EAServer 6.0.2 New Features Guide

Document ID: DC38032-01-0602-01

Last revised: June 2007

Migration tool enhancements

The EAServer 6.0.2 migration tool enables users to migrate a subset of entity types from the 5.*x* repository to the 6.0.2 repository. For example, you can migrate only the connection caches from the 5.*x* repository. This saves time by scanning only those entity types in which users are interested.

❖ **Selecting entities to migrate from EAServer 5.*x* to 6.0.2**

1 Start the migration tool:

   a On the command line, change to the EAServer *bin* subdirectory, and run:

```
migrate -gui
```

The graphical interface to the migration tool displays.

b To view the *EAServer Migration Guide*, select Help | Migration Guide.

2 Select File | Scan EAServer 5.*x* Repository.

3 In the Select Entities dialog, select or unselect the entity types to migrate. You can also click Select All or Select None. By default, all entity types are selected.

Note When you select an entity type, the entity types on which it depends are selected automatically. For example, if you select application, the connection cache and connector entity types are selected.

4 Click Start Scan. EAServer scans the 5.*x* database for the selected entity types. The migration tool displays the entity types in the 5.*x* repository, and depicts each entity type's state using these icons:

- A check mark indicates that the entity type was scanned successfully.
- An "X" indicates that the attempt to scan the entity type failed.
- A question mark indicates that the entity type was not scanned.

5 In the left pane, select any of the entity types that have a check mark; the entities of that type display in the right pane:

- If an entity has not already been migrated, a check mark displays to the left of it to indicate it is selected to migrate.
- If an entity has already been migrated, "Previously Migrated" displays to the right of the entity name, and it is not selected. You can select to migrate the entity again.
- If an entity cannot be migrated (for example, an OCI 8 connection cache, which is not supported in EAServer 6.*x*), the entity name is dimmed.

6 Select File | Migrate to migrate the selected EAServer 5.*x* entities to the EAServer 6.0.2 installation. The migration tool displays the list of entities selected for migration.

7 Select whether to overwrite existing entities in the migration directory. If you migrate from the same EAServer host and port more than once, select this option. Some files are written to the migration directory during the initial migration, and if you do not select to overwrite, subsequent migration attempts fail.

8 Verify the list of entities to migrate, and click Confirm to begin migrating the 5.*x* entities.

SQL Anywhere 10.0

When you upgrade to EAServer 6.0.2, Adaptive Server™ Anywhere (ASA) version 9.0.2 is upgraded to version 10.0, and the name of the database server changes from ASA to SQL Anywhere®. The EAServer 6.0.2 upgrade script installs a new database (default.db) that is used by the server.

SQL Anywhere 10.0 does not work with an ASA 9.*x* database. To continue using the data in an ASA 9.*x* database, run the asa-unload script to move the data into the SQL Anywhere 10.0 database.

❖ **Moving data from an ASA 9.0.2 database to a SQL Anywhere 10.0 database**

First, upgrade EAServer to version 6.0.2, then move the data to the new database.

- In a command window, change to the EAServer *bin* subdirectory and run:

  ```
  asa-unload.bat [options] <directory>@<data>
  ```

  The asa-unload script calls dbunload [*options*] <*directory*>@<*data*>, where:

  - *options* are any of the options listed in Table 1.
  - *data* is either an environment variable or a file from which the data is expanded.
  - *directory* is the path to *data*. The path must be available to the database server, unless you use an external unload.

  See “Sample upgrade” on page 5.

