.

Field	Length (bytes)	Description
UNIX User Id	4 Int	ID of the user on UNIX. On NT is field is 0.
Baan User Name	8 Str	You must log both OS user ID and LN user name, because an OS user can change the USER environment variable to operate using different LN user names. Different Triton users can have various table privileges in SQL databases. Therefore, you must store both. Using an OS user ID, print programs can find the UNIX logon name, and so on.
Old Commit Date/Time	14 Str	Historical commit time in UTC. Format: YYYYMMDDhhmmss
Commit Date/Time	14 Str	Commit time of transaction in UTC. Format: YYYYMMDDhh-mmss. This Commit Date/Time is equal over all audit tables that are involved in this transaction. Change note: This field is only available from Infor Baan IVc onwards.
Session Name	15 Str	Name of the session in which the transaction was carried out.
Transaction Status	1 Int	This field can have one of following values: COMMIT_DONE, ABORT_DONE. From Enterprise Server onwards, the transaction status will be irrelevant, because the status is determined in the transaction notifications.

Field	Length (bytes)	Description
Number of Audit Rows in the Transaction	2 Int	Number of audit rows for this table, so not all audit rows of the complete transaction.
Transaction ID 0	4 Int	Change note: This field is only available from Infor Baan IVc onwards.
Transaction ID 1	4 Int	Change note: This field is only available from Infor Baan IVc onwards.
Reserved Space	8	Change note: This field is only available from Infor Baan IVc onwards.
Total Bytes in Transaction	4 Int	Total bytes of data of this transaction.

Transaction data format

Field	Length (bytes)	Description
Length of Row	2 Int	Length of the audit row.
SNAT	4 Int	Sequence Number in Audit Transaction.
Application Data	4	In every audit row, 4 bytes of space are kept to store application data. This data is stored and manipulated by 4GL applications and not by the audit server. The audit server just keeps the 4 bytes of extra space to be used later.
Type of Operation	1 Str	Type of operation is a single character field, which indicates the database action performed on the table. These characters are used to indicate audited database actions: • C: Create Table. • L: Clear Table. • R: Drop (Remove) Table. • I: Insert row. • D: Delete row. • U: Update row.
Row Data	...	In the audit row, the row data is present only for row-level actions. The field values, whenever stored, are stored in the same format as their data type. Therefore, the value for a long field is also stored as long in the audit row. As a result, the length of each field entry in the audit row is the same as the length of the column. For insert, the row data consists of field values, that is, the values to be inserted, of all audited fields, in the order in which they occur in the audit DD. For delete, the row data consist of field values of the row to be deleted of all audited fields, in the order in which the values occur in audit DD.

Field	Length (bytes)	Description
		In case of update, not all fields in the table can be updated. Yet for all fields, also those that have not been changed, the old and new values are stored. Field numbers are used to identify the fields. A field number for a field is its position in the audit DD, starting with field 0. Change note: In versions prior to Infor Baan 5.0c the following holds for updates: In case of update, not all fields in the table can be updated. Therefore, apart from storing field values, an identifier is required to specify the field to which this data belongs. Here, field numbers are used to identify the fields. A field number for a field is its position in the audit DD (starting with field 0). An entry for a field looks as follows: • Field Number: 2 Bytes. • Is the value changed: 1 Byte. (Y or N). • Old Value: … • New Value: … Here, if the field is changed, both old and new values are stored. If a primary audited key field is not changed, only a single value is stored. For other audited fields, if the fields are not changed, nothing is stored. For all primary key audited fields and all other audited fields that are changed, these entries are made to make the audit-data. For more information on audit rows and examples, refer to "Audit server" on page 144

- Transaction status:
  Length: 1 byte. Transaction status can have the following values:
  - COMMIT_DONE
  - ABORT_DONE
- Number of audit rows in the transaction:
  Length: 2 bytes.
- Total bytes of data of this transaction:
  Length: 4 bytes.

OLE Automation

12

General

OLE Automation is an industry standard that applications use to expose their OLE objects to development tools, macro languages, and other applications that support OLE Automation. For example, a spreadsheet application can expose a worksheet, chart, cell, or range of cells, all as different types of objects. A word processor can expose objects such as an application, document, paragraph, sentence, or selection.

