Oracle Utilities Customer Care And Billing

Operations and Configuration Guide

Ver 2.3.1

E18371-01

August 2010

Oracle Utilities Customer Care And Billing Operations and Configuration Guide, Ver 2.3.1

E18371-01

Table of Contents

Contents

Introduction

Welcome to the Oracle Utilities Customer Care And Billing Operations and Configuration Guide for Ver 2.3.1. This guide outlines the technical concepts for operating and configuring the product on its platforms as outlined in the product installation documentation.

Note: All examples and screen captures are used for publishing purposes only and may vary from the actual values seen at your site.

Note: For publishing purposes the Oracle Utilities Customer Care And Billing product will be referred to as "product" in this document.

Contents

Updates to This Documentation

This documentation is provided with the version of the product indicated. Additional and updated information about the operations and configuration of the product is available from the Knowledge Base section of ORACLE MetaLink (http://metalink.oracle.com). Please refer to MetaLink for more information.

Other Documentation

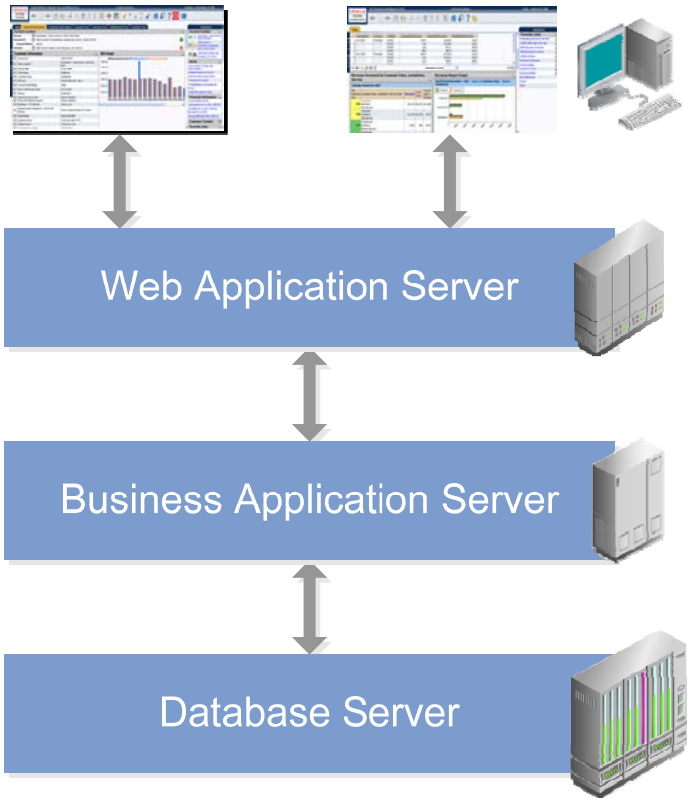
This document is part of the product technical documentation. There are groups of manuals that should also be read for additional specific advice and information:

Title	Content Summary	Location
Oracle Utilities Customer Care And Billing Installation Guide	Installation instructions for the product	http://edelivery.oracle.com
Oracle Utilities Customer Care And Billing Quick Installation Guide	Licensing and installation overview for the product	http://edelivery.oracle.com
Oracle Utilities Customer Care	DBA Administration Guide	http://edelivery.oracle.com

Title	Content Summary	Location
And Billing DBA Guide		

Architecture

The product is a multi-layered product with distinct tiers. The diagram below illustrates the architecture of the product:



The components of the architecture are as follows:

- **Client** – The client component is a browser based interface which is "light" and only requires the Internet Explorer browser to operate.

- Communication between the client and server uses the HTTP protocol across a TCP/IP network. Secure Sockets (HTTPS) is also supported[1]. The user simply uses a URL containing the product hostname and allocated port number in the address bar of Internet Explorer to access the application.

[1] Refer to the J2EE Web application server manuals for information on how to configure the HTTPS protocol.

Note: It is possible to use proxies to hide or translate the hostname and port numbers. Refer to the documentation provided with your J2EE Web application server documentation for proxy support instructions.

- **Web application server** – The product Web Application is housed in a J2EE compliant Web application server *(Refer to the Supported Platforms section of the installation guide for J2EE Web application servers and versions supported*) This server can be run on a variety of supported Windows, Linux and Unix platforms (*Refer to the Supported Platforms section of the guide for operating systems and versions supported*). Within the Web application server the pages for the product are rendered using a combination of meta data and formatting rules to ensure a consistent look and feel. These pages are written using a combination of J2EE Java script and Java. These pages are cached on the Web Server and served to the client upon request. If the page requires business rules to be invoked then business objects are called from this server.

- **Business application server** – As of V2.2 of the product, the business component of the architecture can be installed as part of the Web application server or as a separate component. This means the business application server is also housed in a J2EE compliant Web application server *(Refer to the Supported Platforms section of the installation guide for J2EE Web application servers and versions supported*). This server can be run on a variety of supported Windows, Linux and Unix platforms (*Refer to the Supported Platforms section of the installation guide for operating systems and versions supported*). Within the business application server the following components are implemented:

  - **Business Objects** – The business logic for each object in the system is expressed as a COBOL or Java object. It contains all the SQL, programmatic rules and structures to manage the data for the transactions. In some products, COBOL is supported for our legacy customers and it is up to the site whether COBOL or Java is used for any extensions to the business objects.

  - **DB Connection Pool** – If any database access is required, we use an industry component called Hibernate to manage and pool the connections to the database. This will reserve connections and ensure efficient use of connections to the database.

- To access the database product uses the networking client provided by the DBMS vendors to ensure correct connection. For example, ORACLE provides SQL*NET, DB2 provides UDB Connect and SQL Server uses .NET drivers. These clients are multi-protocol for maximum flexibility.

- **Database Server** – The RDBMS used for the implementation is implemented in the database server. The product supports a number of databases (*Refer to the Supported Platforms section of the installation guide for databases and versions supported*). The database server only stores and retrieves the data for the product as all the business logic is in the business objects.

Roles and Features

Each tier in the architecture has a specific role in the operation of the product. The sections below outline the roles and features of each tier.

Contents

Client

The Browser User interface (BUI) is a combination of HTML and Java-script. AJAX, shorthand for Asynchronous JavaScript and XML, is a Web development technique for creating interactive Web applications. This makes web pages more responsive by exchanging small amounts of data with the server, so that the entire page does not have to be reloaded each time the user makes a change. This increases the Web page's interactivity, speed, and usability.

There are no ActiveX or Java components in the base product installation. This means that the deployment of the browser client is relatively simple as the only required component to use the product is a supported version of Internet Explorer on the client machine. If the implementation requires ActiveX controls for extensions then you can add them.

Note: If your implementation chooses to use the graphing component zones, then the latest version of the Macromedia Flash browser component must be installed. Refer to http://www.adobe.com/products/flashplayer/

The Browser tier of the product is provided for the end users to access the product on a desktop. The client provides the following roles in the architecture:

- **Screen Rendering and Caching** – All the screens are rendered using standard HTML and JavaScript (not Java). The rendering is performed as the screen is served from the Web Application server and stored in the local browser cache.
- **User Interaction** – The client provides the user with the screen interaction. After page is rendered the user can interact (manipulate data and screen elements) as per their business transaction. The browser client is responsible for ensuring that users can navigate and interact with the screen elements (e.g. resizing, display correctly).
- **User Context** – The product is stateless and therefore the client stores the transactional context locally and passes this to the transaction as required. The client records the context of the transaction in the browser memory.
- **Screen Cookie Management** – After a user is authenticated by the Web application server, the client is issued a JSR compliant cookie which is used as the credentials for all subsequent transactions. This cookie is a session cookie, which means it is stored in memory and removed when the browser is shutdown.

No business logic is stored on the client component.

Web Application Server

The product is a J2EE set of Web applications that are housed in a J2EE compliant Web application server. The product and the Web application server provide the following roles in the architecture:

- **Authentication** – The Web application server software that houses the product provides adapters to common security repositories. This means that security products interfaced to the Web application server software can be used in conjunction (with configuration) with the product.

- **Managing Client connections** – The Web application server software manages any client connections (during and after they are authenticated) for processing and availability.
- **Page Server** – The major responsibility of the Web application server is to "serve" pages to the client on demand. At startup time (or at the first request for a particular page) the product generates the screens dynamically using metadata and rendering style sheets. These are cached for reuse locally.
- **Cache management** – For performance reasons, the static data (usually metadata and configuration data) is cached in memory on the Web application server.

No business logic is stored on the Web application server component. The Web application server Component of the product is written in Java and JavaScript.

Business Application Server

The product is a J2EE set of business applications that are housed in a J2EE compliant Web application server (this can be the same instance of the Web application server or a separate one). The product and the business application server provide the following roles in the architecture:

- **Authorization** – After authentication has been performed by the Web application server, the Business Application server is responsible for determining which functions and which data can be accessed.
- **Data Integrity** – The business application server contains the business logic to maintain referential integrity for the product data.
- **Validation** – The business application server contains the business logic that contains all the validation rules for the product data.
- **Business Rules** – The business application server contains the business logic that implements business rules and performs calculations.
- **SQL** – The business application server contains all the SQL statements and formats and processes results from those SQL statements.

The business application server Component of the product is written in Java and may contain COBOL for backward compatibility.

Database Server

The product contains a database schema within a database management system. The database server has the following roles in the architecture:

- **Data Storage** – The database is responsible for efficiently storing all data.
- **Data Retrieval** – The database is responsible for efficiently retrieving data using SQL provided by the business application server.
- **Data Management** – The database is responsible for efficiently managing all data.

No business logic is stored on the Database Server.

Concepts

Before you attempt to configure or operate the product, there are important concepts that you should understand. These concepts are addressed in this document as a basis for the other documents in the technical documentation.

Contents

Environment

In a product implementation and post-implementation there will be a number of copies of the product installed. Each copy of the product is known as an environment. Each environment will be created for a specific purpose, according to your site plans, and accessible to a group of users deemed necessary for that purpose. For example, there will be at least one testing environment where designated personnel will perform their testing tasks.

For planning purposes an environment is a single instance of:

- The Web applications deployed in a J2EE Web application server.
- The business applications deployed in a J2EE Web application server. This can be the same physical J2EE Web application server or another instance (such as a separate server).
- A database containing the product schema. Physically, a schema can exist in an individual database instance or shared within a database instance (i.e. you can install multiple schemas of the product in the same database).

While there is no restriction on the number of environments it is recommended that the minimal number of copies of the product be installed using the guidelines outlined in the "Environment Management" document in the "Software Configuration Management" series on MetaLink (Refer to Other Documentation).

Administration User Id and Group

Prior to installing the product, you create a UNIX administration user ID and administration group. This account is used to install and operate the product. The product administration user ID and product group is provided as a parameter during the installation process. By default, the product administration user ID is **cissys** (**CCBUSER** parameter) and the group is **cissys** (**CCBGROUP** parameter). However, alternative values can be used according to your site standards.

The administration user ID is responsible for the following:

- It is the owner of the majority of the files installed for the product.

- It is the only user ID that should be used to run any of the administration tools provided with the product.
- It is the user ID that owns the UNIX resources used by the product. When the product is running, this user ID owns the processes associated with running the base software.

The administration user ID should be protected from unauthorized use. If components of the responsibility of administration need to be delegated to other users on the machine, we recommend not giving out the administration user ID. Instead, an alternative solution should be sought (such as using sudo or similar security tools).

The administration user ID should *not* be used for any of the following:

- As a product end user. By default, the administration user ID does not have access to the functionality of the product.
- To run product background processes.
- To manipulate data files exported from or imported into the product from any interfaces.

This technical document will refer to the administration user ID as **`cissys`**. If your site uses an alternative user ID as the administration user ID, substitute that user ID value for **`cissys`**.

Implementation Tip. It is possible to implement a different owner per environment in the product. Why would you want to do this? If you want to allow developers or testers to restart environments themselves, you can give access only to appropriate environments to distribute the administration. This can be achieved by installing the product with different userids. Note that you must log in and administrate each environment with its account only.

Directory Structure

In an effort to facilitate upgrades and ease maintenance, the product installation process creates a very specific directory hierarchy under the administration user ID of **`cissys`** (by default). The structure holds all the code, system products, scripts and temporary files that are created by the product during installation and operation.

Note. Every part of the product relies on the fact that this directory structure and the files within remain intact as delivered.

Note. At no time should you modify any of the supplied programs or scripts without the express direction of ORACLE.

There are two different directory structures that the product application uses:

- Base code directory structure known as ***`<SPLDIR>`***
- Application output directory structure / log directory known as ***`<SPLDIROUT>`***

Within each of the structures, there is a mount point and a subdirectory for each environment ***`<environment>`*** installed on the machine. The base mount point ***`<SPLDIR>`*** contains the environment directories that hold all of the application software for each particular environment. The application output mount point ***`<SPLDIROUT>`*** contains the environment directories that hold temporary files (such as the output from Doc 1) as well as batch log files. The default ***`<SPLDIR>`*** directory is **`/spl`** and the default ***`<SPLDIROUT>`*** directory is **`/spl/sploutput`**.

When a user logs on to a particular environment of the product either using the browser-based interface or directly on UNIX/Windows, the environment is set up (i.e. environment variables, etc.) to point to the appropriate directory structure under the mount point. The environment variable that points to an environment directory under ***<SPLDIR>*** is **$SPLEBASE** (or **%SPLEBASE%** in Windows). The environment variable that points to an environment directory under ***<SPLDIROUT>*** is **$SPLOUTPUT** (or **%SPLOUTPUT%** on Windows). The **SPLEBASE** and **SPLOUTPUT** environment variables are two of the standard environment variables used by the utilities provided with the product and runtime.

Implementation Tip. The actual location of the application directory ***<SPLDIR>*** and application output directory ***<SPLDIROUT>*** is up to site standards. The product does not care where it is installed as it internally uses the environment variables to access the correct locations.

The actual location for the mount points can differ per environment if you want. This is handy if you need to vary the location because you do not have enough space for all your non-production environments. Typically the number of environments during an implementation varies according to the level of access and desired amount of testing and training. The only restriction is that there can only be one location for **SPLEBASE** and **SPLOUTPUT** per environment.

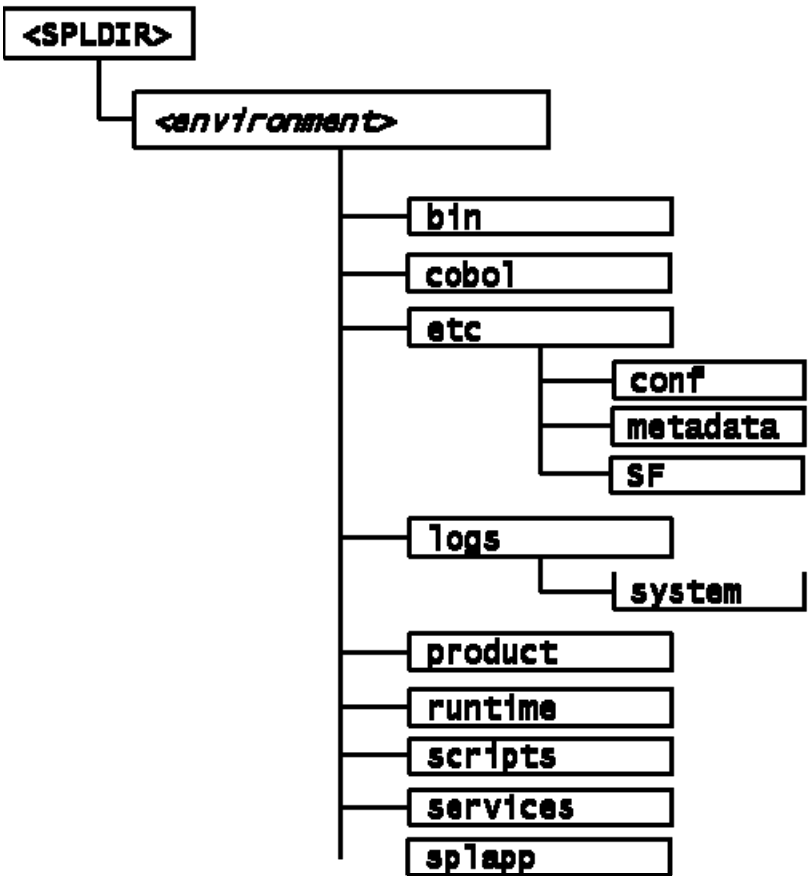
Contents

Software Directory Structure

The following components are stored in the base code directory structure:

- **Runtimes for Components** – All the runtime executables for the base software.
- **Business Object Binaries** – All the binaries that contain the business logic.
- **Configuration Files** – All the configuration files for the business objects and runtimes
- **Scripts** – Any administration or runtime scripts that are supplied to the customer.
- **Supported Plug-ins** – Source and executable for supplied plug-ins.

The following illustration depicts the layout of where the product code is placed upon installation into the file system:

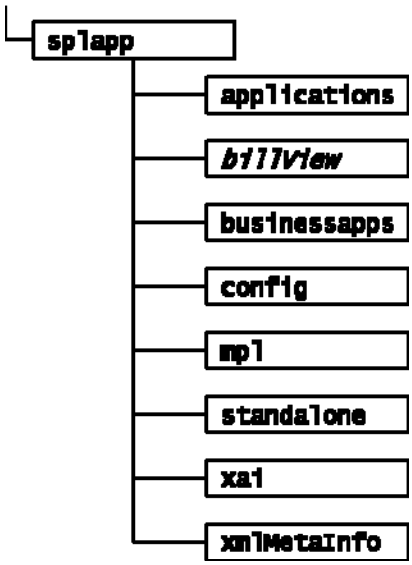


Contents of the directories:

Directory	Contents
bin	Utilities and commands for operations and configuration.
cobol	For products that support COBOL, a set of subdirectories that contain the source and object code for any supplied COBOL based plug-ins. Any compile output is also held in this structure. The source directory can be referenced by the environment variable **SPLSOURCE**. The build directory can be referenced by the environment variable **SPLBUILD**.
etc	A set of directories holding configuration files used in the product as well as template files used to generate the configuration files.
logs\system	Directory containing application logs files. This is independent of Web application server, business application server and Database Server log files.
product	Directories containing any bundled software with the product.
runtime	Directory containing any compiled objects for the product.
scripts	Directory containing any implementation specific scripts.