Table 1: asa-unload options

Option	Description
-ac "*keyword=value*;..."	Connection parameters for the ASA 9.*x* database
-an <*db_file*>	Create a database file with the same settings as the old database file, and move the data into the new database.
-ap <*size*>	Set the page size of the new database
-ar	Rebuild and replace the database
-c "*keyword=value*;..."	Database connection parameters
-d	Unload data only
-dc	Recompute columns
-e <*list*>	Do not display output for the listed tables
-ea <*algorithm*>	Encryption algorithm for the new database; used with -ek or -ep
-ek <*key*>	Encryption key for the new database; used with -an or -ar
-ep	Prompt for the new database encryption key; used with -an or -ar
-er	Remove encryption from encrypted tables
-et	Enable encrypted tables in new database; used with -an or -ar
-g	Generate statements to refresh valid materialized views
-ii	Internal unload, internal reload (the default)
-ix	Internal unload, external reload
-k	Create an auxiliary catalog; used with diagnostic tracing
-m	Do not preserve user IDs for a replicating database
-n	No data; schema definition only
-nl	No data; include LOAD/INPUT statements in reload script
-o <*file*>	Logs output messages to *file*
-p <*char*>	Escape character; the default is "\"
-q	Suppress display of messages and windows
-r <*file*>	Name of the generated *reload* ISQL command file; the default is *reload.sql*
-t <*list*>	Output the listed tables only
-u	Data can be unordered
-v	Verbose messages

Option	Description
-xi	External unload, internal reload
-xx	External unload, external reload
-y	Replace existing command file without asking for confirmation

You can find complete upgrade instructions in Upgrading to SQL Anywhere 10 at http://www.ianywhere.com/downloads/whitepapers/upgrading_sas_to10.pdf.

For an overview of SQL Anywhere 10 features, see What's New in Version 10.0.0 at http://www.ianywhere.com/developer/product_manuals/sqlanywhere/1000/en/html/dbwnen10/wn-newjasper.html.

Sample upgrade

This example calls `asa-unload` to move data from an ASA 9.*x* database to a SQL Anywhere 10.0 database:

```
asa-unload -c "dbf=..\data\default.db;uid=dba;pwd=sql" -ar
-o ..\data\dbunload.txt
```

where:

- `-c "dbf=..\data\default.db;uid=dba;pwd=sql"` specifies the database connection parameters: path and name of the 9.*x* database file, user ID, and password.
- `-ar` replaces and rebuilds the database.
- `-o ..\data\dbunload.txt` specifies the name and location of the output log file.

As the data is moved, the console displays:

```
SQL Anywhere Unload Utility Version 10.0.0.2745
Connecting and initializing
Unloading user and group definitions
Unloading table definitions
Unloading index definitions
Unloading functions
Unloading view definitions
Unloading procedures
Unloading triggers
Unloading SQL Remote definitions
Unloading MobiLink definitions
Creating new database
Creating indexes
```

Jetty 6.1 support

Jetty support is upgraded from version 5.*x* to 6.1, which improves performance.

Socket listener properties

New listener properties support socket channels with Jetty 6.1.

❖ **Setting socket listener properties**

Other listener configuration properties are described in Chapter 3, "Creating and Configuring Servers," in the *EAServer System Administration Guide*.

1 Start EAServer, and use the Management Console to connect to the server.

2 Expand the Listeners folder, then select the listener to configure.

3 On the General tab, enter:

Display name	Configuration property	Description	Default value
Use SocketChannel Type Listener	useSocketChannel	Use a socket channel, instead of a simple socket; true or false. **Note** HTTPS and IIOPS listeners always use simple sockets.	false
Thread per Connection	useBlockingNIO	Use blocking NIO; true or false. **Note** IIOP listeners always use blocking NIO.	false
Maximum Number of Threads	maxThreads	Maximum number of threads that EAServer can use to process requests, including acceptor threads for NIO	15
Number of Acceptors	numberOfAcceptor	Number of acceptor threads For NIO, the number of threads available for processing requests equals the maximum number of threads minus the number of acceptor threads.	1
Header Buffer Size (bytes)	headerBufferSize	Size of the buffer header	8192 bytes
Request Buffer Size (bytes)	requestBufferSize	Size of the request buffer	32768 bytes
Response Buffer Size (bytes)	responseBufferSize	Size of the response buffer	65536 bytes

Display name	Configuration property	Description	Default value
Maximum Idle Time (ms)	maxIdleTime	Number of milliseconds a socket remains idle before it disconnects	3600000ms
Socket Linger Time (ms)	soLingerTime	Number of milliseconds a socket waits to transmit data. After the specified period of time, unsent data is lost.	1000ms