If an application supports OLE Automation, you can use Visual Basic to access the objects the application exposes. To use Visual Basic to manipulate these objects, you can invoke methods on the object or get or set the object's properties. For example, if you create an OLE Automation object named MyObj, you can write the following code to manipulate the object:

```
Dim MyObj As Object 'declaration
Sub PrintWordDoc()
Set MyObj = CreateObject("Word.Basic") 'start MS Word
 MyObj.FileNewDefault 'open a new file
 MyObj.Bold 1
 MyObj.Insert "Hello, world." 'insert new text
 MyObj.FilePrint  'print current file
 MyObj.FileSaveAs "C:\WORDPROCDOCS\TESTOBJ.DOC"
 MyObj.AppClose  'close MS Word
 Set MyObj = Nothing
 Exit Sub
End Sub
```

This example runs Microsoft Word to create and print a document with the text, "Hello, world."

The application that exposes objects is called an OLE Automation server, the application using those objects is called an OLE Automation client.

In fact, the OLE Automation interface can be compared to the C Interface of Enterprise Server. Where the C Interface offers an interface from other C programs to Enterprise Server, OLE Automation offers an interface from other Microsoft Windows applications to Enterprise Server. The implementation is based on the Enterprise Server 4GL **parse_and_exec_function** function in the same way as the C Interface.

Enterprise Server as OLE Automation server

In combination with the BW user interface, Enterprise Server is also an OLE Automation server. Enterprise Server exposes an object through which clients can call Enterprise Server dynamic link library (DLL) functions. This section describes how you can create and use this Enterprise Server object.

Enterprise Server exposes just one object, which is the application object. In Visual Basic this object can be created with:

```
Set BaanObj = CreateObject("Baan.Application")
```

This starts Enterprise Server with the BW user interface invisibly, in other words, only the Option Dialog icon, is displayed. OLE Automation objects, such as BaanObj, can have methods and properties. A method is a function that you can start to perform a particular action. A property is an attribute that has a value. Property values can be set or retrieved.

The Enterprise Server application object has the following methods:

- dllname, functioncall ParseExecFunction
- Quit

The Enterprise Server application object has the following properties:

Name	Type	Value can be
Timeout	long	Set
FunctionCall	string	Set
Error	long	Retrieved
ReturnValue	string	Retrieved
ReturnCall	string	Retrieved
Binary	boolean	Set

These methods and properties enable Visual Basic programmers to call any function in any DLL, for example:

```
BaanObj.Timeout = 10 ' 10 seconds timeout
BaanObj.ParseExecFunction "mydll", "myfunction(arg1, arg2)"
If BaanObj.Error <> 0 Then MsgBox("An error occurred")
MsgBox("The return value is " + BaanObj.ReturnValue)
```

This calls the **myfunction** function in the DLL named mydll. Note how arguments are passed to **myfunction**. If arguments are passed by reference, the arguments can be changed by the function call. Therefore, the ReturnCall property contains the modified arguments after the call.

The Timeout property is used to prevent blocking on erroneous calls. The Error property contains the return status of the ParseExecFunction call, and the ReturnValue property is the actual return value

of the DLL function. If the DLL function returns a long value, the value is converted to a string in ReturnValue.

To return binary data in ReturnCall arguments, the binary property must be set to **True**. ReturnCall can contain null characters.

The possible values for the Error property are:

- 0: Success
- -1: Unknown DLL name
- -2: Unknown function name
- -3: Syntax error in function call
- -10: Timer expired
- -11: Fatal error in function

To end an automation session, you can quit an Enterprise Server application, as in the following example:

```
Set BaanObj = Nothing
```

Restrictions

If you use the OLE Automation interface, you must consider the following restrictions:

- Due to the 3GL **parse_and_exec_function** function, you cannot supply arrays as arguments to a DLL function.
- DLL functions with a variable number of arguments or optional arguments are not supported.
- If a string is passed by reference, the result cannot become longer than the input string. When a string must be filled make sure the string is initialized with the same number of spaces as the maximum length that can be returned. The maximum length is defined in the domain of the variable.
- The maximum size of the FunctionCall string is 4KB (bshell limit).
- The maximum size of the ReturnValue string is 4KB (bshell limit).
- Two client applications cannot simultaneously use the same BW to make DLL function calls.

Example: import Enterprise Server users

This example fills a Microsoft Excel spreadsheet with the names of all Enterprise Server users. To do this, you must do the following:

- Implement DLL functions to obtain the names of the users from the database.
- Implement some Visual Basic code in Microsoft Excel.