Directory	Contents
services	Directory containing COBOL source service definitions for the development kit and compilation
splapp	Directories containing the J2EE Web Applications

Under the splapp subdirectory for each environment there are a number of subdirectories:

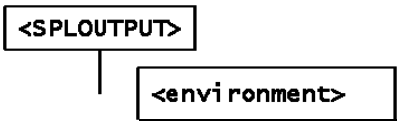


Directory	Contents
applications	Location of the Web application product files
billView	Location of the online bill viewing files
businessapps	Location of the business application product files
config	Location of temporary configuration files.
mpl	Location of the runtime and configuration for the Multi-Purpose Listener component
standalone	Location of common Java libraries and the batch component of the product
xai	Location of the Web services adapter configuration and Incoming service schemas
xmlMetaInfo	Location of the service definitions for the product.

Warning: Under no circumstances should files be manually altered in these directories unless instructed by ORACLE Support. The Oracle Utilities SDK will "deposit" files in the relevant locations in this structure using the Packaging component of the SDK or using the Development tools directly.

Output Structure

The product processes (batch and online) that produce output and logs place information in this directory structure. The environment directories are referenced by the variable **SPLOUTPUT**. By default, this directory is created as **/spl/splapp**, though an alternative may be used during the installation process.



Environment Variables

The product uses a number of environment variables to determine where information is stored and to be placed for its internal operations. Becoming familiar with these variables will assist you in finding information quickly and efficiently.

Note. If a custom script is written to access or write information to the product functionality, it is highly recommended that the following variables be referenced in your scripts. This is to maximize the chance that your script will remain functional across upgrades.

The following table outlines all the variables:

Variable	Usage	source /(Default value)
ANT_HOME	Location of ANT build utilities	Built by **splenviron** utility
ANT_OPTS	Options for ANT (*Development only*)	Built by **splenviron** utility
appViewer	Name of AppViewer Web application	Built by **splenviron** utility (**appViewer**)
BEADIR	Location of ORACLE WebLogic (*Weblogic supported platforms only*)	Logon profile
C3P0_JAR_DIR	Location of C3P0 Java library	Logon profile
CISTABFILE	Location and name of global configuration file	Built by **splenviron** utility
CMPDB	Database Type	Internal use only
COBDIR	Location of COBOL runtime	(**/opt/SPLcobAS50**)
DB2CODEPAGE	DB2 Code Page (*DB2 supported platforms only*)	Built by **splenviron** utility
ENVFILE	Location and name of environment configuration file	Built by **splenviron** utility
help	Name of online help Web application	Built by **splenviron** utility

Variable	Usage	source /(Default value)
		(**help**)
HIBERNATE_JAR_DIR	Location of Hibernate Java library	Logon profile
JAVA_HOME	Location of JDK	Logon profile
JROCKIT_HOME	Location of JRockit (*JRockit supported platforms only*)	Logon profile
OAS_HOME	Location of ORACLE Application Server (*OAS supported platforms only*)	Logon profile
ONLINEBILLINI	Location of DOC1 configuration files (*applicable to DOC1 supported platforms only*)	Built by **splenviron** utility
ONLINEDOCINI	Location of DOC1 configuration files (*applicable to DOC1 supported platforms only*)	Built by **splenviron** utility
ORACLE_SID	ORACLE database name	Built by **splenviron** utility
NLS_LANG	ORACLE language Settings	Built by **splenviron** utility
SPLADMIN	Administration user ID	Built by **splenviron** utility (**cissys**)
SPLADMINGROUP	Administration group	Built by **splenviron** utility (**cisusr**)
SPLAPP	Name of root Web application	Built by **splenviron** utility (**root**)
SPLBUILD	Location of COBOL build directory	Built by **splenviron** utility
SPLCOBCPY	Location of COBOL copy code libraries	Built by **splenviron** utility
SPLCOMP	Name of COBOL compiler vendor	Built by **splenviron** utility (**microfocus**)
SPLDB	Database type	Built by **splenviron** utility
SPLEBASE	Location of software for environment	Built by **splenviron** utility
SPLENVIRON	Name of environment	Built by **splenviron** utility
SPLOUTPUT	Location of output for environment	Built by **splenviron** utility

Variable	Usage	source /(Default value)
SPLRUN	Location of runtime for environment	Built by **splenviron** utility
SPLSDKROOT	Location of SDK (*Development environment only*)	Built by **splenviron** utility
SPLSOURCE	Location of COBOL source	Built by **splenviron** utility
SPLSYSTEMLOGS	Location of product specific logs	Built by **splenviron** utility
SPLVERSION	Version identifier of product	Built by **splenviron** utility
SPLVERSIONFILE	Location of version file	Built by **splenviron** utility
SPLWEB	Location of installed Web applications	Built by **splenviron** utility
SPLWAS	Web application Server type	Built by **splenviron** utility
WAS_HOME	Location of IBM WebSphere software (*WebSphere supported platforms only*)	Logon profile
WEB_isExpanded	Whether Web application is expanded or not (not = WAR/EAR files)	Built by **splenviron** utility (**false**)
WEB_OASAPP	Name of Web application within OAS (*OAS supported platforms only*)	Built by **splenviron** utility
WEB_OC4J_instance	Name of OC4J instance to deploy the application to (*OAS supported platforms only*)	Built by **splenviron** utility
WEB_SVRNAME	Name of Web application server within WebSphere (*WebSphere supported platforms only*)	Built by **splenviron** utility
WEB_WASAPP	Name of Web application within WebSphere (*WebSphere supported platforms only*)	Built by **splenviron** utility
WEB_WASUSER	Name of administration user ID within WebSphere (*WebSphere supported platforms only*)	Built by **splenviron** utility
WL_HOME	Location of ORACLE WebLogic installation (*WebLogic supported platforms only*)	Built by **splenviron** utility
WLVERSION	Version of ORACLE WebLogic installed (*WebLogic supported platforms only*)	Built by **splenviron** utility

Variable	Usage	source /(Default value)
XAIApp	Name of Web services adapter Web application.	Built by **splenviron** utility (**XAIApp**)

Note: If a custom script is written to access or write information to the product functionality, it is highly recommended that the following variables be referenced in your scripts. This is to maximize the chance that your script will remain functional across upgrades.

Note: **HIBERNATE_JAR_DIR** and **C3P0_JAR_DIR** are used for the installation process only. After installation is complete the jar files located at the locations specified by these environment variables are copied to the correct implementation locations.

Common Application Logs

When the product is operating the infrastructure logs messages within its own logs. For example, the database will log database errors or messages to the database logs, the J2EE Web application server will log Web Application errors or messages to the J2EE Web application server logs and so on. The name and location of these logs is set by relevant vendors of those logs. Refer to the documentation provided with that software on where logs are stored and their logging conventions.

The product additionally writes a number of application specific logs to **$SPLSYSTEMLOGS** (or **%SPLSYSTEMLOGS%** on Windows):

- **spl_web.log** - Web application server application messages .
- **spl_service.log** – business application server messages. If the business application server exists on the same host as the Web application server for an environment then this log does not exist and all messages are written to the **spl_web.log**.
- **spl_xai.log** – Web Services Adapter messages.

The format of all logs is as follows:

Field	Comments
<userid>	User ID of transaction (blank or "-" for system generated messages)
<pid>	Process identifier (optional)
<time>	Time of transaction in format HH:MM:SS,SSS
[<transaction>]	Transaction/Class identifier
<type>	Type of message
(<class>)	Java class generating message (see Javadocs in **appViewer**)

<message>	<message contents>

Examples:

```
19:03:16,390 [main] INFO  (support.context.CacheManager) Registering cache
'MenuRepository'
-  19:02:37,812 [main] INFO  (support.context.ContextFactory) 461 services
registered, time 11.742 ms
- 19:03:29,140 [Remote JVM:2 Thread 1] WARN  (cobol.mem.CobolModeHelper)
Unspecified or unrecognized COBMODE (null) - inspecting JVM properties to determine
bit mode ...
19:03:40,875 [Thread-24] ERROR (web.dynamicui.MetaDataHolder) Unable to find UI xml
file '/an/generated/toDoSummaryListGrid.xml' for program 'toDoSummaryListGrid'
DEMO - 259992-101-1 19:17:38,750 [http-6500-5] INFO  (support.context.CacheManager)
Registering cache 'UiMapInfoCache'
```

Attaching to an Environment

Before performing any command against a product environment, you must *attach* to the environment. Attaching to an environment sets system and environment variables so that the correct runtime and code is used in the execution of subsequent commands.

To attach to an environment:

- Make sure that you are logged in using the administration account for the desired environment, for example **cissys**.
- Execute the following command:

```
<SPLDIR>/<environment>/bin/splenviron.sh –e <environment>
```

Or

```
<SPLDIR>\<environment>\bin\splenviron.cmd –e <environment>
```

Where **<SPLDIR>** is the mount point defined for the product and **<environment>** is the name of the environment to access.

Note: This command must be run *before* any UNIX-based command (including running the product background processes) to ensure that the correct environment is in place.

Note: If you are running multiple versions of the product, ensure that you run the correct version of the **splenviron.sh**/**splenviron.cmd** utility for the environment by manually changing to the directory where the **splenviron.sh**/**splenviron.cmd** utility exists for the desired environment prior to running the command.

The following is an example of **splenviron.sh** execution:

```
$ /spl/DEMO/bin/splenviron.sh –e DEMO

Version ................ (SPLVERSION) : V2.2.0
Database Type ............... (SPLDB) : oracle
ORACLE_SID ............. (ORACLE_SID) : DEMODB
NLS_LANG ................ (NLS_LANG) : AMERICAN_AMERICA.WE8ISO8859P15
Environment Name ....... (SPLENVIRON) : DEMO22
Environment Code Directory (SPLEBASE) : /spl/DEMO
App Output Dir - Logs ... (SPLOUTPUT) : /spl/sploutput/DEMO
```



```
Build Directory .......... (SPLBUILD) : /spl/DEMO/cobol/build
Runtime Directory .......... (SPLRUN) : /spl/DEMO/runtime
Cobol Copy Path ......... (SPLCOBCPY) :
/spl/DEMO/cobol/source/cm;/spl/DEMO/services;/spl/DEMO/cobol/source
```

The above example summary of the command illustrates that important environment variables and their values are set. Use this information to confirm that you have successfully attached to the correct environment.

Utilities

The product includes several command scripts to aid with its configuration and operation. This section provides information about these utilities.

Contents

splenviron – Set Environment variables

The **splenviron.sh**/**splenviron.cmd** utility initializes a defined set of environment variables and paths for an environment. This script must be run before any other script or utility is run within the environment.

Command Usage:

Linux/Unix:

```
splenviron.sh -e <environment> [-c <command>] [-q] [-h]
```

Windows:

```
splenviron.cmd -e <environment> [-c <command>] [-q] [-h]
```

-e ***<environment>***	***<environment>*** is the environment id as installed in the **cistab** file.
-c ***<command>***	Execute ***<command>*** after running **splenviron**. Command must be enclosed in "". Default is shell (e.g. **ksh**).
-q	Quiet Mode. Do not show output from command. Any output from the –c command will be shown.
-h	Show usage.

Examples:

```
splenviron.sh –e DEMO
splenviron –e DEV
splenviron.sh –e DEMO –c "cat file.lst"
```

The **splenviron.sh**/**splenviron.cmd** utility is executed whenever an environment needs to be initialized. One of the options to this script allows system administrators to optionally include the execution of an additional command as part of the **splenviron.sh**/**splenviron.cmd** environment initialization. This enables the system administrator to more finely tune the environment shell so they can change such settings as TimeZone, **PATH** or environment variables.

Extending the splenviron Command

If your implementation needs to add environment variables (or modify existing variables) for a third party product you may wish to integrate with that product. For example, you might want to add some custom Java classes from a component that you want to use with the product.

When you run the **splenviron.sh**/**splenviron.cmd** utility it sets the environment variables for the environment. These are standard variables as well as any required for operation of the product. For example, there are variables that can be used in utilities so they can be used across environments.

These environment variables can be extended (or added to) using one of the following options:

- Change to ALL environments on machine - If your integration is common across all environments then you can set or alter environment variables using the following technique:
    - Create a script in a central location on the machine that sets or alters the appropriate environment variables. Ensure that the product administrator user ID has read/execute access to the location and the script.
    - Set the **CMENV** environment variable with the location and name of the script to execute prior to running the **splenviron.sh**/**splenviron.cmd** utility (for example, in your logon profile).
    - When the **splenviron.sh**/**splenviron.cmd** utility is run it will detect the script specified in the **CMENV** environment variable and execute the script to set or alter the environment variables.
- Change to a specific environment on machine - If your integration is specific to an environment (or different for each environment, for example if you have a development as well as a test copy of the third party product) then you can set or alter environment variables using the following technique:
    - Create a script called **cmenv.sh** (or **cmenv.cmd** on Windows) in scripts subdirectory of the environment (usually **$SPLEBASE/scripts** or **%SPLEBASE%\scripts**). Ensure the permissions are set appropriately for the product administration account to execute the script.
    - When the **splenviron.sh**/**splenviron.cmd** utility is run it will detect the **cmenv.sh** script (or **cmenv.cmd** on Windows) and execute the script to set or alter the environment variables at the end of the **splenviron.sh**/**splenviron.cmd** utility.
- Combination of both previously outlined options – It is possible to combine the techniques in a combination which can mean you can have maximum flexibility. If you follow the instruction of both techniques then the following will happen in the following order:
    - When the **splenviron.sh**/**splenviron.cmd** utility is run it will detect the script specified in the **CMENV** environment variable and execute the script to set or alter the environment variables.

- If there is a **cmenv.sh** script (or **cmenv.cmd** on Windows) in the scripts subdirectory of the environment, it will execute the script to set or alter the environment variables. This may override, add or alter environment variables already set.

Note: In using this technique remember:

If you alter any pre-existing environment variables then ensure your changes are not going to circumvent product requirements. For example, do not alter paths used by the product.

If you add files or directories to library variables or **CLASSPATH** ensure your changes are suffixed at the end of the variable. This is especially important for java classes as classes you use may conflict with product supplied ones; adding them at the end of the **CLASSPATH** will minimize the effects of conflicts.

Do not remove any environment variables used by the product.

configureEnv – Setup Environment settings

The **configureEnv.sh**/**configureEnv.cmd** utility is an interactive method for configuring an environment on the system stored in the **etc/ENVIRON.INI**. This configuration script sets up important parameters used by other scripts within the system.

Normally this script is executed without parameters and the current environment (i.e., the environment that you are currently attached to) is configured.

Command Usage:

Linux/Unix:

```
configureEnv.sh -e <environment> [-h]
```

Windows:

```
configureEnv.cmd -e <environment> [-h]
```

-e ***<environment>***	***<environment>*** is the environment id as installed in the **cistab** file.
-h	Show usage.

Examples:

```
configureEnv.cmd –e DEMO
```

Refer to ENVIRON.INI - Environment Configuration File for more information on the output of this command.

spl – Start/Stop Environment

Note: The **splenviron.sh**/**splenviron.cmd** utility must be executed *before* this utility can be used. See Attaching to an Environment for details.

The `spl.sh`/`spl.cmd` utility is used to start up and shut down an environment or individual components (web server or multi-purpose listener) of an environment. Usage of this utility is optional as described in the sections Contents

Starting an Environment and Stopping an Environment.

Use the command without a parameter to start up, reboot or shut down all components of an environment (note that the action must still be used). To start up or shut down an individual component, use the option that specifies that applies to that specific component.

Command Usage:

Linux/Unix:

```
spl.sh [-h | -w | -m ] <action>
```

Windows:

```
spl.cmd [-h | -w | -m ] <action>
```

`-h` Show usage.

`-w` Perform ***`<action>`*** on Web application server/Business application server only

`-m` Perform ***`<action>`*** on Multi-Purpose Listener (MPL) only

If no option specified, then command applies to all components.

Note: MPL is only started if enabled.

`<action>` **`start`** – start the component/environment

`stop` – stop the component/environment

`reboot` – stop then start the component/environment

`isup` – Check component/environment is running (*applies to ORACLE WebLogic supported platforms only*)

When using the **`isup`** action, the script returns:

Return Code ($?)	Comments
0	Environment is unavailable
1	Environment is available

Example

Action	Linux/Unix Command	Windows Command
To start an environment	**spl.sh start**	**spl start**
To stop an environment	**spl.sh stop**	**spl stop**
To reboot (stop and then start) an environment	**spl.sh reboot**	**spl reboot**
To start the MPL only	**spl.sh –m start**	**spl –m start**

genappvieweritems – generate AppViewer

Note: The **splenviron.sh**/**splenviron.cmd** utility must be executed *before* this utility can be used. See Attaching to an Environment for details.

If the environment is used for reference or development then it may be necessary to regenerate the **AppViewer** from the metadata. A utility is provided that runs a number of provided background processes to regenerate the **appViewer** from the current environment.