Connection types

The values of useSocketChannel and useBlockingNIO define the connection type:

For this connection type	Set useSocketChannel to	Set useBlockingNIO to
Sockets	false	true or false
Blocking NIO	true	true
Select channel NIO	true	false

HTTP server properties

The following HTTP properties are obsolete in EAServer 6.0.2:

- httpUseCustomGetServerInfo
- httpProxyProtocol
- httpProxyPort
- httpGetServerInfoFrom

The server uses the HTTP header “host” to determine which host and port the client used to connect. The protocol used to connect to a Web redirector is defined by the header “sybaseredirectorheader,” which is set when a connection is made through the redirector.

If you call the HttpRequest methods getServerName, getServerPort, or getProtocol, the values are retrieved from the browser. This is the same behavior that was achieved by setting httpGetServerInfoFrom to “source” in versions of EAServer earlier than 6.0.2.

Faster deployment

EAServer 6.0.2 provides better performance for deploying:

- J2EE JARs
- EJBs from EJB-JARs
- EJBs and Web applications from EARs
- PowerBuilder® and other CORBA components

New database type and data source properties

EAServer 6.0.2 includes new properties for database types and data sources.

Database type properties:

- **jdbcDriverDownloadURL** Specifies the URL from which you can download the client-side database driver. The default value is an empty string.
- **isDownloadZipped** Specifies whether the driver file downloaded from jdbcDriverDownloadURL is in ZIP format. If set to true, EAServer copies the file to *%DJC_HOME%\lib\jdbc* (Windows) or *$DJC_HOME/lib/jdbc* (UNIX), then unzips it; if set to false, the driver file is copied but not unzipped. The default value is false. This property is ignored if the value of jdbcDriverDownloadURL is an empty string.

Data source properties:

- **useTransactionalPing** Specifies whether to start a transaction before pinging a connection. To ping a resource that supports XA connections, you must first start a transaction. The default value is true.
- **jdbcDriverDownloadURL** Same as database type property jdbcDriverDownloadURL.
- **isDownloadZipped** Same as database type property isDownloadZipped.

unixODBC support

In EAServer 6.0.2, unixODBC replaces Intersolv ODBC on UNIX and Linux platforms. unixODBC supports most APIs defined in the ODBC specification, and fully supports Unicode client requests.

An EAServer ODBC client can use either ODBC or ODBC Unicode data sources. To initialize the *odbc.ini* or *odbcinst.ini* file, you can use an Ant configuration script.

unixODBC and SQL Anywhere

To use unixODBC with a SQL Anywhere database, configure an ODBC data source.

❖ **Configuring ODBC for SQL Anywhere**

1 If EAServer is running, shut it down.

2 Change to the EAServer *bin* directory, and start SQL Anywhere with the default.db database:

```
./asa-init.sh ../data/default.db
./asa-start.sh -x "tcpip{MyIP=127.0.0.1;
Host=localhost;Port=2638;DoBroadcast=no;BroadcastListener=no}"
-n EAS_TEST_DB ../data/default.db
```

3 Change to the EAServer *odbc* directory.

4 On Solaris, source either *odbc.sh* (Bash shell) or *odbc.csh* (C shell).

On Linux, source *odbc.sh*.

5 Edit *odbc.ini*, and add these lines:

```
[EAS_TEST_DB]
Driver =SYBASA
Trace =no
TraceFile =sql.log
Port =2638
ServerName =EAS_TEST_DB
TDS_Version =5.0
```

Note The value of ServerName must match the value of the -n option in step 2.

On Solaris, there must not be a space between “ServerName =” and the value.

6 Change to *odbc/bin*, then use isql to verify that you can connect to the database:

```
./isql -v "EAS_TEST_DB" dba sql
```

A successful connection displays the `SQL>` prompt.

7 If this error message displays:

```
./isql: error while loading shared libraries:
libreadline.so.4: cannot open shared object file: No
such file or directory
```

isql cannot find *libreadline.so.4*.