DLL function example

The following DLL, called demo.dll, contains two functions that retrieve the names of the users from the database. One function initiates the query and receives the first name, and the other receives the next name. If the last record has been returned, an empty string is returned.

```
table tttaad200
long sql_id
function extern string get_first_user()
{
string sql_query(512)
string user(512)

switch.to.company(0)
sql_query = "select ttaad200.user from ttaad200"
sql_id = sql.parse(sql_query)
sql.exec(sql_id)
e = sql.fetch(sql_id)

if e = 0 then
       user = ttaad200.user
else
       user = ""
endif

return(user)
}
function extern string get_next_user()
{
string user(512)

e = sql.fetch(sql_id)
if e = 0 then
       user = ttaad200.user
else
       user = ""
endif

return(user)
}
```

Visual Basic example

Microsoft Excel contains Visual Basic for Applications (VBA), which enables scripting to control both the Excel application and other applications through OLE Automation. The following is the Visual Basic source that calls the DLL functions and displays the results in a spreadsheet. The subroutine

GetBaanERPUsers is a Microsoft Excel macro that can be invoked through buttons or menus on the spreadsheet:

```
Dim user As String
Dim BaanObject As Object

Sub GetBaanERPUsers()
On Error GoTo CannotConnectToBaanERP

' run Infor Enterprise Server
Set BaanObject = CreateObject("Baan.Application")

On Error GoTo BaanAutomationError

' get first user
BaanObject.ParseExecFunction "demodll",  "get_first_user()"
If (BaanObject.Error <> 0) Then GoTo  BaanAutomationError
user = BaanObject.ReturnValue

Row = 1
Column = 1
While (user <> "")
' fill the spreadsheet
Worksheets("Main Sheet").Cells(Row, Column)
= user
Row = Row + 1
If Row > 15 Then
Row = 1
Column = Column + 2
End If
' get next user
BaanObject.ParseExecFunction "demodll"
"get_next_user()"
If (BaanObject.Error <> 0) Then GoTo     BaanAutomationError
user = BaanObject.ReturnValue
Wend

' exit Infor Enterprise Server
Set BaanObject = Nothing
Exit Sub

' error handling
CannotConnectToBaanERP:
MsgBox "Unable to connect to Infor Enterprise Server"
Exit Sub

BaanAutomationError:
MsgBox "An Infor Enterprise Server automation error occurred"
Set BaanObject = Nothing
Exit Sub

End Sub
```

Example: To use Enterprise Server SQL

The following example fills a Microsoft Excel spreadsheet using Enterprise Server SQL. To implement this, a DLL called ottdllsql_query has been developed to parse and execute the query. To receive more information about this library, enter the following command:

```
*> bic_info -eu ottdllsql_query
```

DLL information

The DLL ottdllsql_query contains:

- Functions to parse, execute, and stop a query:
  - olesql_parse
  - olesql_fetch
  - olesql_break
  - olesql_close
- Functions to import query results into Visual Basic:
  - olesql_getstring
  - olesql_getint
- A function to convert an Easy SQL query to a string:
  - easysql_to_olesql
- Declaration of external variables, such as:
  - olesql_count
  - olesql_float0, olesql_float1 ... olesql_float9 (double variables)
  - olesql_int0, olesql_int1 ... olesql_int9 (long variables)

A Microsoft Excel example is installed in the SAMPLES directory of BW (BAAN.XLS). This spreadsheet contains several macros.

The first macro, Enterprise Server, shows a number of Enterprise Server users on the Worksheet Users spreadsheet. To use macro, click the button on the Worksheet Users that starts this macro. For more information, refer to the Visual Basic code in Worksheet Module 1.

The second macro, Enterprise Server, shows all item groups in Enterprise Server with the numbers of items for each item group on the Worksheet Itemcount spreadsheet. Click the button on the Worksheet Itemcount to start this macro. To run the macro you must insert an Easy SQL query in the Enterprise

Server application. To insert the query, start the Query Data (ttadv3580m000) session, insert a record named item, and choose Enterprise Server. In the text editor, enter the following query:

```
Select tiitm001.citg,     | Item group
 count(tiitm001.item)     | Item
 from  tiitm001           | Item data
 group by tiitm001.citg
 order by tiitm001.citg
```

LN Environment Variables and Resources 13

This chapter contains the following sections:

- “LN Environment Variables and Resources,” describes environment variables and resources you can use in the LN environment.
- “To set Environment Variables and Resources,” describes how to set environment variables and resources.
- “Search Structure,” describes the search structure used to retrieve the values of the variables and resources at runtime.
- “Miscellaneous,” describes miscellaneous topics, such as how the ipc_boot and bshell executables can expand environment variables, and how jar files in directories, which occur in the BSE_CLASSPATH, are added to the java.class.path property.

Note: For more information on environment variables, refer to *Infor Baan IV, Baan 5.0, LN - Installation Guide Infor BW (U7434 US)*.

LN Environment Variables and Resources

Introduction

LN uses two kinds of variables:

- Environment variables:

  These are operating system variables that have a meaning for LN. For example, BSE, BSE_TMP, and PATH.