Command Usage:

Linux/Unix:

```
genappvieweritems.sh
```

Windows:

```
genappvieweritems.cmd
```

Examples:

genappvieweritems.cmd

```
…
Application Viewer is delivered with the system including cobol source code
and xml services. This script will extend Application Viewer capabilities
on site by generating additional items.

The Following Programs will be ran
F1-AVALG     Generate XML file(s) for Algorithm data
F1-AVMO      Generate XML file(s) for Maintenance Object data
F1-AVTBL     Generate XML file(s) for Table data
F1-AVTD      Generate XML file(s) for To Do Types XML
F1-AVBT      Generate XML file(s) for Batch Control Types XML

The Application EAR file will also be re-created if required.
Proceed (Y/N)?
…
Calling F1-AVALG
program F1-AVALG got a 0 response code
Calling F1AVMO
program F1-AVMO got a 0 response code
```



```
Calling F1-AVTBL
program F1-AVTBL got a 0 response code
Calling F1AVTD
program F1-AVTD got a 0 response code
Calling F1-AVABT
program F1-AVABT got a 0 response code
If you received a non response code 0 above, you should consult the logfiles
```

Note: For platforms that use EAR/WAR files, the **genappvieweritems** utility will automatically rebuild the EAR/WAR files ready for deployment (deployment will need to be performed if **WEB_isAppViewer** is set to **true**).

This generates the HTML files to be included in the **appViewer** application. This will only generate the necessary files from the current environment. To deploy the **appViewer**, the relevant **gendeploy*** command must be executed.

genoasdeploy – Deploy on ORACLE Application Server

Note: The **splenviron.sh**/**splenviron.cmd** utility must be executed *before* this utility can be used. See Attaching to an Environment for details.

Note: This command is provided as an alternative to the utilities provided by the Web application server vendor (in fact, it calls the command line interface provided by the vendor).

Note: This utility is only applicable to ORACLE Application Server implementations.

Note: This script is only used in environment where **Expanded Mode** is **false**. Refer to ENVIRON.INI - Environment Configuration File for more details.

The **genoasdeploy.sh**/**genoasdeploy.cmd** utility deploys the Web applications and business application server to the ORACLE Application Web application server. The utility uses values from the **ENVIRON.INI** (refer to ENVIRON.INI - Environment Configuration File) to determine the locations to deploy the application within. This script is typically used to reflect any configuration, code or base changes in the Web application server and business application server components.

Command Usage:

Linux/Unix:

```
genoasdeploy.sh
```

Windows:

```
genoasdeploy.cmd
```

Examples:

genoasdeploy.sh
…


```
Proceed with the Installation of Web Server Application myWebApp:testing into
OracleAppServer on tugbu.web.oracle.com ? Y/N :
…
Proceed with the Installation of Business Server Application myBusApp:testing into
OracleAppServer on tugbu.bas.oracle.com ? Y/N :
…
```

genoc4jdeploy – Deploy on OC4J

Note: The **splenviron.sh**/**splenviron.cmd** utility must be executed *before* this utility can be used. See Attaching to an Environment for details.

Note: This command is provided as an alternative to the utilities provided by the Web application server vendor (in fact it calls the command line interface provided by the vendor).

Note: This utility is only applicable to ORACLE OC4J implementations for SDK use on Windows only.

Note: This script is only used in environment where **Expanded Mode** is **false**. Refer to ENVIRON.INI - Environment Configuration File for more details.

The **genoc4jdeploy.cmd** utility deploys the Web applications and business application server to the ORACLE OC4J Web application server. The utility uses values from the **ENVIRON.INI** (Refer to ENVIRON.INI - Environment Configuration File) to determine the locations to deploy the application within. This script is typically used to reflect any configuration, code or base changes in the Web application server and business application server components.

Note: If the product is already deployed on the target J2EE Web application server instance then this script will also undeploy the application prior to redeployment.

Command Usage:

Windows:

```
genoc4jdeploy.cmd
```

Examples:

```
genoc4jdeploy.cmd
…
Deploying the splWeb EAR file…
Deploying the splService EAR file…
…
```

genupdatewar – Generate J2EE WAR and EAR files

Note: The **splenviron.sh**/**splenviron.cmd** utility must be executed *before* this utility can be used. See Attaching to an Environment for details.

Note: This script is only used in environment where **Expanded Mode** is **false**. Refer to ENVIRON.INI - Environment Configuration File for more details.

The **genupdatewar.sh**/**genupdatewar.cmd** utility builds the WAR and EAR files used by the product. This script is used to reflect changes in the product WAR/EAR files from configuration changes, code based customizations and/or base code changes (from patches and upgrades). This utility uses ANT (http://ant.apache.org/) to build the WAR/EAR files.

For a description of the WAR or EAR format, refer to http://en.wikipedia.org/wiki/EAR_(file_format), http://en.wikipedia.org/wiki/WAR_(Sun_file_format) or http://java.sun.com/javaee/5/docs/tutorial/doc/bnaby.html#indexterm-47.

Command Usage:

Linux/Unix:

```
genupdatewar.sh
```

Windows:

```
genupdatewar.cmd
```

Examples:

```
genupdatewar.sh
…
080807.12:39:33 <info> genupdatewar.sh : Generate Environment Started Thu Aug  7
12:39:33 PST 2008
080807.12:39:33 <info> Environment DEMO
080807.12:39:33 <info> Update war with the latest environment configuration.
080807.12:40:36 <info> Build servicebean jar
080807.12:40:42 <info> Build ear files
```

genwasdeploy – Deploy on WebSphere

Note: The **splenviron.sh**/**splenviron.cmd** utility must be executed *before* this utility can be used. See Attaching to an Environment for details.

Note: This command is provided as an alternative to the utilities provided by the Web application server vendor (in fact it calls the command line interface provided by the vendor).

Note: This utility is only applicable to IBM WebSphere implementations.

Note: This script is only used in environment where **Expanded Mode** is **false**. Refer to ENVIRON.INI - Environment Configuration File for more details.

Note: This utility calls the genwasstubs.sh – Generate WebSphere Stubs for business application server utility automatically.

The **genwasdeploy.sh** utility assists in the deployment of the Web Applications and business application server to the WebSphere Web application server. The utility uses values from the **ENVIRON.INI**, (refer to ENVIRON.INI - Environment Configuration File) to determine the locations to deploy the application within. This script is typically used to reflect any configuration, code or base changes in the Web application server and business application server components.

Note: If the product is already deployed on the target IBM WebSphere Web application server instance then this script will also undeploy the application prior to redeployment.

Command Usage:

Windows:

```
genwasdeploy.sh
```

Examples:

genwasdeploy.sh

```
…
Connecting to WebSphere Server: myWebApp hosted on tugbu.web.oracle.com to find
SOAP PORT for Web Server: testing
…
Proceed with the Installation of Web Server Application myWebApp:testing into
WebSphere on tugbu.web.oracle.com ? Y/N :
…
Proceed with the Installation of Business Server Application myBusApp:testing into
WebSphere on tugbu.bas.oracle.com ? Y/N :
…
```

genwasstubs.sh – Generate WebSphere Stubs for business application server

Note: This utility is not intended to be executed by itself. It is provided as a service script to be used by the genwasdeploy – Deploy on WebSphere utility.

Note: This command is provided as an alternative to the utilities provided by the Web application server vendor (in fact, it calls the command line interface provided by the vendor).

Note: This utility is only applicable to IBM WebSphere implementations.

Note: This script is only used in environment where **Expanded Mode** is **`false`**. Refer to ENVIRON.INI - Environment Configuration File for more details.

Additionally for WebSphere the **`genwasstubs.sh`** utility is called automatically prior to deploying the business application server EJB to the WebSphere Web application server. The utility uses values from the **`ENVIRON.INI`**, (refer to ENVIRON.INI - Environment Configuration File) to determine the locations to deploy the application within. This script is typically used to reflect any configuration, code or base changes in the business application server components.

Command Usage:

Windows:

```
genwasstubs.sh
```

Examples:

genwasstubs.sh


```
…
080808.14:40:36 <info> Build servicebean jar
080808.14:40:42 <info> Build ear files
…
```

initialSetup – Reset configuration to template

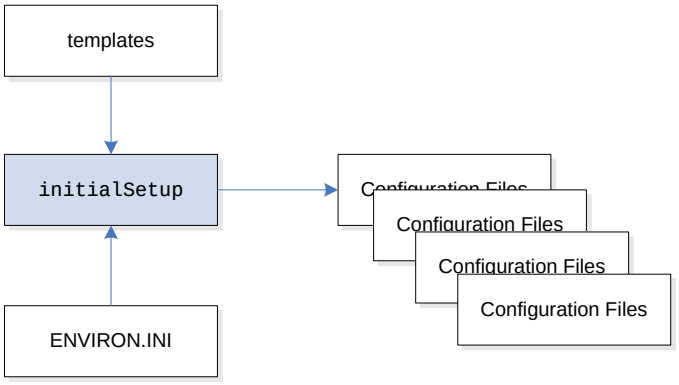
Note: The **splenviron.sh**/**splenviron.cmd** utility must be executed *before* this utility can be used.

Warning: This command will reset *all* configuration files to template settings. Any customization to configuration files will be lost. Backup configuration files prior to running this script.

During the installation process a number of configuration files used by the components of the architecture are built to be used by the various components of the architecture. The utility takes the ENVIRON.INI - Environment Configuration File settings and using a set of provided templates (located in the **etc** directory), builds the necessary configuration files for the product components.

Note: This utility not only builds the configuration files it also builds the WAR/EAR file ready for deployment to the J2EE Web application server. The files must be deployed before any changes are effective.

While this utility is used at installation time, it can also be used to reset the configuration files to the original settings as well as reflect changes to the ENVIRON.INI - Environment Configuration File. This concept is shown in the diagram below:



Command Usage:

Linux/Unix:

```
initialSetup.sh
```

Windows:

```
initialSetup.cmd
```

Examples:

```
initialSetup.sh

080807.02:37:33 <info> Template generation step.
```

```
080807.02:37:43 <info> FW template generation step.
080807.02:37:43 <info> Create war file for SPLApp.war.
080807.02:38:14 <info> Create war file for XAIApp.war.
080807.02:38:26 <info> Create war file for appViewer.war.
080807.02:39:14 <info> Create war file for help.war.
080807.02:41:11 <info> FINISHED INITIAL INSTALL SETUP at Thu Aug  7 02:41:11 EST
2008
080807.02:41:11 <info> See file /spl/TRAINING/logs/system/initialSetup.sh.log for
details
```


Common Operations

There are a number of common operations that a site will perform on the product. This section outlines the steps involved in these common operations.

Contents

Starting an Environment

Note: This section will outline a particular method for starting the product using the supplied utilities. Sites can use the consoles and utilities provided by the Web application server/database vendors to start the product as an alternative.

To ensure a successful startup of the product the components should be started in the following order:

- The database server must be started according to local standards. This includes any communications software such as listeners to enable the product to communicate to the database. After starting the database server, the batch interface can be used.
- The business application server must be started to enable the web application server to use the business objects and the business object conduit to accept web transactions.
- The web application server must be started to enable web clients to access the screens and business objects. After starting the business application server and the web application server, the XAI incoming calls, the batch interface, and online users have access to the system.
- The end users can start the browser to access the product front-end screens.
- Optionally, if the Multi-purpose Listener (MPL) is configured correctly it is also started to support outgoing XAI transactions as well as enable incoming calls from JMS and File.

Contents

Starting All Tiers on a Single Server

If the business application server and web application server for an environment are on the same physical machine they can be started using the following set of tasks:

- Start the database using the utilities provided by the database vendor.
- Log on to the server containing the Web application server and Business application server using the product Administration account for the desired environment (for example, **`cissys`**).
- command:

Linux/Unix:


```
<SPLDIR>/<environment>/bin/splenviron.sh –e <environment>
```

Windows:

```
<SPLDIR>\<environment>\bin\splenviron.cmd –e <environment>
```

Where **<SPLDIR>** is the mount point defined for the product and **<environment>** is the name of the environment to start.

- Start the Web application server, business application server and MPL using the following command:

Linux/Unix:

```
spl.sh start
```

Windows:

```
spl.cmd start
```

Refer to the spl – Start/Stop Environment utility for more options.

Note: As an alternative, it is possible to start the Web application server and business application tiers using the console or utilities provided with the J2EE Web application server software.

The script will display the startup messages as dictated by the J2EE Web application server vendor.

To restart a business application server and web application server replace the action **start** with **reboot**.

Starting/Stopping at Boot Time (UNIX/Linux)

One of the implementation questions that may arise is to start all the environments at UNIX/Linux boot time. This is possible by writing a script and placing it in **/etc/init.d** (or equivalent for your platform). A suggested standard is to provide a script that takes a parameter of start or stop. The script could then be used to start or stop product environments on the machine:

```
#!/usr/bin/ksh
#
# Purpose  : Start/Stop all copies of the product on a machine
#

Usage() {
  echo "Usage :"
  echo "  $0 [start|stop]"
  exit 1
}

#---------------------------------------------------------------
#
#################################################################
#
# Main

# check command line arguments
if [ "$#" -eq "0" ]
then
 Usage
```



```
 exit 1
fi

# Loop through all environments in /etc/cistab

if [ ! -f /etc/cistab ]
then
 echo "/etc/cistab file does not exist. Product is not installed correctly"
 exit 1
fi

cat /etc/cistab | while IFS=: read _env _filler1 _splebase _splapp _filler2 _start
do
  # Only environments with the start parameter set to Y should be started
  if [ ${_start} = "Y" ]
  then
      if [ -d ${_splebase} ]
      then
      # Determine owner of the environment
      export PRDOWNER=`cat ${_splebase}/etc/ENVIRON.INI | grep SPLUSER | cut –f2 –
d"="`

      # Format start command

       _startcmd="${_splebase}/bin/splenviron -e {_env} -c ""spl.sh start"""
       _stopcmd="${_splebase}/bin/splenviron -e {_env} -c ""spl.sh stop"""

       # Run command

       case $1 in
          "start")  su – $PRDOWNER -c "${_startcmd}" ;;
           "stop")  su – $PRDOWNER -c "${_stopcmd}" ;;
                *)  Usage
                    exit 1;;
       esac
     fi
  fi
done
# Finished
```

Note. The above script is provided as a sample only. Use the above script as an example for any custom scripts to start the product at boot time.

What to Look for in Startup

As outlined in Common Application Logs the application logs all information to application logs during the startup, operation and shutdown of the application. These logs can be used to check that the startup of the product is successful. The logs contain the following sections for a startup (class indicates startup message):

- The Web Application is initialized (class = *web.startup.SPLWebStartup*) within the J2EE Web application server.
- Configuration Settings are loaded from the relevant configuration files (class = *shared.environ.ApplicationProperties*).

- The product is set to Production mode (this denotes Development versus Production settings) (*class = shared.context.ApplicationMode*). Most installations are "Production" mode. Only environments where the Oracle Utilities SDK is used will not be in "Production" mode.
- The state of compression is verified (*class = web.dynamicui.TransformServletHelper*). Refer to Web application server Configuration for details of this setting.
- The framework used by the product is initialized and settings within the framework are prepared to be loaded (*class = support.context.ContextFactory*).
- The metadata is loaded into memory for configuration control (*class = shared.context.ContextLoader*).
- Any checks for any customizations (*class = shared.environ.ContextManagedObjectSet*). In most cases, environments that *do not* have any product customizations will report a warning about a resource not loading. This can be ignored.
- Any lookups are loaded into memory (*class = support.context.ComponentContainerLookupHelper*). Lookups are metadata used to enumerate valid values for flags, common values etc.
- Additional metadata is loaded into memory (*class = support.context.ContextFactory*). The metadata used to configured the product includes entities, Code Descriptions, algorithms, batch controls, components, Change Handlers and COBOL objects (if used).
- Hibernate ORM mappings used by the product are loaded (*class = support.context.ApplicationContext*). The number of mappings will vary between releases and parts of the product that are used.
- The connection pool to the database is initialized according to the configuration settings (*class prefix hibernate.**). If the connection information is incorrect or the database is down the connection pool connection will retry (according to the configuration settings). If this is the case you will see the connection information and error messages, such as "**`Connections could not be acquired from the underlying database!`**", in this log.

  Note: The messages seen will vary depending your database type and version.

- A successful database connection is shown in the message "**`Done building hibernate session`**" (*class = support.context.ApplicationContext*). A number of additional messages may appear as dictated by the database vendor to indicate versions and connectivity information.
- The database statement cache is initialized within the product (*class = support.sql.PreparedStatementImpl and class = support.context.CacheManager*).
- The owner of the system is initialized. This identifies the application owner for implementation purposes. In all cases the implementation value is "CM" for Custom Modification. Other values are supported for ORACLE internal use only.
- If COBOL is used for the product then the COBOL Child (or "Worker") Java Virtual Machines (JVM) are initialized (*class = cobol.host.CobolHostStartup*). During the startup of the JVM's various startup messages will indicate the status of each JVM startup (*class prefix cobol.host*). Each JVM will have individual messages outlining loading and startup of the JVM for COBOL/java integration (*JVM number is indicated in the message*). Completion of COBOL loading is indicated by message "**`Remote JVM setup complete`**" (*class = cobol.host.RemoteJVM*). As COBOL components are detected additional messages will appear in the log to load additional metadata necessary for the execution of the COBOL/java interface (*class prefix support.cobol and cobol.mem*).