To rectify the problem:

a Find *libreadline.so.4* by running:

```
whereis libreadline.so.4
```

The output should look similar to:

```
libreadline.so: /lib/libreadline.so.5
```

b Change to *odbc/lib*, and create a soft link to *libreadline.so.4*:

```
ln -s /lib/libreadline.so.5 libreadline.so.4
```

c Change to *odbc/bin*, then use isql to verify that you can connect to the database:

```
./isql -v "EAS_TEST_DB" dba sql
```

8 Start EAServer, then start the Management Console.

9 In the Management Console, create a data source named "EAS_TEST_DB":

a Expand the Resources folder, right-click Data Sources, and select Add.

b To test the connection, click Ping.

unixODBC and Adaptive Server Enterprise

To communicate with an Adaptive Server Enterprise database, ODBC clients need an ODBC driver. On Linux platforms, the Open Client™ package includes an ODBC driver; on other UNIX platforms, it does not.

On UNIX, FreeTDS allows ODBC clients to talk to Adaptive Server Enterprise databases. FreeTDS is an open source implementation of the TDS (Tabular Data Stream) protocol. You can install FreeTDS to use with unixODBC.

❖ **Installing FreeTDS**

1. Download FreeTDS from the FreeTDS Web site at http://www.freetds.org. Under Quick Links | Latest Versions, select Stable Release.
2. Save *freetds-stable.tgz* to your machine, then uncompress the file with gunzip.
3. Change to the FreeTDS installation directory, and run:

   On Solaris and HP UNIX platforms:

   ```
   ./configure --prefix=$DJC_HOME/freetds \
   --with-tdsver=5.0 \
   --with-unix-nodm=$DJC_HOME/odbc
   ```

   On AIX:

   ```
   ./configure --prefix=$DJC_HOME/freetds \
   --with-tdsver=5.0 \
   --disable-shared \
   --with-unix-nodm=$DJC_HOME/odbc
   ```

   The console displays the configuration status.
4. On Solaris, verify that cc is before gcc in the path to use the SUNWspro6.1 compiler.
5. Change to the *src* subdirectory, and run:

   ```
   make; make install
   ```

   If you see the following error messages, the problem may be errors in the *libtool* script.

```
ld: fatal: option -z has illegal argument `-Wl,allextract
ld: fatal: option -z has illegal argument `-Wl,defaultextract
ld: fatal: Flags processing errors
```

   To fix the problem:

   a. In the FreeTDS installation directory, open *libtool* with a text editor.
   b. Search for this line:

```
whole_archive_flag_spec="\${wl}-z \${wl}allextract\$convenience \
${wl}-z \${wl}defaultextract"
```

      and replace it with:

```
whole_archive_flag_spec="\${wl}-z allextract\$convenience \
${wl}-z defaultextract"
```

   c. Change to *src*, and run:

```
make; make install
```

❖ **Configuring FreeTDS and unixODBC support**

1 Change to the FreeTDS *etc* subdirectory, open *freetds.conf* with a text editor, and add these lines:

```
[MySybase]
host=<ASE host>
port=5000
tds version=5.0
```

where *ASE host* is the name of the machine where Adaptive Server Enterprise is installed.

To test the connection to MySybase, change to the FreeTDS *bin* subdirectory, and run

```
tsql
```

2 Change to *$DJC_HOME/unixodbc*, and edit *odbcinst.ini* to define the ODBC driver:

```
[SybaseASE]
Description=Sybase ODBC
Driver =EAServer/freetds/lib/libtdsodbc.so
Setup =EAServer/freetds/lib/libtds.so
FileUsage =1
UsageCount =3
```

where *EAServer* is the full path to the EAServer installation directory.

3 Edit *odbc.ini* to define an ODBC data source:

```
[testase]
Driver =EAServer/freetds/lib/libtdsodbc.so
Trace =yes
Server =<ASE host>
Port =5000
Database =tempdb
uid =sa
```

where the value of Driver is the same as in *odbcinst.ini*, and *ASE host* is the name of the machine where Adaptive Server Enterprise is installed.