- Resource variables:

  These variables are not available in the operating system, but are only recognized within the context of LN. For example, high_ascii_tolerance, bshell_max_strbuf_size, and use shm_info.

Various LN variables have a corresponding resource.

Environment Variables and Resources overview

The following table lists the environment variables you can use in the LN environment and, if available, the corresponding resources.

Variable	Resource	Description
AUDITLOG	n/a	Used to enable printing of various types of audit information. The variable must have an octal value, such as 0001, 002, 0004, and 0010. • Used by: Audit server • Configuration: Command field in Infor BW configuration, or environment of server
ATTACHMENTS_IMPL	n/a	Through the Object Data Management (ODM) and Enterprise Content Management (ECM) applications, attachments can be linked to records. If both applications are present, this variable is used to indicate which application should be used to handle attachments. Possible values: “ECM” or “ODM”.
BAANLOGIN_PORT	n/a	The port number for remote login. Only use this variable if the port number differs from the default. This variable is usually specified in remote user files. • Used by: fs binary for contacting the LN server • Configuration: Enterprise Server Service Manager or environment
BAANLOGIN_PROTOCOL	n/a	The login protocol for remote login. This variable is usually specified in remote user files. • Used by: fs binary for contacting the LN server • Configuration: Enterprise Server Service Manager or environment
BAAN_WIN_TITLE	n/a	Creates a string that contains the title for the window, depending on the formatting string set by environment variable BAAN_WIN_TITLE. These formatting variables are available with BAAN_WIN_TITLE: • %h: Hostname • %c: Company number • %C: Company description • %f: Name of current filter • %s: Session code • %S: Session description • %I: Name of current index • %u: Name of the user • %p: Package combination • %o: Window type • %d: No debug

Variable	Resource	Description

		Command line example:
		`-set BAAN_WIN_TITLE="%h: %s : %S< [%c]><(filter: %f)> "`
		If you run the ttadv2500m000 session in company 000 on host Saturn with filter Tools, this will result in this window title:
		Saturn: ttadv2500m000: Sessions [000] (filter: Tools)
		You can also remove the format variable and surrounding string if the value is considered empty. To suppress showing empty variables and the surrounding substring place the text between '**<**' and '**>**'. For example:
		`<filter: %f>`
		If no filter is active for a session, the title will not contain any filter information. If a filter is active for a session, the title will contain the string: filter: "name of the filter".
		• Used by: Web UI, Infor BW
		• Configuration: Web UI user profile or Command field in Infor BW configuration

`BAAN_SCM_GRP`	n/a	The scm group, which is used in an .fd file on development systems. • Used by: Bshell • Configuration: Command field in Infor BW configuration For details on SCM, refer to *Infor Enterprise Server - Administration Guide (U8854)*.
`BAANHOME`	n/a	Location of the shared binaries for different LN environments. For example, ASM and older versions of SLM will be installed in this directory. • Used by: Bshell • Configuration: bse_vars (UNIX) or Enterprise Server Service Manager (Windows)
`BDBRECONFIG_PARALLEL`	`db_resource`	Determines the number of parallel processes started by bdbreconfig to reconfigure tables. • Used by: bdbreconfig • Configuration: bse_vars (UNIX) or Enterprise Server Service Manager (Windows)
`BIRT_HOME`	n/a	Location of BIRT runtime binaries used to convert 4GL reports to Report Viewer enabled reports. • Used by: BIRT reporting • Configuration: bse_vars (UNIX) or Enterprise Server Service • Default value: ${BSE}/birt-runtime

Variable	Resource	Description
		For details on Report Viewer enabled reporting, refer to *Infor LN - Development Tools Development Guide (U8883)*.
BIRT_TEMP	n/a	Directory to store temporary files for the first version of the LN homepages, delivered with Enterprise Server 8.4. • Used by: LN homepages • Configuration: bse_vars (UNIX) or Enterprise Server Service Manager (Windows) For details on LN homepages, refer to *Infor LN - Development Tools Development Guide (U8883)*.
BSE	n/a	The directory where the LN software environment is stored. • Used by: All porting set binaries • Configuration: Command field in Infor BW configuration
BSE_CLASSPATH	n/a	The search path used by the bshell (jvmi) to search for java classes. • Used by: Bshell • Configuration: Command field in Infor BW configuration, bse_vars (UNIX), or Enterprise Server Service Manager (Windows)
BSE_COMPNR	n/a	The company number, such as 050 or 055. When you log on, the LN sessions operate on the tables of the specified company number. This variable overrules the company number specified in the LN user data. • Used by: Bshell • Configuration: Command field in Infor BW configuration • Possible values: Any company number in your LN environment. To view the companies, use the Companies (ttaad1100m000) session. **Note:** Only select a company number linked to the user's package combination.
BSE_DATA_LANG	datalang	The data language, such as nl (Dutch/Flemish), en (English), de (German), or fr (French). When you log on, the LN application data is displayed in the specified language. This variable overrules the Data Language setting in the LN user data. • Used by: Bshell • Configuration: Environment or as resource in $BSE/lib/all • Possible values: Any data language code used in your LN environment. To view data language codes, use the Data Languages (ttaad1111m000) session. • Default value: undefined