- The Web application server/business application server static cache is then loaded (*class = api.globalContext.GlobalContextHelper*) which includes:
    - Preloading language settings (*class = web.startup.PreloadLoginInfo*). If preloading is enabled then the progress of preloading is shown on the startup log. Preloading ends with message "**`XSLT main preload`**" (*class = web.startup.PreloadLoginInfo*).
    - Loading product based style sheets (XSL) for screen generation.
    - Navigation Keys (for static menus and context sensitive menus) (*class = web.dynamicui.NavigationInfoCache*)
    - Metadata is loaded as indicated (class = *support.context.CacheManager*)
    - Service Interceptors are loaded (*class = api.serviceinterception.InterceptorRepository*)
    - Menus are loaded (*class = domain.web.MenuLoginService*)
    - Navigation information is loaded (*class = domain.web.SystemLoginInfoHelperService*)
    - Service definitions are loaded (*class = service.metainfo.MetaInformationRepository*)
    - Installation record defaults are loaded (*class = web.common.WebInstallationDataHelper*)
- If the online batch daemon is enabled then the daemon is loaded into memory and started (*class = grid.node.DistributedGridNode and prefix grid.space*). Any work to be detected will result in additional messages (*class = grid.node.WorkProcessor*).
- The Web service adapter (XAI) component is then loaded (delay is configurable) with similar messages as the root application startup. Refer to the top of this list to reference the messages that are loaded.

Once the application is loaded the J2EE Web application server will indicate the product is available (the message for this varies – refer to the J2EE Web application server documentation for details).

Stopping an Environment

Note: This section will outline a particular method for starting the product using the supplied utilities. Sites can use the consoles and utilities provided by the Web application server/Database vendors to start the product as an alternative.

To ensure a successful shut down of the product the components should be stopped in the following order:

- The end users should shut down the browser containing the product front-end screens.
- The MPL must be shutdown (if used) to prevent outgoing XAI transaction from being processed.
- The Web application server must be shutdown to disable web clients' access to the system. After the web application server is shutdown, end users do not have access to the system but batch processes may still run.
- The business application server must be shutdown to disable the Web application server completely.

- The database server must be shut down according to local standards. This includes any communications software such as listeners to enable the product to communicate to the database. At this point all users (batch and online) do not have access to the environment.

Contents

Stopping All Tiers on a Single Server

If the business application server and web application server for an environment are on the same physical machine they can be stopped/shutdown using the following set of tasks:

- Logon to the server containing the Web application server and business application server using the product Administration account for the desired environment (for example, **cissys**).
- command:

Linux/Unix:

```
<SPLDIR>/<environment>/bin/splenviron.sh –e <environment>
```

Windows:

```
<SPLDIR>\<environment>\bin\splenviron.cmd –e <environment>
```

Where **<SPLDIR>** is the mount point defined for the product and **<environment>** is the name of the environment to stop.

- Start the Web application server, business application server and MPL using the following command:

Linux/Unix:

```
spl.sh stop
```

Windows:

```
spl.cmd stop
```

Refer to the spl – Start/Stop Environment utility for more options.

Note: As an alternative, it is possible to stop the Web application server and business application tiers using the console or utilities provided with the J2EE Web application server software.

The script will display the shutdown messages as dictated by the J2EE Web application Server vendor.

- Stop the database using the utilities provided by the database vendor.

Stopping individual tiers

If the business application server and web application server for an environment are separated physically they can be stopped individually using the following set of tasks:

- Logon to the server containing the Web application server or Business application server using the product Administration account for the desired environment (for example, **cissys**).

- command:

Linux/Unix:

```
<SPLDIR>/<environment>/bin/splenviron.sh –e <environment>
```

Windows:

```
<SPLDIR>\<environment>\bin\splenviron.cmd –e <environment>
```

Where **<SPLDIR>** is the mount point defined for product and **<environment>** is the name of the environment to start.

- Start the web application server or business application server using the following command:

Linux/Unix:

```
spl.sh –w stop
```

Windows:

```
spl.cmd –w stop
```

Note: The command is the same but the utility will recognize by the host name what component is to be started.

Refer to the spl – Start/Stop Environment utility for more options.

Note: As an alternative, it is possible to stop the Web application server and Business application tiers using the console or utilities provided with the J2EE Web application server software.

The script will display the shutdown messages as dictated by the J2EE Web application server vendor.

- Shutdown the database using the utilities provided by the database vendor.

What to Look For in Shutdown Messages

As outlined in Common Application Logs the application logs all information to application logs during the startup, operation and shutdown of the application. These logs can be used to check that the shutdown of the product is successful. The logs contain the following sections for a shutdown (*class indicates message class used*):

- If the online batch daemon was enabled, it is shutdown (*classes = grid.node.OnlineGridNode, grid.node.DistributedGridNode, grid.space.SpaceManager, grid.space.TaskScheduler, grid.space.TaskScheduler* and *grid.space.ThreadPool*). The "**Thread pool shutting down**" message indicates a successful shutdown.
- The Web application server/business application server applications are asked to shutdown (*class = web.startup.SPLWebStartup*).
    - JMX connectors to the product are shutdown
    - The Application Context within the J2EE Web application server are shutdown. This may be delayed if COBOL is installed.

- If COBOL is used, then the COBOL Child (or Worker) JVMS are shutdown. The term used is "shunned". Each JVM is shunned individually.

  Note: A message "**`java.net.SocketException closing connection`**" may be displayed. This indicates that the socket has been closed.

- Database connections are closed (*class = hibernate.impl.SessionFactoryImpl*).
- Application shutdown is complete when the message "**`(web.startup.SPLWebStartup) Application Context shutdown successfully`**" is displayed.

Monitoring

This section outlines some basic monitoring regimes and methods for the product. It is highly recommended that you read the **Performance Troubleshooting Guides** available on MetaLink.

During monitoring you are typically looking for unusual activity and seeing if the current configuration of the product can handle the peaks and troughs of usage.

Unusual activity is activity that is not representative of the normal activity. For example, maybe during a marketing campaign the call center traffic doubles. This would be regarded “unusual activity”. At this point the current configuration may not be configured to handle the traffic so the problem needs to be identified and the configuration changed to cater for the new load.

Also during normal operations underlying problems may surface in the form of long running transactions, increases in error rates (in logs and timeouts) or “runaway transactions”. “Runaway transactions” are transactions that seem to be looping. These can be caused by data inconsistencies or bugs. Most of them are due to an unusual combination of data entries.

Some customers collect usage information to identity and analyze unusual activity. This is known as Site Profiling, Capacity Planning or Availability Planning. This is typically "Proactive" activity.

The product stores usage information within the database that can be extracted for this purpose. This section outlines the methods and techniques you can use to extract this information reactively and proactively.

Contents

Monitoring Regimes

Typically the art of monitoring is the collection and analysis of various pieces of information and then making changes to the configuration to address any issues or problems that occur.

With the various monitoring facilities available in the product a combination that is valid for the site becomes a monitoring regime for that site. Typically monitoring regimes pickup trends in the business or traffic volumes that require changes to the configuration. As part of the implementation of the product the monitoring regime for your site should be determined.

Typically the monitoring regimes that are chosen fall into a number of categories:

- **Reactive** - Monitoring for any exception after it happens and making changes to the configuration to prevent the exception from occurring again. This is the most common regime adopted by IT groups. The only problem with this approach is that you have to experience potentially threatening outages before stabilization happens.
- **Proactive** - Setting monitoring tolerances so that exception conditions are recognized before they happen and making configuration changes to prevent them from happening. This is also known as “Problem Anticipation” or “Problem Prevention”. This is the goal of most of the IT groups to ensure high availability.
- **Mixed** - This is a mixture of pro-active and re-active regime. This is not uncommon.

Monitoring Client Machines

The product's front end is the Microsoft Internet Explorer browser. Typically any Internet Explorer or operating system monitoring specified by Microsoft can be performed against the client to yield performance information.

While collecting this information can be performed using various tools, it is usually not applicable in all monitoring situations unless the client machine is below the specification outlined in the Installation Guide for the platform and version of the product you are using. The browser collection points specified here are typically the ones that are more applicable to the product than all of the available ones for the client.

Refer to the Microsoft documentation on how to fully monitor a client machine for performance information

Contents

Monitoring Client Machines

One of the areas that customers tend to monitor is the desktop client. Typically this involves using tools provided by Microsoft (and other vendors) to collect typical statistics, such as cpu, disk activity, memory usage and network usage. It is possible to monitor the client using the following tools:

- Microsoft tools (Performance Monitor) – The Performance Monitor (located in the "Administration Tools" menu from Windows) is a starting point for monitoring the client. Refer to Microsoft documentation on what aspects of a client machine to monitor.
- Network Monitor (**`netMon`** or other) – Windows Server includes a network capture facility that is handy to locate problems on a client machine. Alternatives are available such as Ethereal etc.
- Network Latency - Network tools like **`ping`** and **`traceroute`** measure latency by determining the time it takes a given network packet to travel from source to destination and back, the so-called round-trip time. Round-trip time is not the only way to specify latency, but it is the most common. Inconsistent ping times or long ping times can indicate network issues.
- Bandwidth Saturation levels - A number of tools exist for computer networkers to measure the bandwidth of network connections. On LANs, these tools include **`netperf`** and **`ttcp`**.
- Packet Loss - Packet loss is when data packets appear to be transmitted correctly at one end of a connection, but never arrive at the other. This might be because:
    - Network conditions are poor and the packet became damaged in transit.
    - The packet was deliberately dropped at a router because of congestion.
- Packet loss can be detected from the client PC using **`netstat`** and calculating the percentage of the "Segments Sent" that become "Segments Retransmitted".

> *Note:* **`ping`** and **`traceroute`** also include packet loss statistics.

- Failed Connection Attempts - When the client and/or server cannot accept a connection it generates a "Failed Connection Attempt" on either the client or the server (or both). A large number of "Failed Connection Attempts" can indicate networking or capacity issues on the client or server. The most common cause is that the accept queue on the network parameters (usually on the network cards) is full, and there are come requests waiting on the sync queue (usually on the network card).

Client Debug facility

Before a problem is to be registered with ORACLE support the transaction that caused the problem should be traced to help support solve the issue quickly. A debug facility is provided within the product to help capture this additional information.

Logging of debug information can be set at a global level or at a "local" level. The global debug setting is not recommended for a production system as it reduces overall performance and therefore is not covered in this document.

The "local" level enables you to navigate to the problem area and then to switch debugging on for that individual user to recreate the problem. You can then collate the debug information to be sent to support.

To use this facility you must specify an additional parameter at the end of the URL. For example:

```
http://<host>:<port>/<server>/cis.jsp?debug=true
```

After the debug control menu is displayed, you navigate to the screen where the problem is encountered and then enable "Global debug" by "toggling" the checkbox on. To turn off global debug toggle the check box off.

You should select "Trace All" for effective tracing. The other options are used by Developers only. The trace information is written to the **`spl*.log`** in the **`$SPLSYSTEMLOGS`** (**`%SPLSYSTEMLOGS%`** in Windows).

Note: The product has introduced **`spl_web.log`** and **`spl_service.log`** and they may or may not appear depending on the installation, therefore **`spl*.log`** is mentioned on the slide.

Debug allows specific information to be logged:

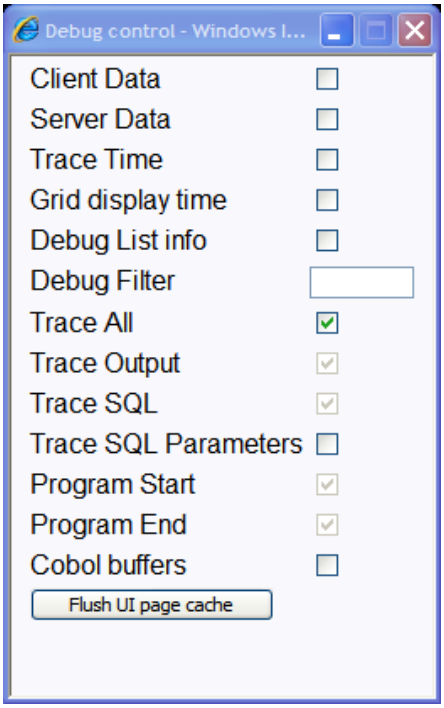
- **Client Data** – Data presented to the browser. This pops up an additional window displaying the object as it is built.
- **Server Data** – Data presented to the server. This pops up an additional window displaying the object as it is received by the server.
- **Trace time** – Include time tracing in the log.
- **COBOL buffers (if COBOL is used), Debug List Info, Debug Filter and Grid Display Time** – Used for development to display internal information and filter for specific information. It is recommended that these options should not be used unless performing development.
- **Trace All** – Enable all trace modes below except Trace SQL Parameters.
- **Trace Output** – Dump output from all calls
- **Trace SQL** – Dump SQL statements
- **Trace SQL Parameters** – Dump all result sets (*Warning: This is not recommended for production systems as it will result in performance degradation.*)
- **Program Start** – Write a record for ever module start
- **Program End** – Write a record for ever module end

Most tracing in non-development uses "Trace All" unless otherwise instructed by ORACLE Support. All debug information is written to the **`spl*.log`** files.

Steps to using the debug facility

To use the debug facility you follow the process:

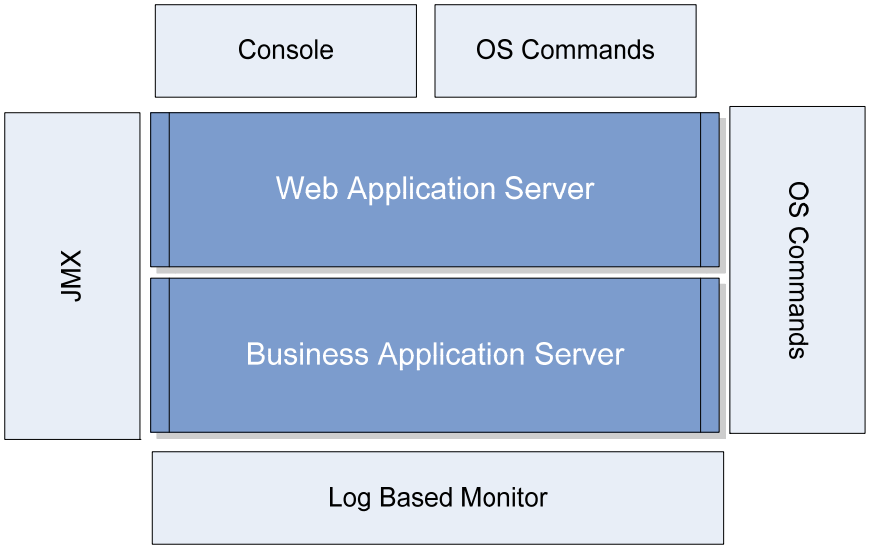
- Add **?debug=true** to your URL for the product. This will display the debug buttons on the browser screen.
- Navigate to the transaction that you wish to trace as a user would normally operate.
- Select "Global debug" so that debugging is enabled. This will display the "Debug Control" window where you should ensure that "Trace All" is selected. Other options should only be used if instructed by ORACLE Support.



- Run the transaction that you want to trace and to recreate the issue. While you work the trace information is written to the log files.
- Deselect "Global Debug" so that debugging is disabled. This will close the writing to the log. If you select "Show User Log" it will display the number of debug lines indicated next to the button. This will only show lines applicable to the Current User only.

Monitoring Web/business Application Server

There are a number of methods that are available for monitoring a Web Application from a J2EE Web application server:



- **Java Management Extensions (JMX)** – Most Web application servers expose JMX Management Beans (MBeans) to allow JMX browsers to view and use this information. Java 5 has a predefined set of MBeans that can be enabled automatically.
- **Web application server console** – All Web application server offer a web based console that provides both administration and basic monitoring functions. These are usually sufficient for spot real time checking of tolerances and basic monitoring. Some console use calls to JMX API's provided by the Web application server vendor and built into Java 5 (and above).
- **Command Based Utilities** – Apart from the console, most Web application server vendors offer a command line utility to extract performance information (or perform administration). Most console utilities call JMX MBeans and provide a command line interface into JMX that can be used natively.
- **Log-based monitoring** – Most Web application servers provide standardized logs that can be analyzed using consoles, log monitors or simple scripts.
- **Native OS utilities** – Most operating systems are becoming java aware and provide OS and Java monitoring from OS monitoring facilities.

Contents

Key Statistics

Note: Please refer to the Performance Troubleshooting Guides for details of key statistics to track.

When monitoring the J2EE Web application server there are a number of statistics that are important to track:

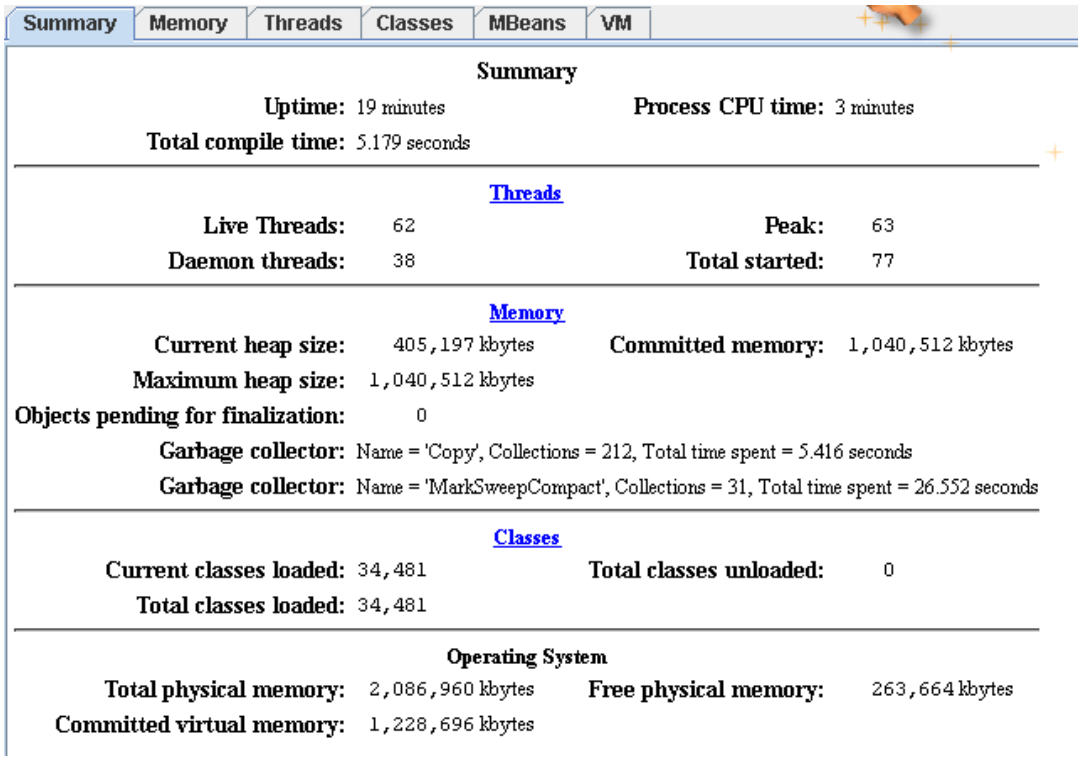
- Thread Pool – Size and usage of the thread pool connections between the users and the Web application server. Useful for determining if you have enough pooled connections.