4 Edit *odbc.sh* (Bash shell) or *odbc.csh* (C shell), and add `$DJC_HOME/freetds/lib` to the LD_LIBRARY_PATH.

5 To test the connection, change to *$DJC_HOME/unixodbc/bin*, and run:

```
./isql -v "testase" <user name> <password>
```

6 Change to $DJC_HOME, and edit *local-setenv.sh* to add:

```
if [ "$LD_LIBRARY_PATH" = "" ]; then
LD_LIBRARY_PATH=$DJC_HOME/freetds/lib
else
LD_LIBRARY_PATH=$DJC_HOME/freetds/lib:$LD_LIBRARY_PATH
fi
export LD_LIBRARY_PATH
```

7 Start EAServer.

CSI 2.5 support

Common Security Infrastructure (CSI) 2.5 support includes:

- Backward compatibility with the existing CSI implementation
- Exposure of the HttpServletRequest in authentication requests
- Ability to configure a security domain
- Ability to retrieve information about failed authentication requests
- Miscellaneous administrative and configuration changes

New features are based on the existing security infrastructure.

Asynchronous EJB methods

Asynchronous EJB methods enable EAServer clients to make asynchronous calls to EJB methods running in EAServer. Asynchronous methods must have a void return type, and must be subclassed from the component (not final methods).

Pinging the server

To verify whether EAServer is running, you can use either the Ant pingserver task or the jagtool pingserver command. If you specify a timeout, pingserver waits the specified number of seconds before it returns.

Ant

To ping the server from an Ant configuration script, use this syntax:

```
<pingserver host="host_name" port="iiop_port" timeout="seconds"/>
```

where:

- *host_name* is the name of the machine on which EAServer is running. If not specified, the default is the value of the HOSTNAME environment variable.
- *iiop_port* is the server IIOP port number. If not specified, the default is 2000.
- *seconds* is the number of seconds to wait before the command returns.

The *EAServer Automated Configuration Guide* describes how to create and run Ant configuration scripts.

jagtool

To ping the server using the jagtool command line tool, run:

```
jagtool [-host host_name -n iiop_port] pingserver [-timeout seconds]
```

SSL client runtime support

EAServer 6.0.2 supports SSL for C/C++ and PowerBuilder clients that connect to EAServer CORBA components.

Global binding support

EAServer 6.0.2 supports global-level name bindings. These are useful when components run outside of a context; for example, before an EJB is invoked, you can use global bindings inside static initializers to perform work. Any name binding—combination of a logical name and its referenced object—can be global. Global bindings are server-level properties.

❖ **Defining global bindings**

1. In the Management Console, expand the Configuration Files folder. If *default-application-servers-user.xml* already exists, select it, then skip to step 4.
2. Highlight the Configuration Files folder, and choose Add.
3. Run the Add wizard and specify *default-application-servers-user.xml* as the name for the new file.
4. In the right pane, inside the `configure` target, add the binding definition to the script, then click Apply. For example:

```
<target name="configure">

   <setProperties applicationServer="serverName" merge="true">
      <bind name="java:comp/env/jdbc/portaldb" dataSource="portaldb"/>
   </setProperties>

</target>
```

   where *serverName* is the name of the server. See information about using Ant configuration scripts in the *Automated Configuration Guide*.

5. For the changes to take effect:
   a. In the left pane, select *default-application-servers-user.xml*, right-click, and select Run Ant Configure.
   b. Select the server, right-click, and select Refresh.

   The name binding displays on the server's Name Bindings pane. The sample code in step 4 creates this name binding:

Bind name	Bind object
java:comp/env/jdbc/portaldb	com.sybase.djc.sql.DataSource:portaldb

Class loader tracing

To trace class loaders, you can use the system property djc.traceClassLoader. To set up class-loader tracing, run:

```
run-server.bat[sh] -Ddjc.traceClassLoader=true
```