Variable	Resource	Description
		Note: You can only use this variable if Multilanguage Fields Support is enabled on your LN server. For details, refer to *Infor Enterprise Server - Administration Guide (U8854)*.
BSE_FIN_COMPNR	n/a	The financial company. This variable overrules the company number specified in the LN user data. • Used by: Bshell • Configuration: Command field in Infor BW configuration
BSE_LANG	language	The software language, such as 1 (Nederlands), 2 (English), 3 (Deutsch), or 4 (French). When you log on, the LN software is displayed in the specified software language. This variable overrules the Language setting in the LN user data. • Used by: Bshell • Configuration: Command field in Infor BW configuration • Possible values: Any language code installed on your LN server. To view the language codes, use the Software Languages (ttaad1510m000) session. • Default value: 2
BSE_LOG_COMPNR	n/a	The logistic company. This variable overrules the company number specified in the LN user data. • Used by: Bshell • Configuration: Command field in Infor BW configuration
BSEREM	n/a	Identifies the MAS host and BSE path in environments with distributed application servers (AS/MAS concept). This variable is set during the installation of LN. • Used by: Bshell • Configuration: Command field in Infor BW configuration, bse_vars (UNIX), or Enterprise Server Service Manager (Windows)
BSE_SORT	n/a	Path where temporary files are stored during the sort process. It is used by the sort binary, for example, when reports are printed. • Used by: Bshell • Configuration: bse_vars (UNIX), or Enterprise Server Service Manager (Windows) • Default value: ${BSE}/tmp
BSE_STD_PROGRAM	std_program	A program name that overrides the standard program (4GL engine), which Tools developers and support personnel use. • Used by: Bshell

Variable	Resource	Description
		• Configuration: Command field in Infor BW configuration, or LN user data
BSE_TMP	n/a	The directory where LN stores its temporary files. • Used by: Bshell • Configuration: Environment or Command field in Infor BW configuration • Default value: ${BSE}/tmp
BSE_USER_AS_FOUND_BY_SSO	n/a	Variable needed for passing the validated SSO user on to ipc_boot and to the Bshell. • Used by: Bshell • Configuration: Environment
CUSTOMIZE	n/a	This variable indicates whether end user customizations on forms will be displayed. Possible values: yes/no. Value “NO” will hide the end user customizations on forms at runtime. **Note:** This is not a porting set variable.
DISPLAY	n/a	UNIX/Linux variable that must be set to perform operations on images uploaded in Web UI.
FGL_STUDIO_ENABLED	n/a	Indicates whether the LN environment is enabled for software development through Enterprise Server Application Studio. Possible values: 1 (enabled) / 0 (disabled). For details, refer to the Application Studio documentation.
JAVA_HOME	n/a	Indicates the installation Java directory. • Used by: Bshell Java integration • Configuration: Environment or bse_vars (UNIX), or Enterprise Server Service Manager (Windows)
JRE_HOME	n/a	Installation directory of the Java runtime engine. • Used by: Bshell Java integration • Configuration: Environment or bse_vars (UNIX), or Enterprise Server Service Manager (Windows)
LD_LIBRARY_PATH	n/a	HP, Sun Solaris, and Linux operating system variable that indicates the Library path. • Used by: Bshell and database drivers on HP, Sun Solaris, and Linux • Configuration: Environment