- J2EE 5 Server Statistics (JSR77) (http://java.sun.com/javaee/5/docs/api/javax/management/j2ee/statistics/package-summary.html) – A collection of statistics that provide basic Java 5 statistics including:
    - Java heap size
    - CPU time
    - Java thread statistics
    - Uptime calculation
    - Servlet statistics by page (each Web Application collects different information)
    - Garbage collection statistics
    - Class loading statistics

Using Jconsole

As with any J2EE Web based application it is possible to use **`jconsole`** (or other JMX browser) to view JMX MBeans and statistics from the Java process (this includes any background process as well).

- Ensure your class files are in the path (for example **`c3p0-0.9.1.2.jar`** file – not Hibernate as c3p0 collects pools statistics).
- Enable **`com.sun,management.jmxremote`** with appropriate settings for your JVM on the java command lines. This can be done at the script level or within your Web application server. This will open a JMX port. Refer to http://java.sun.com/j2se/1.5.0/docs/guide/management/agent.html for more information.
- Start **`jconsole`** with the jar files (e.g. **`c3p0-0.9.1.2.jar`**) in the class path



Configuration

Contents

Configuration Files

There are a number of configuration files that are available with each environment.

Contents

cistab - Global Configuration Files

The **cistab** file is a key configuration file for both the Web application server and the database application server. It is built during the installation process and is used by the product administration utilities to ensure that any output or log files generated by the product are stored in the correct location. It holds the mount points (e.g. directories) used during the installation of the product to hold the product and its log files.

Location of **cistab** file:

Linux/Unix:

```
/etc/cistab
```

Windows:

```
c:\spl\etc\cistab
```

Warning! Do not alter the **cistab** file unless instructed to do so by ORACLE support personnel unless otherwise directed.

An example **cistab** file is outlined below:

```
DEV::/spl/DEV:/spl/sploutput/DEV::N
DEMO::/spl/DEMO:/spl/sploutput/DEMO::N
TEST::/spl/TEST:/spl/sploutput/TEST::N
TEST2::d:\spl\TEST2:e:\sploutput\TEST2::N
```

The format of the file is described below:

Position	Usage
1	Environment Name – specified at installation time. It is in UPPER case.
2	*Reserved for future use.*
3	Mount point for the product software and configuration files (the **SPLEBASE** environment variable definition).
4	Mount point for the product output files the **SPLOUTPUT** environment variable definition).
5	*Reserved for future use.*
6	This flag may be used in custom start up scripts to indicate whether to start the environment at system boot time. Valid values are **Y** or **N**. *This is the only setting that should be altered after installation*.

ENVIRON.INI - Environment Configuration File

The **ENVIRON.INI** file is used by the Web application server and the business application server to define the environment and provide the basis for starting and stopping the environment. The file is created during the installation process and is used to generate other files.

This file is maintained using the configureEnv – Setup Environment settings script provided in the installation.

Warning! Do not alter the **ENVIRON.INI** manually. Always use configureEnv – Setup Environment settings because additional configuration files depend on the settings in this file. If the configurations mismatch, improper operation of the product may occur.

Location of **cistab** file:

Linux/Unix:

```
$SPLEBASE/etc/ENVIRON.INI
```

Windows:

```
%SPLEBASE%\etc\ENVIRON.INI
```

The file contents are in text format and are of the form:

```
<parameter>=<value>
```

Where:

<parameter> is the configuration parameter.

<value> is the value of the parameter.

For example:

```
…
appViewer=appViewer
DBCONNECTION=jdbc:oracle:thin:@myserver:1521:train
```

```
DBDRIVER=oracle.jdbc.driver.OracleDriver
DBENCPASS=21424852545244510F0F21
DBPASS=ENC(Qssb0Lawc1/b+jPu7lM2eS4wbFbrHIVobujyS8EQ6iM=)
DBNAME=TRAIN
DBUSER=spluser
…
```

The settings contained in the **ENVIRON.INI** file are outlined in the table below:

Legend:

Tier	Blank = all, WAS = Web application server, BAS = business application server, XAI = Web Services Adapter, DB = Database.
Platform	Blank = all, WLS = ORACLE WebLogic, OAS = ORACLE Application Server, OC4J = ORACLE OC4J, WAS = IBM WebSphere, ORA = ORACLE Database, MSSQL = Microsoft SQL Server, DB2 = IBM DB2

Parameter	Description	Tier	Platform
AppViewer	Name of appViewer WAR file	WAS	
BATCHDAEMON	Inbuilt Batch Daemon enabled	BAS	
BATCHENABLED	Inbuilt Batch Server enabled	BAS	
BATCHTHREADS	Maximum # of threads for Inbuilt Batch Server	BAS	
BSN_JVMCOUNT	Number of JVM Processes	BAS	
BSN_OASAPP	OAS Application Name	BAS	OAS
BSN_OASREQPORT	OAS Request Port	BAS	OAS
BSN_OASRMIPORT	OC4J Instance RMI Port	BAS	OAS
BSN_OASSYSPASS	OAS System Password	BAS	OAS
BSN_OASSYSUSER	OAS System Userid	BAS	OAS
BSN_OC4J_instance	OAS OC4J Instance	BAS	OAS
BSN_OC4JORMPORT	OC4J Standalone ORMI Port	BAS	OC4J
BSN_OC4JSYSPASS	OC4J Standalone Password	BAS	OC4J
BSN_OC4JSYSUSER	OC4J Standalone Userid	BAS	OAS
BSN_RMIPORT	JVM Child process Port Number	BAS	
BSN_SVRNAME	WebSphere Server Name	BAS	WAS
BSN_WASAPP	WebSphere Application Name	BAS	WAS
BSN_WASBOOTSTRAPPORT	Bootstrap Port	BAS	WAS
BSN_WASPASS	WebSphere deployment password	BAS	WAS

Parameter	Description	Tier	Platform
BSN_WASUSER	WebSphere deployment userid	BAS	WAS
BSN_WLHOST	Business App Server Host	BAS	
BSN_WLSYSPASS	WebLogic System Password	BAS	WLS
BSN_WLSYSUSER	WebLogic System Userid	BAS	WLS
CMPDB	Database Type (`ORA`, `MSSQL` or `DB2`)	DB	
COLLATE	SQL Server Collation Name	DB	MSSQL
DB2CODEPAGE	DB2 Code Page	DB	DB2
DB2COLL	DB2 Collection Name	DB	DB2
DB2LOCL	DB2 Location Name	DB	DB2
DB2MPLUse	Whether DB2 is used by the MPL	DB	DB2
DB2SERVER	DB2 Host Name	DB	DB2
DBCONNECTION	JDBC Connection string (generated)	DB	
DBDRIVER	Hibernate DB driver	DB	
DBENCPASS	Encrypted `DBPASS`	DB	
DBPASS	Database password for Database User. If the value is prefixed by "ENC" then the password is encrypted.	DB	
DBPORT	Port Number for Database	DB	
DBUSER	Database User used by product	DB	
DESC	Environment Description		
DIALECT	Hibernate Dialect	DB	
DIRSEP	Directory separator		
DOC1BILLSCRIPT	DOC1 Bill Script location (*if DOC1 installed*)	BAS	
DOC1SCRIPT	DOC1 Letter Script location (*if DOC1 installed*)	BAS	
ENCODING	Whether encryption is enabled		
help	Name of online help WAR file	WAS	
highValue	Language specific highvalues	BAS	
JavaEncoding	Java Language Encoding	BAS	
Jvmcommand	Generated java command for Child JVM (if COBOL used)	BAS	
JVMMEMORYARG	Child JVM Memory Allocation (if COBOL used)	BAS	
modules	Names of Modules installed		

Parameter	Description	Tier	Platform
MPLADMINPORT	MPL Administration Port	XAI	
MPLSTART	MPL Automatic Start (Y/N)	XAI	OAS
NLS_LANG	NLS Language setting	DB	ORA
ORACLE_SID	ORACLE Database Id	DB	ORA
OWNERUSER	Owner of Database Schema	DB	
REL_CBL_THREAD_MEM	Release COBOL memory (if COBOL used)	BAS	
RJVM	Whether Child JVM is considered remote (if COBOL used)	BAS	
SPLADMIN	OS Administration userid		
SPLADMINGROUP	OS Administration group		
SPLApp	Name of product WAR file	WAS	
SPLDB	Database Type (`ORA`, `MSSQL` or `DB2`)	DB	
SPLENCPASS	Encrypted `WEB_SPLPASS`	WAS	
SPLENVIRON	Environment Identifier		
SPLSERVICEAPP	Name of Business App Server Application	BAS	
SPLWAS	Web application server installed (`WAS`, `WLS`, `OAS` or `OC4J`)	WAS	
SPLWEBAPP	Name of Web App Server Application	WAS	
SQLDB	SQL Server Database Name	DB	MSSQL
WEB_isAppViewer	Deploy AppViewer to Web Server	WAS	
WEB_isDevelopment	Is environment used for Development	WAS	
WEB_isExpanded	Exploded directory or WAR/EAR files	WAS	
WEB_maxAge	Length of time IE cache entry for text	WAS	
WEB_maxAgeI	Length of time IE cache entry for images	WAS	
WEB_OASAPP	OAS Application Name	WAS	OAS
WEB_OASREQPORT	OAS Request Port	WAS	OAS
WEB_OASSYSPASS	OAS System Password	WAS	OAS
WEB_OASYSUSER	OAS System Userid	WAS	OAS
WEB_OC4J_instance	OAS OC4J Instance	WAS	OAS
WEB_OC4JSYSPASS	OC4J System Password	WAS	OC4J
WEB_OC4JSYSUSER	OC4J System Userid	WAS	OC4J

Parameter	Description	Tier	Platform
WEB_preloadall	Preload all pages on startup	WAS	
WEB_SPLPASS	Application Administration Password	WAS	
WEB_SPLUSER	Application Administration Userid	WAS	
WEB_SVRNAME	WebSphere Server Name	WAS	WAS
WEB_TCATSHUTPORT	Tomcat Shutdown Port	WAS	TCAT
WEB_WASAPP	WebSphere Application Name	WAS	WAS
WEB_WASPASS	WebSphere deployment password	WAS	WAS
WEB_WASUSER	WebSphere deployment userid	WAS	WAS
WEB_WLAUTHMETHOD	Authentication Method (`BASIC` or `FORM`)	WAS	
WEB_WLHOST	Web Server Host	WAS	
WEB_wlpageCheckSeconds	Interval for recompilation of JSP	WAS	WLS
WEB_WLPORT	Web Server HTTP Port	WAS	
WEB_WLSSLPORT	WebLogic SSL HTTP Port	WAS	WLS
WEB_WLSYSPASS	WebLogic System Password	WAS	WLS
WEB_WLSYSUSER	WebLogic System Userid	WAS	WLS
WLSSPL	Bundled WebLogic or External	WAS	WLS
XAIApp	Name of Web Services Adapter WAR file	WAS	
XAISTARTWAITTIME	Delay before XAI Starts (seconds)	XAI	

Note: All passwords are encrypted using the AES-128. NEW

Extracting Information from ENVIRON.INI for Scripts

It is possible to write your own calls to the **ENVIRON.INI** using the same utilities used by the product to get values of configuration parameters for your own utilities. Do not hardcode values that can be obtained from **ENVIRON.INI**.

To obtain values of parameters use the command line:

Linux/Unix:

```
perl $SPLEBASE/bin/getconfvalue.plx –k <parameter>
```

Windows:

```
perl %SPLEBASE%\bin\getconfvalue.plx –k <parameter>
```

where

<parameter> is the parameter from **ENVIRON.INI** you wish to get the value of.

For example:

ENVIRON.INI:

```
…
DBNAME=TRAIN
…

$ export DB=`perl $SPLEBASE/bin/getconfvalue.plx –k DBNAME`
$ echo $DB
TRAIN
```

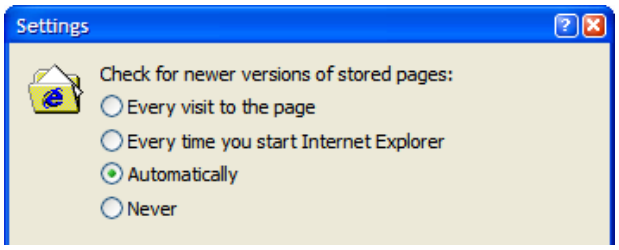
Note: If the value is NOT set or the key is invalid the value of the call is null or blank.

Web Browser Configuration

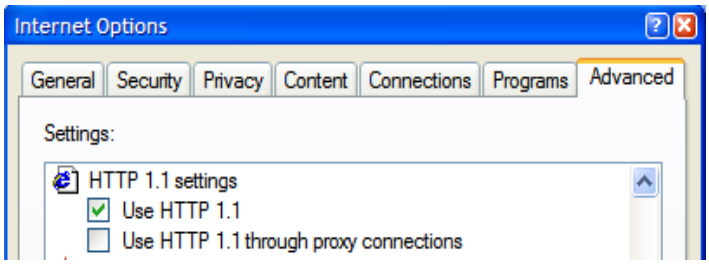
This section outlines additional settings that are not covered in the Installation Guide for your platform.

The product is browser based (browsers, versions and platforms are documented in the Installation Guide for your platform. Additionally the following settings are applicable to the browser:

- Cache settings need to be "Every visit to the page" or "Automatically". For non-production it is recommended to be set to "Every visit to the page" or "Automatically". For production it is recommended to be set to "Automatically" to fully exploit performance caching. See http://www.microsoft.com/windows/ie/ie6/using/howto/customizing/clearcache.mspx for more details on caching.

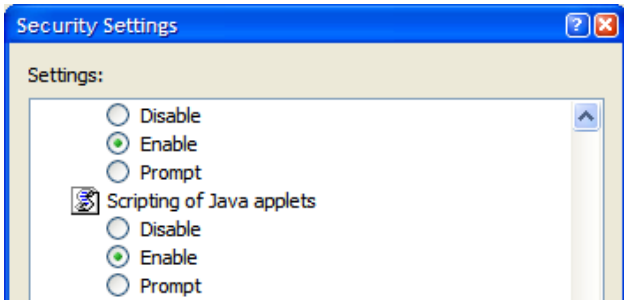


- Per-session cookies (not stored) need to be enabled. The Browser user interface uses a session cookie (in memory) for holding server credentials to be passed to the server for each transaction.
- The product requires support for the HTTP 1.1 protocol to support compression and client cache management.



Note: If a network proxy is used then "Use HTTP 1.1 through proxy connections" may need to be selected as well.

- The product uses Java scripting for user interactivity therefore "Scripting of Java Applets" must be enabled.



- The product uses popup windows for searches, therefore popup blockers must be configured to allow popups from Web application server hosts.
- Set your Internet Explorer cache size to a reasonable size to hold the cached pages as needed. See http://www.microsoft.com/windows/ie/ie6/using/howto/customizing/clearcache.mspx for more details.

Web application server Configuration

Contents

Caveat

The product supports a number of J2EE Web application servers. Each J2EE Web application server is configured differently and has additional options (clustering, logging etc) that can be used. This document is neutral to the differences of each J2EE Web application server. Refer to the documentation provided with the J2EE Web application servers for the location of specific configuration settings discussed in this section as well as advanced settings supported.

Web Server Concepts

Each Web application server has a number of levels and each uses different terminology. The following "neutral" terminology will be used:

- The software exists on a physical machine.
- An installation of the Web Application Software is called an instance. Typically one instance of the software exists on a machine but you can have more than one installed.
- Within an instance you can define a server. This is also called a Java "container" which will house one or more J2EE applications. You will have at least one server per environment. A server uses one Java Virtual Machine (JVM).
- Within a server is the J2EE application. It can be a single J2EE application or multiples depending on the Web application server supported.

The Web application server you use may have different terminology for these same concepts. For the remainder of this section we will use the above terminology.

Web application server Web Applications

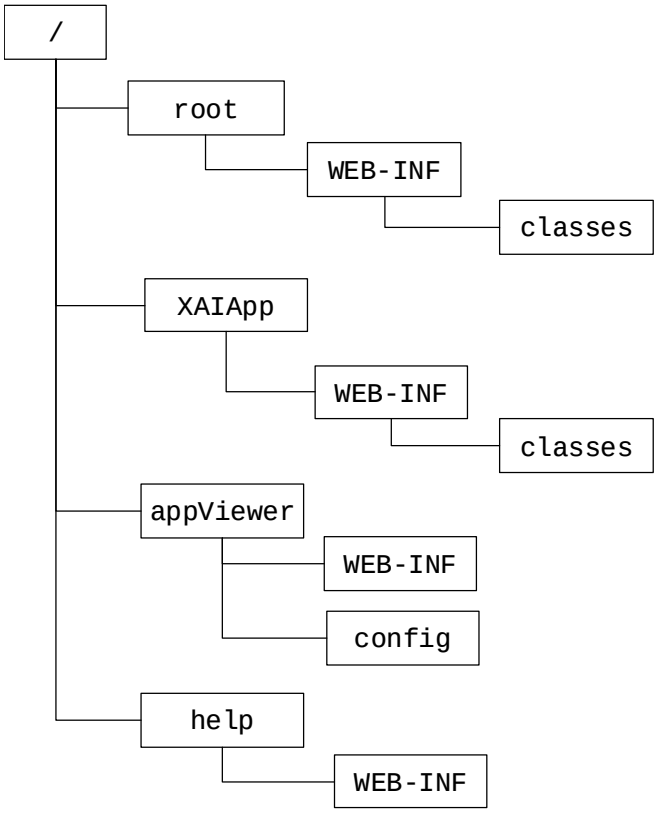
The product is deployed as a set of Web applications within the Web applications server:

- **root** – This is the product itself is installed.
- **XAIApp** – This is the Web services adapter component.
- **appViewer** – An Application Viewer which contains a data dictionary and source viewer.
- **help** – Online Help.

Each of these J2EE Web Applications has its own configuration files and are combined together when the product is "built" into a WAR/EAR file by the genupdatewar – Generate J2EE WAR and EAR files utility.

Web application server Configuration Files

Within each J2EE Web Application within the J2EE Web application server has it's own configuration files. These files are typically "embedded" within the EAR/WAR files deployed with the product following the J2EE specification. In terms of configuration, the product structure within the EAR/WAR file looks like the following:



Location	Contents	Configuration Files
root/WEB-INF	J2EE Application Descriptor for online application	**Contents** web.xml – J2EE Application Descriptor log4j.properties – Logging Configuration spl.properties – Product configuration settings weblogic.xml – WebLogic Extensions web.xml – J2EE Application Descriptor
root/WEB-INF/classes	Application Configuration files for online application	log4j.properties – Logging Configuration spl.properties – Product configuration settings weblogic.xml – WebLogic Extensions
XAIApp/WEB-INF	J2EE Application Descriptor for Web Services Adapter	**Contents** web.xml – J2EE Application Descriptor log4j.properties – Logging Configuration spl.properties – Product configuration settings weblogic.xml – WebLogic Extensions web.xml – J2EE Application Descriptor
XAIApp/WEB-INF/classes	Application Configuration files for Web Services Adapter	log4j.properties – Logging Configuration spl.properties – Product configuration settings
appViewer/WEB-INF	J2EE Application Descriptor for Application Viewer	**Contents** web.xml – J2EE Application Descriptor log4j.properties – Logging Configuration spl.properties – Product configuration settings weblogic.xml – WebLogic Extensions web.xml – J2EE Application

Location	Contents	Configuration Files
		Descriptor
appViewer/config	Application Viewer configuration	log4j.properties – Logging Configuration spl.properties – Product configuration settings
help/WEB-INF	J2EE Application Descriptor for online help	**Contents** web.xml – J2EE Application Descriptor log4j.properties – Logging Configuration spl.properties – Product configuration settings weblogic.xml – WebLogic Extensions web.xml – J2EE Application Descriptor

web.xml – J2EE Application Descriptor

The Web deployment descriptor editor lets you specify deployment information for modules created in the Web development environment. The information appears in the **web.xml** file. The **web.xml** file for a Web project provides information necessary for deploying a Web application module. It is used in building a WAR/EAR file from a project.

The Web Application is controlled by a configuration file that holds behavioural information for the applications. Refer to **http://jcp.org/en/jsr/detail?id=109** for more details of the format. For example:

```
…
<env-entry>
      <description>Value of HTTP 1.1 max-age header parameter for
JSPs</description>
      <env-entry-name>maxAge</env-entry-name>
      <env-entry-value>28800</env-entry-value>
      <env-entry-type>java.lang.Integer</env-entry-type>
   </env-entry>
   <env-entry>
      <description>How long to cache drop down values in seconds</description>
      <env-entry-name>fieldValuesAge</env-entry-name>
      <env-entry-value>3600</env-entry-value>
      <env-entry-type>java.lang.Integer</env-entry-type>
   </env-entry>
   <env-entry>
      <description>Is this a development environment</description>
      <env-entry-name>isDevelopment</env-entry-name>
```

```
      <env-entry-value>false</env-entry-value>
      <env-entry-type>java.lang.Boolean</env-entry-type>
   </env-entry>
   <env-entry>
      <description>Preload ALL Pages</description>
      <env-entry-name>preloadAllPages</env-entry-name>
      <env-entry-value>false</env-entry-value>
      <env-entry-type>java.lang.Boolean</env-entry-type>
   </env-entry>
   <env-entry>
      <description>Disable preloading of Pages</description>
      <env-entry-name>disablePreload</env-entry-name>
      <env-entry-value>false</env-entry-value>
      <env-entry-type>java.lang.Boolean</env-entry-type>
   </env-entry>
…
```

The following settings apply to Web Application Descriptor for Web application server:

web.xml Parameter	Context	Source
disableCompression	Enables or disables compression between browser and web application server (**true** or **false**). Default is **false**.	Derived from **WEB_isDevelopment** parameter from ENVIRON.INI - Environment Configuration File
maxAge (Images)	How long images are stored in the IE cache in seconds?	Derived from **WEB_maxAgeI** parameter from ENVIRON.INI - Environment Configuration File
auth-method	Security setup for product.	Derived from **WEB_WLAUTHMETHOD** parameter from ENVIRON.INI - Environment Configuration File
maxAge	How long texts are stored in the IE cache in seconds?	Derived from **WEB_maxAge** parameter from ENVIRON.INI - Environment Configuration File
fieldValuesAge	How long the static cache is kept on the Web application server in seconds?	Defaulted to **3600** in template
preloadAllPages	Whether server builds screens from main menu only (**false**) or all menus (**true**)? Defaults to **false**.	Derived from **WEB_preloadall** parameter from ENVIRON.INI - Environment Configuration File
disablePreload	Enables or disables preload altogether (**true** or **false**). Defaults to **false**.	Defaulted in template

web.xml Parameter	Context	Source
XAIServerURL	Web Services URL	Derived from Web application server parameters (**WEB_WLHOST**, **WEB_WLPORT**) in ENVIRON.INI - Environment Configuration File
HTTPBasicAuthUser	Default User IDused by Web Services Adapter	Derived from **WEB_SPLUSER** parameter from ENVIRON.INI - Environment Configuration File
HTTPBasicAuthPasswordEnc	Encrypted password for **HTTPBasicAuthUser**	Derived from **SPLENCPASS** parameter from ENVIRON.INI - Environment Configuration File
disableUIPageCompression	Enables or disables compression between browser and web application server (**true** or **false**). Default is **false**.	Derived from **WEB_isDevelopment** parameter from ENVIRON.INI - Environment Configuration File
ClassicServerURL	URL used for backward compatibility (**XAIApp** only)	Derived from Web application server parameters (**WEB_WLHOST**, **WEB_WLPORT**) in ENVIRON.INI - Environment Configuration File

Note: It is *highly* recommended that you do *not* change this configuration file by extracting the configuration file from the EAR/WAR file using Java utilities, making the change manually and rebuilding the EAR/WAR file. Use genupdatewar – Generate J2EE WAR and EAR files to build the EAR/WAR file as documented in Web application server Configuration Process

log4j.properties – Logging Configuration

Note: This log file should *not* be altered unless specified. The generated configuration file has *all* the recommended settings for *all* sites.

The product uses the log4j Java classes to centralize all log formats into a standard format. The details of the configuration settings and log4j itself are available at http://logging.apache.org/log4j/ or http://en.wikipedia.org/wiki/Log4j.

spl.properties – Product configuration settings

The product Web Application has a specific number of settings outside of the J2EE specification to control the internals of the product. This file exists as similar files exist for *all* modes of operation of the product (for example, Batch can be run outside the J2EE Web application server). Because of this a common configuration standard was adopted:

For the Web application server the **spl.properties** uses the following settings.

Parameter	Context	Source
spl.tools.loaded.applications	List of applications installed. Values are typically base,ccb,cm	Generated by installation script.
spl.runtime.options.isDevelopmentMode	Whether the environment is used for development (true or false).	Derived from WEB_isDevelopment parameter from ENVIRON.INI - Environment Configuration File
spl.runtime.socket.file.dir	Working directory for workable sockets	Defaulted from template
spl.runtime.environ.init.dir	Location of the base configuration files.	Defaulted from template
spl.runtime.service.extraInstallationServices	Name of Application service used for installation defaults. Default: CILTINCP	Defaulted by template.

weblogic.xml – WebLogic Extensions

Note: This configuration file only applies to ORACLE WebLogic implementations.

For backward compatibility with ORACLE WebLogic environments, an additional ORACLE WebLogic configuration file **weblogic.xml** is generated and used to influence the ORACLE WebLogic Server to exhibit additional behavior (targeted for development primarily).

Parameter	Context	Source
url-rewriting-enabled	Provides methods for configuring a J2EE web application that is deployed on an ORACLE WebLogic Server instance. ORACLE WebLogic Server instantiates this interface only when you deploy a web application. This interface can configure web applications that are deployed as a WAR file or an exploded directory.	Derived from WEB_isExpanded parameter from ENVIRON.INI - Environment Configuration File

Parameter	Context	Source
page-check-seconds	Determines the interval at which a server checks to see if JSP files in a Web application have changed and need recompiling. Used for development	Defaults from template
servlet-reload-check-secs	Defines whether an ORACLE WebLogic Server will check to see if a servlet has been modified, and if it has been modified, reloads it. The -1 value tells the server never to check the servlets, 0 tells the server to always check the servlets, and the default is to check each 1 second. A value specified in the console will always take precedence over a manually specified value.	Defaults from template
resource-path	A path which, if included in the URL of a request, signals ORACLE WebLogic Server to use the Java character set specified by java-charset-name.	Defaults from template
java-charset-name	Specifies the Java character set to use.	Derived from NLS_LANG parameter from ENVIRON.INI - Environment Configuration File
context-root	The context-root element defines the context root of this stand-alone Web application. If the Web application is part of an EAR, not stand-alone, specify the context root in the EAR's web.xml file. A context-root setting in web.xml takes precedence over context-root setting in weblogic.xml.	Defaults from template

Note: This configuration file is not usually altered by an implementation as it applies to development (SDK) platforms only. It is documented for completeness here.

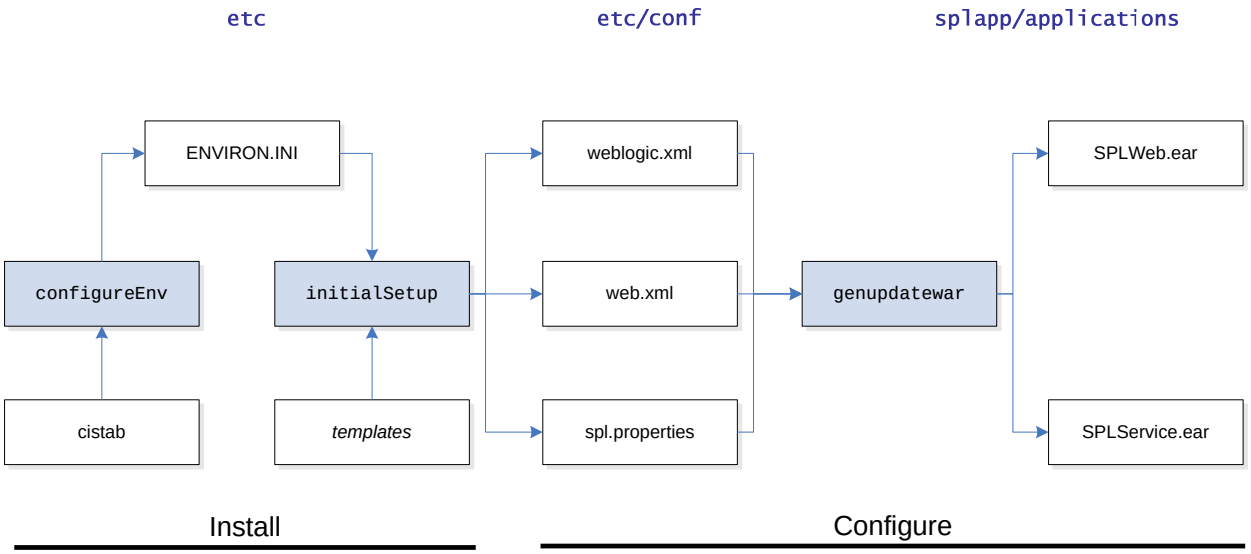
Example:

```
<weblogic-web-app xmlns="http://www.bea.com/ns/weblogic/90">
   <session-descriptor>
      <url-rewriting-enabled>false</url-rewriting-enabled>
   </session-descriptor>
   <jsp-descriptor>
      <page-check-seconds>43200</page-check-seconds>
   </jsp-descriptor>
  <container-descriptor>
    <servlet-reload-check-secs>-1</servlet-reload-check-secs>
```

```
    <prefer-web-inf-classes>true</prefer-web-inf-classes>
  </container-descriptor>
  <charset-params>
    <input-charset>
      <resource-path>/*</resource-path>
      <java-charset-name>UTF-8</java-charset-name>
    </input-charset>
  </charset-params>
  <context-root>/</context-root>
</weblogic-web-app>
```

Web application server Configuration Process

To configure the Web application server during the installation process and post-installation then the following process should be used:



- The configureEnv – Setup Environment settings utility is used during installation time and can be used post implementation to set parameters in the ENVIRON.INI - Environment Configuration File. If any parameters are derived or set from the ENVIRON.INI - Environment Configuration File (see "Source" column in the relevant section) then the configureEnv – Setup Environment settings should be used to maintain them.

  Note: The configureEnv – Setup Environment settings utility should be used to make *any* changes to the ENVIRON.INI - Environment Configuration File. Manual changes to this configuration file are not recommended.

- After the ENVIRON.INI - Environment Configuration File has been set or altered, the settings must be reflected in the relevant configuration files used by the Web application server by running the initialSetup – Reset configuration to template utility:
  - log4j.properties – Logging Configuration
  - spl.properties – Product configuration settings
  - weblogic.xml – WebLogic Extensions

- The utility uses the templates from the etc directory to create substituted copies of these files in a standard location:

Configuration File	Location of Configuration File
Online Application (root)	
web.xml	Linux/Unix: **$SPLEBASE/etc/conf/WEB-INF/web.xml** Windows: **%SPLEBASE%\etc\conf\WEB-INF\web.xml**
spl.properties	Linux/Unix: **$SPLEBASE/etc/conf/root/WEB-INF/classes/spl.properties** Windows: **%SPLEBASE%\etc\conf\root\WEB-INF\classes\spl.properties**
log4j.proprties	Linux/Unix: **$SPLEBASE/etc/conf/root/WEB-INF/classes/log4j.properties** Windows: **%SPLEBASE%\etc\conf\root\WEB-INF\classes\log4j.properties**
weblogic.xml	Linux/Unix: **$SPLEBASE/etc/weblogic.xml** Windows: **%SPLEBASE%\etc\weblogic.xml**
Web Services Adapter (XAIApp)	
web.xml	Linux/Unix: **$SPLEBASE/etc/conf/WEB-INF/web.xml.XAIApp** Windows: **%SPLEBASE%\etc\conf\WEB-INF\web.xml.XAIApp**
spl.properties	Linux/Unix: **$SPLEBASE/etc/conf/XAIApp/WEB-INF/classes/spl.properties**

Configuration File	Location of Configuration File
	Windows: **%SPLEBASE%\etc\conf\XAIApp\WEB-INF\classes\spl.properties**
log4j.proprties	Linux/Unix: **$SPLEBASE/etc/conf/XAIApp/WEB-INF/classes/log4j.properties** Windows: **%SPLEBASE%\etc\conf\XAIApp\WEB-INF\classes\log4j.properties**
Application Viewer (appViewer)	
web.xml	Linux/Unix: **$SPLEBASE/etc/conf/WEB-INF/web.xml.appViewer** Windows: **%SPLEBASE%\etc\conf\WEB-INF\web.xml.appViewer**
Help Application (help)	
web.xml	Linux/Unix: **$SPLEBASE/etc/conf/WEB-INF/web.xml.help** Windows: **%SPLEBASE%\etc\conf\WEB-INF\web.xml.help**

- At this point you may perform manual changes to the above files to parameters not implemented in the ENVIRON.INI - Environment Configuration File.

> *Note:* Any manual changes are overwritten after running the initialSetup – Reset configuration to template utility. Backups should be made of any changes and then manually reapplied to reinstate all manual changes.

- To reflect configuration changes into the product Web Applications the genupdatewar – Generate J2EE WAR and EAR files utility must be executed. This will build the necessary EAR/WAR files to be deployed into the J2EE Web application server.

Depending on the architecture, the genupdatewar – Generate J2EE WAR and EAR files utility will generate one or more EAR files. Refer to Business Application Server Configuration for a description of the EAR files.

At this point the product Web Applications are ready for deployment into the J2EE Web application server.