Variable	Resource	Description
LIBPATH	n/a	AIX operating system variable that indicates the search path for shared (dynamic) libraries. For example: *LIBPATH=${BSE}/shlib:/opt/db2/lib* • Used by: Bshell and database drivers on AIX • Configuration: Environment
LTS_LANGUAGE	n/a	Sets the Language Translation Support (LTS) language (overrides the language specified in the user data). **Note:** This is not a porting set variable. For details on LTS, refer to the Infor Web Help and the *Infor LN - Development Tools Development Guide (U8883)*.
LTS_RESOLVE_MODE	n/a	The mode to retrieve labels during the LTS process. Possible values/modes: “dump”, “database”, and “fallback”. **Note:** This is not a porting set variable. For details, refer to *Infor Baan IV, Baan 5.0, LN - Installation Guide Infor LN BW (U7434)*.
MAX_RETRY	n/a	This variable is used to test transaction management programming structures. It indicates the number of times the system may return to a retry point as a result of an abort in an update action. The default value is 10. For details, refer to the LN Programmer’s Guide.
PACKAGE_COMB	pacc	Sets the package combination and overrides the user data. • Used by: Bshell • Configuration: Environment or Command field in Infor BW configuration
PATH	n/a	The search path for Windows/UNIX operating system executables. **Note:** The ${BSE}/bin directory must be included in the PATH. • Used by: All porting set binaries • Configuration: Environment
SHLIB_PATH	n/a	Operating system variable that indicates the Library path as set on HP RISC (HP-UX) systems. • Used by: Bshell and database drivers on HP RISC (HP-UX) • Configuration: Environment
SH_TABLE	n/a	The name of the file that contains your own shift table. This variable is used in Native Language Support (NLS).

Variable	Resource	Description
SLMHOME	n/a	Installation directory for newer SLM versions. Replaces the BAANHOME variable.
SORT_TABLE	n/a	The name of the file that contains your own sort table. This variable is used in Native Language Support (NLS). • Used by: Bshell • Configuration: Environment or Command field in Infor BW configuration
SSI_MAX_ROWS	n/a	The number of records that can be exported simultaneously from an LN session to MS Excel. If a session contains a lot of data, a full export from that session can slow down or block the entire database. To prevent this, you can limit the number of records that can be exported by setting this parameter. To limit the number of records that can be exported, set SSI_MAX_ROWS to a value greater than zero. For example, if SSI_MAX_ROWS = 500, 500 records can be exported at most. If SSI_MAX_ROWS has a value that is less than or equal to zero, there is no limit (except the MS Excel limit) for the number of records that can be exported.
SUPPRESS_INHELP	n/a	Variable to suppress windows help (Infor Baan IV/ Infor Baan 5.0c). To use ASCII Help instead of Windows Help, set the variable to 1. **Note:** This is not a porting set variable.
TC_STARTUP_TIME	n/a	This variable is used when you access LN through Web UI. The variable indicates the number of seconds the back-end waits, after a new session is started, before a response is sent to Web UI. If you do not set the variable, the back-end waits a maximum of 15 seconds. • Used by: Web UI • Configuration: Web UI user profile **Note:** This variable is only relevant for application developers who debug LN applications. Within the specified time duration, a newly started session must notify the tcserver—a back-end component that handles communication with Web UI—whether it requires a UI. If the tcserver is not notified timely, Web UI and tcserver ignore all UI actions of the session, so the session runs without a UI. When you debug the back-end, you must sometimes increase TC_STARTUP_TIME. For example, because a session stops in a breakpoint before it can notify tcserver.

Variable	Resource	Description
TEST_RETRY	n/a	This variable is used to test transaction management programming structures. It indicates the number of times the system must go back to a retry point at the moment of committing. The variable cannot be used when testing different sessions with retry points parallel. For details, refer to the LN Programmer's Guide.
USER	n/a	Variable to operate using different LN user names. • Used by: Porting set binaries • Configuration: Environment, user data, or Command field in Infor BW configuration • Default value: Your operating system login name For details, refer to the online help of the User Data (ttams1100s000) details session.

For details on resources not listed in the previous table, refer to the following documents:

- The technical reference manual of your database driver.
- The *Infor LN - Performance, Tracing and Tuning Guide (U9357)*.

To set Environment Variables and Resources

Environment variables can be set in various ways:

- Users can set their own variables.
- Administrators can define global variables for all users.

Important: If the same variable is set twice, as a global variable and as a user variable, the setting of the user variable overrules the global variable.

To set user variables

Web UI users can set variables in their Web UI user profiles. WorkTop users can set variables in their Infor BW configuration.

To set a variable, a user must enter the following syntax in the **Command** field in the Web UI user profile or in the BW configuration:

```
-set <variable name>=<value>
```

For example:

```
-set BSE_COMPNR=150
```

To set multiple variables, a user must enter the following syntax:

```
-set <variable name1>=<value1> -set <variable name2>=<value2>
```

For example:

```
-set BSE_COMPNR=150 -set BSE_LANG=3
```

To set global variables

UNIX/Linux

Global variables for all LN environments

On a UNIX/Linux platform, administrators can set system wide variables in the /etc/profile file. These variables apply to all LN environments on the system. The following syntax must be used:

```
<variable name>=<value>
export <variable name>
```

For example:

```
BSE_TMP=${BSE}/tmp
export BSE_TMP
```

Global variables for a single LN environment

Administrators can set the variables for a single LN environment in the ${BSE}/lib/bse_vars file of that environment. The following syntax must be used:

```
<variable name>=<value>
```

For example:

```
SLMHOME=/usr/slm
```

Note: In bse_vars, administrators can use only global variables listed in "Environment Variables and Resources overview" on page 176.