Quick Reference Guide for Web application server Configuration

To make configuration changes to the Web application server component of the product uses the following Quick Reference Guide to identify which process should be used:

- If the change is to any setting contained in the ENVIRON.INI - Environment Configuration File for the Web application server then you must run the following utilities in the order indicated:
    1. Execute the configureEnv – Setup Environment settings utility to reflect the parameter change in the ENVIRON.INI - Environment Configuration File.
    2. Execute the initialSetup – Reset configuration to template utility to rebuild the configuration files using the ENVIRON.INI - Environment Configuration File and provided template file. This will reset the configuration to the contents of the template files.
    3. (*Optional*) Make any manual changes to the configuration files potentially lost in Step 2.
    4. Execute the genupdatewar – Generate J2EE WAR and EAR files utility to implement the configuration files in the product Web Application files.
- If the change is to any setting not contained in the ENVIRON.INI - Environment Configuration File for the Web application server but is in the configuration files for the Web application server then you must run the following utilities in the order indicated:
    1. (*Optional*) Make any manual changes to the relevant configuration files.
    2. Execute the genupdatewar – Generate J2EE WAR and EAR files utility to implement the configuration files in the product Web Application files.

Web Application Server Deployment Process

After the configuration of the Web Application is complete (as outlined in Web application server Configuration Process) the final step to implement the product technically is to deploy the product within the J2EE Web application server.

There are three methods of deploying the product within the J2EE Web application server:

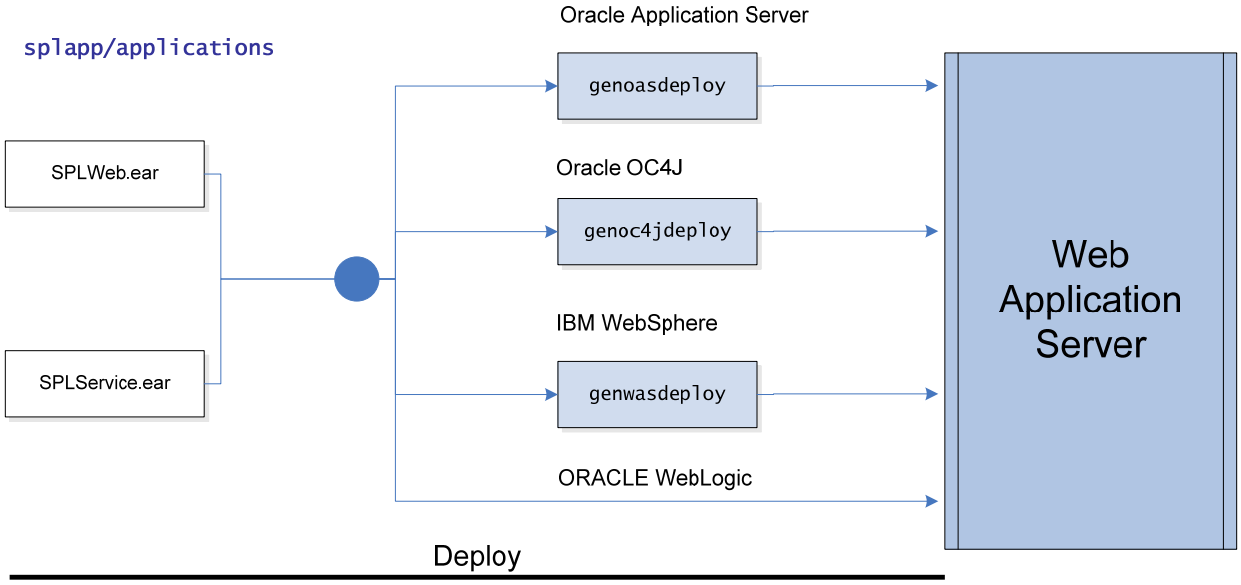
1. Use the deployment utilities provided on the console of the J2EE Web application server. The EAR/WAR files that are available under **`$SPLEBASE/splapp/applications`** (or **`%SPLEBASE%\splapp\applications`** for Windows) can be manually deployed using the console. Refer to the Installation Guide for specific platform instructions and the administration guide for the J2EE Web application server.

    Note: This is the only method that can be used if virtual Web application servers are used with the product.

2. Use the deployment utilities provided on the command line of the J2EE Web application server. The EAR/WAR files that are available under **`$SPLEBASE/splapp/applications`** (or **`%SPLEBASE%\splapp\applications`** for Windows) can be manually deployed using the command line utilities supplied with your J2EE Web application server. Refer to the Installation Guide for specific platform instructions and the administration guide for the J2EE Web application server.
3. A number of specific utilities for J2EE Web applications are provided with the product to deploy the Web Application to the J2EE Web application server. These call the same utilities provided in Option 2 but are provided with the product.

This section will outline Option 3 only.

A number of utilities are provided in the bin directory of the product to deploy the product to the J2EE Web application server. These utilities are outlined below:



- For the ORACLE Application Server platform, use the genoasdeploy – Deploy on ORACLE Application Server utility.
- For the ORACLE OC4J platform, use the genoc4jdeploy – Deploy on OC4J utility.
- For the IBM WebSphere platform, use the genwasdeploy – Deploy on WebSphere utility.
- For ORACLE WebLogic, no additional deployment is necessary as the product automatically detects WebLogic and allows WebLogic to read the EAR/WAR files directly.

These utilities will attempt to deploy the Web Applications within the J2EE Web application server as follows:

J2EE Web application server	Deployment details
ORACLE WebLogic	Deployed to root application by default using **WEB_SYSUSER** and **WEB_SYSPASS** from ENVIRON.INI - Environment Configuration File as administration credentials.
ORACLE Application Server	Deployed to **WEB_OASAPP** application by default using **WEB_OASYSUSER** and **WEB_OASYSPASS** from ENVIRON.INI - Environment Configuration File as administration credentials.
ORACLE OC4J	Deployed to **WEB_OC4J_instance** application by default using **WEB_OC4JSYSUSER** and **WEB_OC4JSYSPASS** from ENVIRON.INI - Environment Configuration File as administration credentials.
IBM WebSphere	Deployed to **WEB_WASAPP** Application on **WEB_SVRNAME** server by default using **WEB_WASUSER** and **WEB_WASPASS** from

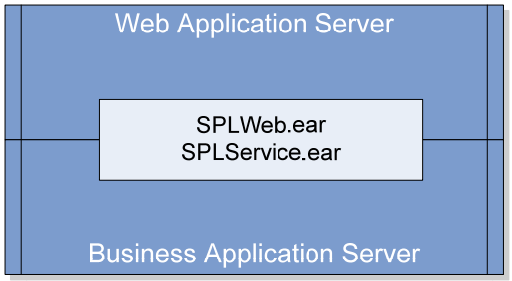
J2EE Web application server	Deployment details
	ENVIRON.INI - Environment Configuration File as administration credentials.

Business Application Server Configuration

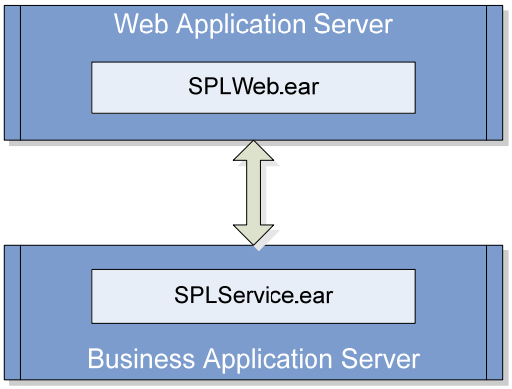
In V2.2 the business application server logic can be separated from the Web application server component. Essentially the product has been split into TWO distinct EAR files:

- **SPLWeb.EAR** – This contains the Web application server component for the product.
- **SPLService.EAR** - This contains the business application server component for the product.

There are two modes of installation:



Local install (a.k.a. expanded install)



Decoupled install

- **"Local" Installation** (also applicable to "expanded" installations for Development environments) - The Web application server and Business Application are on the same instance of the J2EE Web application server. This is the default behavior of the product for backward compatibility. If this is the mode installed then for configuration the process is a combination of the Web application server and business application server configuration and deployment process.

  Note: Local installations are only supported on development platforms and ORACLE WebLogic installations only.

- **Decoupled Installation** – The business application server is on a separate instance of the J2EE Web application server. This may be the same machine or different machines. In this case the Web application server and business application server are managed and configured separately. To perform a decoupled installation the following must be performed:

  1. The product is installed on the machines housing the Web application server and business application server.

2. A set of "servers" within one or more instances of the J2EE Web Application server must be created to house the Web application server and business application server separately. This can be on the same machine or across machines.
3. The Web application server and business application server are configured as outlined in Web application server Configuration and Business Application Server Configuration.
4. The EAR/WAR files generated are deployed separately with the **SPLWeb** EAR file deployed to the Web application server as outlined in Web Application Server Deployment Process and **SPLService** EAR file deployed to the business application server as outlined in Business Application Server Deployment Process.

In terms of the product itself there are negligible performance differences between a local or decoupled installation.

Contents

Business Application Server Concepts

In V2.2 (and above) the business application server component can be deployed within a separate instance of the J2EE Web application server software. This effectively allows the business application server to be on separate hardware for architectures where this is a requirement. Typically this separation is implemented for a number of reasons:

- The site has an architectural principle for separating the business application server and Web application server.
- The site prefers to optimize the individual servers for the individual tiers rather than having to compromise when two or more tiers are on the same platform.

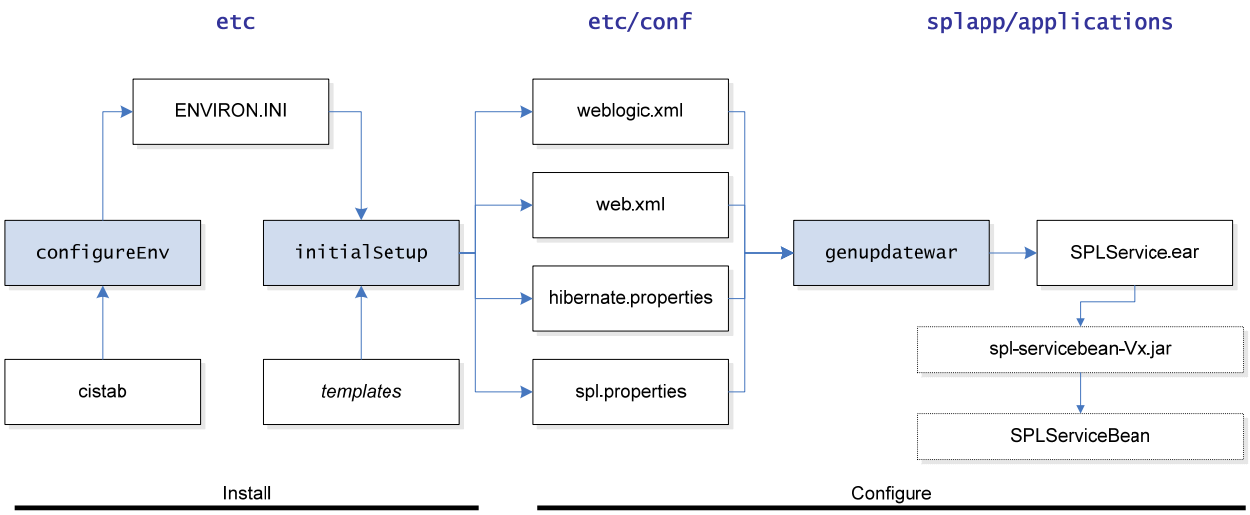
The business application server was designed to fit within the same concepts as the Web application server. The main differences are:

- Enterprise Java Beans (stateless) are used in the business application server instead of Java Server Pages as used in the Web application server. The name of the EJB is **spl-servicebean-*<version>*.jar** (where ***<version>*** is the version of the product e.g. 2.2.0).
- Database connectivity is configured in the business application server.

The rest of this section will outline the differences specifically for the business application server.

Business Application Server Configuration Process

To configure the business application server during the installation process and post-installation then the following process should be used:



- The configureEnv – Setup Environment settings utility is used during installation time and can be used post implementation to set parameters in the ENVIRON.INI - Environment Configuration File. If any parameters are derived or set from the ENVIRON.INI - Environment Configuration File (see "**Source**" column in the relevant section) then the configureEnv – Setup Environment settings should be used to maintain them.

  Note: The configureEnv – Setup Environment settings utility should be used to make ANY changes to the ENVIRON.INI - Environment Configuration File. Manual changes to this configuration file are not recommended.

- After the ENVIRON.INI - Environment Configuration File has been set or altered, the settings must be reflected in the relevant configuration files used by the business application server by running the initialSetup – Reset configuration to template utility:
  - log4j.properties – Logging Configuration
  - spl.properties – Product configuration settings
  - hibernate.properties – Database connectivity properties

Contents

- web.xml – J2EE Application Descriptor

- The utility uses the templates from the etc directory to create substituted copies of these files in a standard location:

Configuration File	Location of Configuration File
Service Bean	

Configuration File	Location of Configuration File
web.xml	Linux/Unix: **$SPLEBASE/etc/conf/WEB-INF/web.xml** Windows: **%SPLEBASE%\etc\conf\WEB-INF\web.xml**
spl.properties	Linux/Unix: **$SPLEBASE/etc/conf/root/WEB-INF/classes/spl.properties** Windows: **%SPLEBASE%\etc\conf\root\WEB-INF\classes\spl.properties**
log4j.proprties	Linux/Unix: **$SPLEBASE/etc/conf/root/WEB-INF/classes/log4j.properties** Windows: **%SPLEBASE%\etc\conf\root\WEB-INF\classes\log4j.properties**
hibernate.properties	Linux/Unix: **$SPLEBASE/etc/conf/root/WEB-INF/classes/hibernate.properties** Windows: **%SPLEBASE%\etc\conf\root\WEB-INF\classes\hibernate.properties**

- At this point you may perform manual changes to the above files to parameters not implemented in the ENVIRON.INI - Environment Configuration File.

> *Note:* Any manual changes are overwritten after running the initialSetup – Reset configuration to template utility. Backups should be made of any changes and then manually reapplied to reinstate all manual changes.

- To reflect configuration changes into the product EJB service beans the genupdatewar – Generate J2EE WAR and EAR files utility must be executed. This will build the necessary EAR/WAR files to be deployed into the J2EE Web application server.

Depending on the architecture the genupdatewar – Generate J2EE WAR and EAR files utility will generate the **SPLService.EAR** file which will contain a number of JAR files but primarily **spl-servicebean-*<version>*.jar** (where ***<version>*** is the version of the product used) which will contain the service beans.

At this point the product business applications are ready for deployment into the J2EE Web application server.

Quick Reference Guide for business application server Configuration

To make configuration changes to the business application server component of the product uses the following Quick Reference Guide to identify which process should be used:

- If the change is to any setting contained in the ENVIRON.INI - Environment Configuration File for the business application server then you must run the following utilities in the order indicated:
    1. Execute the configureEnv – Setup Environment settings utility to make the reflect the parameter change in the ENVIRON.INI - Environment Configuration File.
    2. Execute the initialSetup – Reset configuration to template utility to rebuild the configuration files using the ENVIRON.INI - Environment Configuration File and provided template file. This will reset the configuration to the contents of the template files.
    3. (*Optional*) Make any manual changes to the configuration files potentially lost in the previous Step.
    4. Execute the genupdatewar – Generate J2EE WAR and EAR files utility to implement the configuration files in the product EJB Application files.
- If the change is to any setting not contained in the ENVIRON.INI - Environment Configuration File for the business application server but is in the configuration files for the business application server then you must run the following utilities in the order indicated:
    1. (*Optional*) Make any manual changes to the relevant configuration files.
    2. Execute the genupdatewar – Generate J2EE WAR and EAR files utility to implement the configuration files in the product EJB Application files.

Business Application Server Deployment Process

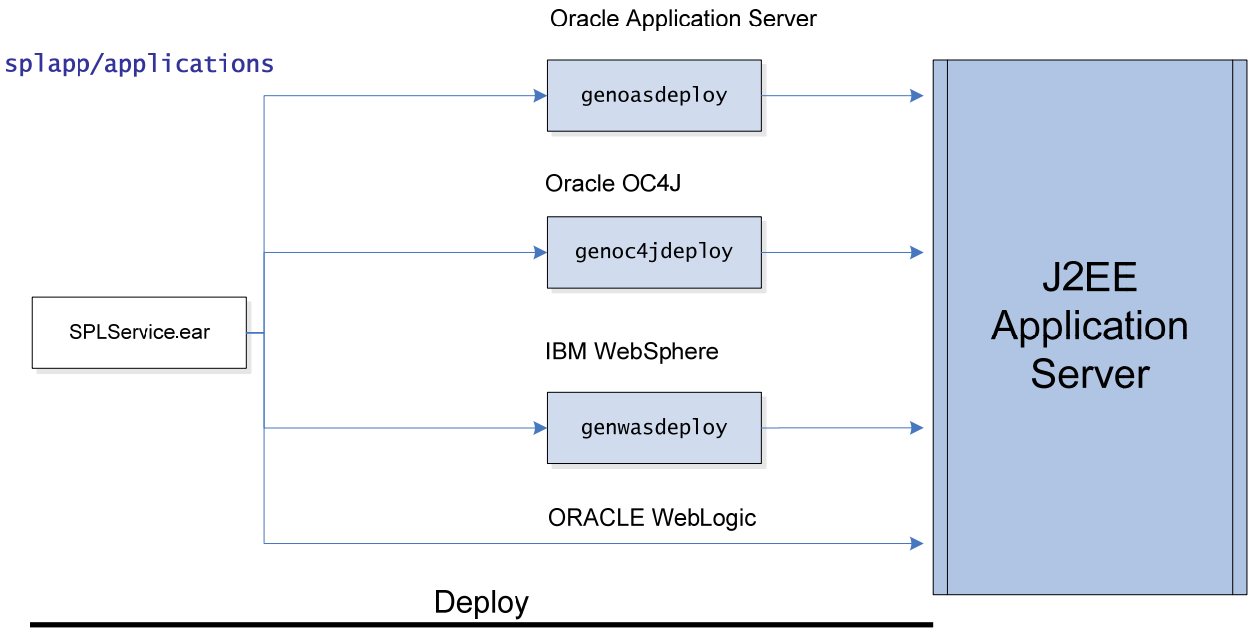
After the configuration of the business application server is complete (as outlined in Business Application Server Configuration Process) the final step to implement the product technically is to deploy the product within the J2EE Web application server.