Windows

On a Windows platform, administrators cannot set variables in bse_vars. In Windows, administrators can set any environment variable in the Enterprise Server Service Manager.

Alternatively, administrators can set system wide variables in the **Environment Variables** dialog. To start this dialog, complete the following steps:

1. Right-click **My Computer** and, on the shortcut menu, click **Properties**. The **System Properties** dialog appears.
2. Go to the **Advanced** tab and click **Environment Variables**.

Web UI

In the Web UI Administration Console, administrators can define global variables for all Web UI users. The variables must be set in the **Command** field in the LN Environment properties. For details, refer to *Infor Web UI - Installation and Configuration Guide (U8715 US)*.

The syntax to set a variable is identical to the syntax used in BW configurations and Web UI user profiles. For details, refer to "To set user variables" on page 183.

To set resources

Resources are more static than environment variables and cannot be defined by end users.

Administrators can set resources in the following files:

File	Description
${BSE}/lib/user/u<user>	The user file of a specific user. Administrators can edit this file to set resources for a user. **Note:** Resource settings that were added manually are lost when the user data is converted to runtime.
${BSE}/lib/defaults/<binary>	This is for internal use by Infor only. For example, resources for the bshell executable can be set in the ${BSE}/lib/defaults/bshell6.2 file.
${BSE}/lib/defaults/db_resource	This file is used to set database driver related resources, such as rds_full, lock_retry, and use_shm_info.
${BSE}/lib/defaults/all	This file is used to set remaining global resources.

To set a resource in the files mentioned, the following syntax must be used:

```
<resource name>:<value>
```

For example:

```
use_shm_info:0
```

Note: To overrule a resource, users can use the corresponding environment variable in their Web UI user profile or BW configuration.

Search Structure

When a user logs on, the bshell initializes the LN variables and resources. The search structure used to retrieve the values of the variables and resources depends on the operating system of the LN server.

UNIX/Linux

On UNIX/Linux systems, the bshell uses the following search order:

1. The user's Infor BW configuration or Web UI user profile
2. Any system-wide global resource setting defined in, for example, /etc/profile or /etc/environment. Global settings must be defined before the bshell executable is started.
3. ${HOME}/bse_vars
4. ${BSE}/lib/bse_vars
5. /usr/bse/bse_vars
6. ${BSE}/lib/user/u<user> (user file)
7. ${BSE}/lib/defaults/<binary>
8. ${BSE}/lib/defaults/db_resource
9. ${BSE}/lib/defaults/all

Examples

- A variable is defined in a user's Web UI user profile, in /etc/profile, and in ${BSE}/lib/bse_vars .The Web UI user profile setting overrules the other settings.
- A resource is set in ${BSE}/lib/defaults/db_resource and in ${BSE}/ lib/defaults/all/. The setting in ${BSE}/lib/defaults/db_resource takes precedence.
- A resource is set in ${BSE}/lib/defaults/db_resource. Users define the corresponding environment variable in their Web UI user profile. The Web UI user profile setting overrules the resource setting.

Windows

On Windows systems, the bshell uses the following search order:

1. The user's Infor BW configuration or Web UI user profile
2. Global variable settings defined through the Enterprise Server Service Manager.
3. Global Windows environment variable settings, defined in the **Environment Variables** dialog.
4. ${BSE}/lib/user/u<user> (user file)
5. ${BSE}/lib/defaults/<binary>
6. ${BSE}/lib/defaults/db_resource
7. ${BSE}/lib/defaults/all

Examples

- A variable is set in the user's Web UI user profile and in the Enterprise Server Service Manager. The Web UI user profile setting takes precedence.
- A variable is set in the **Environment Variables** dialog and in the Enterprise Server Service Manager. The setting defined in the Enterprise Server Service Manager takes precedence.

Miscellaneous

Variable expansion

ipc_boot and bshell can expand environment variables listed in the $BSE/lib/bse_vars file.

A variable is only expanded if its name is surrounded by braces ({ and }).

Example

```
BSE=/usr/bse
```

$BSE/lib/bse_vars contains the following setting:

```
LIBPATH=${BSE}/shlib:/opt/db2/lib
```

This is expanded to the following:

```
LIBPATH=/usr/bse/shlib:/opt/db2/lib
```

Note: If the braces are missing, the variable is not expanded; for example,

```
LIBPATH=$BSE/shlib:/opt/db2/lib
```

is not expanded.

Directories in BSE_CLASSPATH

Bshell checks if parts of the BSE_CLASSPATH variable contain directory names. In case of a directory name, the bshell adds all jar files in that directory to the java.class.path property.

Example

The /opt/birt/lib directory contains 3 jar files: a.jar, b.jar, and c.jar.

The BSE_CLASSPATH=/opt/birt/lib setting results in the following:

```
-Djava.class.path=/opt/birt/lib:/opt/birt/lib/a.jar:
/opt/birt/lib/b.jar:/opt/birt/lib/c.jar
```

In previous versions, all jar files from $BSE/java were added in a different order.

```
-Djava.class.path=/usr/bse/java/bjvmi.
```

Document Authorization

14

General

This chapter introduces Document Authorization, also known as Database Change Management (DBCM). DBCM provides a way to change business objects in a controlled way. Only after (external) approval, changes become available for general use.

Objects

Document Authorization is not per se on the record level. Therefore objects that comprise multiple tables and records can be modeled. When an object is modified, for example by updating an attribute, a draft, or checked-out, version of that object is stored. If the object change requires approval and as long as no approval is received, the checked-out version cannot be used. After approval the checked-out version is checked-in, overwriting the current (approved) version and making it available to the system.

The updates on an object are maintained by storing only the modified records. This saves both storage during the lifetime of the checked-out object and time during the checking in of the object.

Runtime files

If Document Authorization is deployed, these runtime files exist:

- ${BSE}/lib/models/deployment.<package combination>.xml
- ${BSE}/lib/models/model.<id>.xml

The model file contains this information:

- Which object types are available
- The structure of the object types

The deployment file contains this information:

- Which object types are under Document Authorization

- The companies and actions for which this applies
- A reference (“Model” attribute of the “Deployment” node) to the correct model file.

The runtime DD files of the tables that are under DBCM in any company have the following attribute, which triggers the data model changes described below:

- :DBCM:1

Data model changes

All tables that are under DBCM get a “checked-outs” table. The layout of this table is identical to the normal, or “checked-ins”, table. The name of the “checked-outs” table is identical to the name of the normal table, except that the first “t” is replaced with a “v”. For example, the ttdsls400812 table gets a “checked-outs” table with the name vtdsls400812. The checked-outs table is used to store non-approved, or draft, changes to rows.

All tables that are under DBCM are extended with these columns:

- rcd_toid
- rcd_cmac
- rcd_seqn (Db2 and Microsoft SQL Server)

The values of these columns are maintained by the system. We strongly discourage that you manually modify them.

The rcd_toid column

This column groups all records in the database that belong to one object. For example, if the model would contain the “sls400” SalesOrder type, then all records that belong to the same SalesOrder object would have the same rcd_toid value. The first 6 positions of the rcd_toid column contain the object type, such as “sls400”; together with the remaining positions they form a unique identifier.

The value of the rcd_toid column may be **empty** (a string of spaces only), in which case it is at this time unknown to which object this record belongs. As soon as the record is modified, the rcd_toid value is determined. The value of the rcd_toid column may also be the “-NONE-“ value, in which case it is known that the record does not belong to any object.

The rcd_cmac column

This column keeps track of the change management status of individual records within an object. It can have these values:

- 0: The record has not been modified; occurs in checked-ins table only.

- 1: The record is inserted; occurs in checked-outs table only.
- 2: The record is updated.
- 3: The record is deleted.

If the rcd_cmac value of a record in the checked-ins table is non-zero, then a record with identical primary key, rcd_toid, and rcd_cmac values must exist in the checked-outs table.

The rcd_seqn column

The rcd_seqn column is used only on Db2 and Microsoft SQL server. This column is used to guarantee read stability on the row set of SQL select statements that operate on DBCM tables. Without this column, the same row could appear twice in the row set, or not appear at all.

Database components

For Document Authorization these new database components are introduced:

- The “Checked Out Business Objects” (tttocm999000) table
- A database sequence named SEQ_rcd_seqn

The Checked Out Business Objects table

The tttocm999000 table contains metadata about the checked-out objects, such as the user who initiated the check-out and the timestamp when this event occurred. The primary key of the table is the *toid* column, which uniquely identifies the checked-out object.

The SEQ_rcd_seqn sequence

The SEQ_rcd_seqn sequence is used to populate the rcd_seqn column and to guarantee read stability as described before. The sequence is created automatically by the database driver, if the driver needs it.

It is important that the sequence generates an ever increasing sequence of numbers. Do not remove or alter the sequence.