There are three methods of deploying the product within the J2EE Web application server:

- Use the deployment utilities provided on the console of the J2EE Web application server. The EAR/WAR files that are available under **`$SPLEBASE/splapp/applications`** (or **`%SPLEBASE%\splapp\applications`** for Windows) can be manually deployed using the console. Refer to the Installation Guide for specific platform instructions and the administration guide for the J2EE Web application server.
- Use the deployment utilities provided on the command line of the J2EE Web application server. The EAR/WAR files that are available under **`$SPLEBASE/splapp/applications`** (or **`%SPLEBASE%\splapp\applications`** for Windows) can be manually deployed using the J2EE Web application server vendor supplied deployment command line utilities. Refer to the Installation Guide for specific platform instructions and the administration guide for the J2EE Web application server.
- A number of specific utilities for J2EE Web Application are provided with the product to deploy the EBJ Application to the J2EE Web application server. These call the same utilities provided in the previous option but are provided with the product.

  This section will outline the latter option.

A number of utilities are provided in the **`bin`** directory to deploy the product to the J2EE Web application server. These utilities are outlined below:



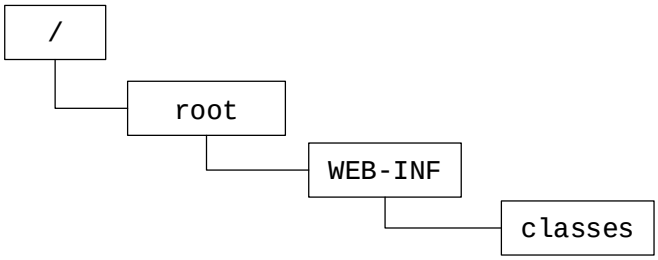
- For the ORACLE Application Server platform, use the genoasdeploy – Deploy on ORACLE Application Server utility.
- For the ORACLE OC4J platform, use the genoc4jdeploy – Deploy on OC4J utility.
- For the IBM WebSphere platform, use the genwasdeploy – Deploy on WebSphere utility.
- For ORACLE WebLogic, no additional deployment is necessary as the product automatically detects WebLogic and allows WebLogic to read the EAR/WAR files directly.

These utilities will attempt to deploy the EJB Applications within the J2EE Web application server as follows:

J2EE Web application server	Deployment details
ORACLE WebLogic	Deployed to root application by default using **BSN_SYSUSER** and **BSN_SYSPASS** from ENVIRON.INI - Environment Configuration File as administration credentials.
ORACLE Application Server	Deployed to **BSN_OASAPP** application by default using **BSN_OASYSUSER** and **BSN_OASYSPASS** from ENVIRON.INI - Environment Configuration File as administration credentials.
ORACLE OC4J	Deployed to **BSN_OC4J_instance** application by default using **BSN_OC4JSYSUSER** and **BAN_OC4JSYSPASS** from ENVIRON.INI - Environment Configuration File as administration credentials.
IBM WebSphere	Deployed to **BAN_WASAPP** Application on **BAN_SVRNAME** server by default using **BSN_WASUSER** and **BSN_WASPASS** from ENVIRON.INI - Environment Configuration File as administration credentials.

Business Application Server Configuration Files

Each J2EE Web Application within the J2EE Web application server has its own configuration files. These files are typically "embedded" within the EAR/WAR files deployed with the product following the J2EE specification. In terms of configuration, the product structure within the EAR/WAR file looks like the following:



Location	Contents	Configuration Files
root/WEB-INF	J2EE Application Descriptor for business application server	
root/WEB-INF/classes	Application Configuration files for business application server	log4j.properties – Logging Configuration spl.properties – Product configuration settings hibernate.properties – Database connectivity properties

Contents

web.xml – J2EE Application Descriptor

The Web deployment descriptor editor lets you specify deployment information for modules created in the Web development environment. The information appears in the **web.xml** file. The **web.xml** file for a Web project provides information necessary for deploying a Web application module. It is used in building a WAR/EAR file from a project.

The Business Application is controlled by a configuration file that holds behavioural information for the applications. Refer to **http://jcp.org/en/jsr/detail?id=109** for more details of the format. For example:

```
…
   <env-entry>
      <description>How long to cache drop down values in seconds</description>
      <env-entry-name>fieldValuesAge</env-entry-name>
      <env-entry-value>3600</env-entry-value>
      <env-entry-type>java.lang.Integer</env-entry-type>
   </env-entry>
…
```

The following settings apply to Web Application Descriptor for the business application server:

web.xml Parameter	Context	Source
fieldValuesAge	How long the static cache is kept on the Web application server in seconds?	Defaulted to **3600** in template

Note: It is *highly* recommended that you do *not* change this configuration file by extracting the configuration file from the EAR/WAR file using java utilities, making the change manually and rebuilding the EAR/WAR. Use the genupdatewar – Generate J2EE WAR and EAR files utility to build the EAR/WAR file as documented in Business Application Server Configuration Process

log4j.properties – Logging Configuration

Note: This log file should *not* be altered unless specified. The generated configuration file has *all* the recommended settings for *all* sites.

The product uses the log4j java classes to centralize all log formats into a standard format. The details of the configuration settings and log4j itself is available at http://logging.apache.org/log4j/ or http://en.wikipedia.org/wiki/Log4j

spl.properties – Product configuration settings

The product business application has a specific number of settings outside of the J2EE specification to control the internals of the product. This file exists as similar files exist for ALL modes of operation of the product (for example, Batch can be run outside the J2EE Web application server) so a common configuration standard was adopted.

For the business application server the **spl.properties** uses the following settings:

Parameter	Context	Source
spl.tools.loaded.applications	List of applications installed. Values are typically **base,ccb,cm**	Generated by installation script.
spl.runtime.cobol.cobrcall	If COBOL is used, whether remote calls are supported. (**true** or **false**). Defaults to **false**.	Generated from template.
spl.runtime.utf8Database	Whether the database supports the UTF8 characterset. (**true** or **false**)	Derived from **NLS_LANG** or **DB2CODEPAGE** parameters from ENVIRON.INI - Environment Configuration File

Parameter	Context	Source
spl.runtime.cobol.remote.jvm	If COBOL is used, whether the COBOL Child JVM's be considered remote JVM's. (**true** or **false**). Defaults to **true**.	Generated from template.
spl.runtime.cobol.remote.rmiStartPort	If COBOL is used, Starting port range for COBOL Child JVM's. Defaults to **6503**	Derived from **BSN_RMIPORT** parameters from ENVIRON.INI - Environment Configuration File
spl.runtime.cobol.remote.jvmcount	If COBOL is used, the Number of COBOL Child JVM's. Default is **2**.	Derived from **BSN_JVMCOUNT** parameters from ENVIRON.INI - Environment Configuration File
spl.runtime.cobol.remote.jvmcommand	If COBOL is used, the Java executable to be used for COBOL Child JVMs.	Derived from **Jvmcommand** parameters from ENVIRON.INI - Environment Configuration File
spl.runtime.cobol.remote.jvmoptions	If COBOL is used, the Java memory footprint to be used for COBOL Child JVMs.	Derived from **JVMMEMORYARG** parameters from ENVIRON.INI - Environment Configuration File
spl.runtime.sql.highValue	High value to be used for java sorting routines.	Derived from **highValue** parameters from ENVIRON.INI - Environment Configuration File
spl.runtime.cobol.encoding	If COBOL is used, the character set supported by the Web application server	Derived from **NLS_LANG** or **DB2CODEPAGE** parameters from ENVIRON.INI - Environment Configuration File

Parameter	Context	Source
spl.runtime.oracle.statementCacheSize	The SQL cache size allocation for ORACLE SQL statements (ORACLE platforms only). Defaults to **300**.	Defaulted from template
spl.runtime.cobol.sql.fetchSize	Size of fetch buffers for SQL statements. Defaults to **150**.	Defaulted from template
spl.runtime.cobol.sql.cache.maxTotalEntries	Number of SQL statement entries stored in the cache. Defaults to **1000**.	Defaulted from template
spl.runtime.cobol.sql.cursoredCache.maxRows	Number of cursors cached. Defaults to **10**.	Defaulted from template
spl.runtime.cobol.sql.disableQueryCache	False	Defaulted from template
spl.runtime.socket.file.dir	Working directory for workable sockets	Defaulted from template
spl.runtime.environ.init.dir	Location of the base configuration files.	Defaulted from template
com.splwg.grid.online.enabled	Whether the online daemon is uses the lightweight batch framework or not. (**true** or **false**)	Derived from **BATCHENABLED** parameter from ENVIRON.INI - Environment Configuration File
com.splwg.grid.distThreadPool.threads.DEFAULT	Maximum number of threads (jobs or threads of a job) supported concurrently by batch daemon. Defaults to **5**	Derived from **BATCHTHREADS** parameter from ENVIRON.INI - Environment Configuration File
com.splwg.batch.scheduler.daemon	Whether the online daemon is used or not. (**true** or **false**)	Derived from **BATCHDAEMON** parameter from ENVIRON.INI - Environment Configuration File
spl.runtime.service.extraInstallationServices	Name of Application service used for	Defaulted by

Parameter	Context	Source
	installation defaults. Default: **CILTINCP**	template.
spl.serviceBean.jndi.name	Name of internal EJB JNDI	Derived from template
spl.ejbContainer.contextFactory	EJB Container factory	Defaulted by template.
spl.ejbContainer.url	URL to business application server	Derived from various parameters from ENVIRON.INI - Environment Configuration File
spl.ejbContainer.user	Administration user to EJB Container	Derived from various parameters from ENVIRON.INI - Environment Configuration File
spl.ejbContainer.password	Administration password to EJB Container	Derived from various parameters from ENVIRON.INI - Environment Configuration File
spl.runtime.cobol.remote.jvmMaxLifetimeSecs	Amount of time in seconds before automatic shun of COBOL child JVM's	Manual
spl.runtime.cobol.remote.jvmMaxRequests	Number of COBOL requests before automatic shun of COBOL child JVM's	Manual

hibernate.properties – Database connectivity properties

Opening a connection to a database is generally much less expensive than executing an SQL statement. A connection pool is used to minimize the number of connections opened between application and database. It serves as a librarian, checking out connections to application code as needed. Much like a library, your application code needs to be strict about returning connections to the pool when complete, for if it does not do so, your application will run out of available connections. Hence, the need for having a connection pooling mechanism such as Hibernate using C3P0 connection pooling.

Hibernate is a powerful Object Relational Mapping (ORM) technology that makes it easy to work with relational databases. Hibernate makes it seem as if the database contains plain Java objects, without having to worry about how to get them out of (or back into) database tables. Coupled with the C3P0 connection pooler, it provides a comprehensive connectivity tool for the COBOL/java to operate effectively against the database.

The product uses the Hibernate and C3P0 libraries to create a connection pool and connect the java/COBOL objects to the database to store, update, delete and retrieve data. It is used for all the database access for online as well as batch.

Refer to http://www.hibernate.org and http://sourceforge.net/projects/c3p0 for more information on the technology aspects of Hibernate and C3P0.

The product has a configuration file for the database connectivity and pooling called the **hibernate.properties** configuration file. This file contains the configuration settings for the database connections and the connection pool to be used by any of the SQL statements accessing the database.

The configuration settings contained in the **hibernate.properties** file are summarized in the following table:

Setting	Usage
hibernate.connection.driver_class	This is the JDBC driver class used by Hibernate.
hibernate.connection.url	This is the connection string used to connect to the database. The URL is built using the protocol outlined by the JDBC driver and uses the values from the ENVIRON.INI - Environment Configuration File.
hibernate.connection.username	This is the user IDused to connect to the database. This value is sourced from the **DBUSER** parameter from the ENVIRON.INI - Environment Configuration File
hibernate.connection.password	This is the user IDused to connect to the database. This value is sourced from the **DBPASS** parameter from the ENVIRON.INI - Environment Configuration File. If the value is prefixed by "ENC" then the password is encrypted.
hibernate.dialect	This is the SQL dialect (database type) for the database being used. Any valid Hibernate dialect may be used. Refer to http://www.hibernate.org/hib_docs/v3/api/org/hibernate/dialect/package-summary.html for a full list. This value is sourced from the **DIALECT** parameter from the ENVIRON.INI - Environment Configuration File.
hibernate.show_sql	Write all SQL statements to console. Defaults to **false**.
hibernate.max_fetch_depth	Sets a maximum "depth" for the outer join fetch tree for single-ended associations (one-to-one, many-to-one). A 0 disables default outer join fetching. Defaults to **2**
hibernate.transaction.factory_class	The classname of a Transaction Factory to use with

Setting	Usage
	Hibernate Transaction API.
hibernate.cglib.use_reflection_optimizer	Enables use of CGLIB instead of runtime reflection (System-level property). Reflection can sometimes be useful when troubleshooting, note that Hibernate always requires CGLIB even if you turn off the optimizer. Tends to make Hibernate load faster if value is **false**. Defaults to **false**.
hibernate.jdbc.fetch_size	Determines a hint to the JDBC driver on the the number of rows to return in any SQL statement. Defaults to **100**
hibernate.jdbc.batch_size	A non-zero value enables use of JDBC2 batch updates by Hibernate. Defaults to **30**
hibernate.query.factory_class	Chooses the HQL parser implementation.
hibernate.cache.use_second_level_cache	May be used to completely disable the second level cache, which is enabled by default for classes which specify a **<cache>** mapping. Defaults to **false**
hibernate.query.substitutions	Mapping from tokens in Hibernate queries to SQL tokens (tokens might be function or literal names, for example). The product uses **true 'Y', false 'N'**
hibernate.connection.provider_class	The classname of a custom Connection Provider which provides JDBC connections to Hibernate. product uses the C3P0 Connection provider. Other providers are not supported.
hibernate.c3p0.min_size	Initial number of database connections. Defaults to **1**
hibernate.c3p0.max_size	Maximum number of database connections to open. Defaults to **150**
hibernate.c3p0.timeout	Maximum idle time for a connection (in seconds) Defaults to **300**
hibernate.c3p0.max_statements	Maximum size of c3p0 statement cache (0 to turn off). Defaults to **0**
hibernate.c3p0.idle_test_period	Idle time before a c3p0 pooled connection is validated (in seconds). Defaults to **0**
hibernate.c3p0.acquire_increment	Number of connections in a clump acquired when pool is exhausted. Defaults to **1**

For a more indepth description of these parameters and others not included with the product see http://www.hibernate.org and http://sourceforge.net/projects/c3p0.

Miscellaneous Operations And Configuration

Enabling Email Logging from Log4j

The following sample configuration will enable email logging of **ERROR** level log messages in the product. When an error is encountered in startup and during operations of the product any ERROR message displayed on the console log file will be emailed to an Administrator's email account or email group.

Note: This change outlined below will make manual changes to a configuration file. Execution of initialSetup – Reset configuration to template may overwrite these changes. Please ensure you make adequate backups to preserve this change. Refer to http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/net/SMTPAppender.html for details of the appender.

The following changes are required to enable this.

1) Open the **log4j.properties** in the relevant **$SPLEBASE/etc/conf** (**%SPLEBASE%\etc\conf** in Windows) subdirectory:
   - Web application server – log4j.properties – Logging Configuration
   - business application server - log4j.properties – Logging Configuration
2) Add the following lines to the file:

```
###  E1 is an EmailAppender
log4j.appender.E1 = org.apache.log4j.net.SMTPAppender
log4j.appender.E1.Threshold = ERROR
log4j.appender.E1.layout = org.apache.log4j.PatternLayout
log4j.appender.E1.layout.ConversionPattern = %d{ISO8601} [%t] %-5p %c %x - %m%n
log4j.appender.E1.From = <from>
log4j.appender.E1.SMTPHost = <SMTPHost>
log4j.appender.E1.Subject = <subject>
log4j.appender.E1.To = <to>
###
### The following settings are optional
###
log4j.appender.E1.SMTPUsername = <SMTPUsername>
log4j.appender.E1.SMTPPassword = <SMTPPassword>
log4j.appender.E1.CC = <cc>
log4j.appender.E1.BCC = <bcc>
```

Parameter	Field from example	Usage
From	*<from>*	Email address for emails

Parameter	Field from example	Usage
To	***<to>***	Email address/group to send emails to
CC	***<cc>***	Email address/group to send courtesy copy of emails to
BCC	***<bcc>***	Email address/group to send "blind" courtesy copy of emails to
SMTPHost	***<SMTPHost>***	Host Name of SMTP Server
SMTPUsername	***<SMTPUsername>***	Logon User for SMTP Server (if supported)
SMTPPassword	***<SMTPPassword>***	Password for Logon User for SMTP Server (if supported)
Subject	***<subject>***	Subject for email message

3) Modify the following lines in the **log4j.properties** file:

```
## System-wide settings
# set log levels - for more verbose logging change 'info' to 'debug' ###
log4j.rootCategory=info, A1, F1, E1
```

4) Add **activation.jar** and **mail.jar** to **$SPLEBASE\etc\conf\root\WEB-INF\lib** (**%SPLEBASE%/etc/conf/root/WEB-INF/lib** for Windows) for the Web application server (root application). These files can be obtained from http://java.sun.com/javase/technologies/desktop/javabeans/jaf/downloads/index.html and http://java.sun.com/products/javamail/downloads/index.html respectively.

5) Execute the genupdatewar – Generate J2EE WAR and EAR files utility to reflect the changes in the EAR/WAR file.

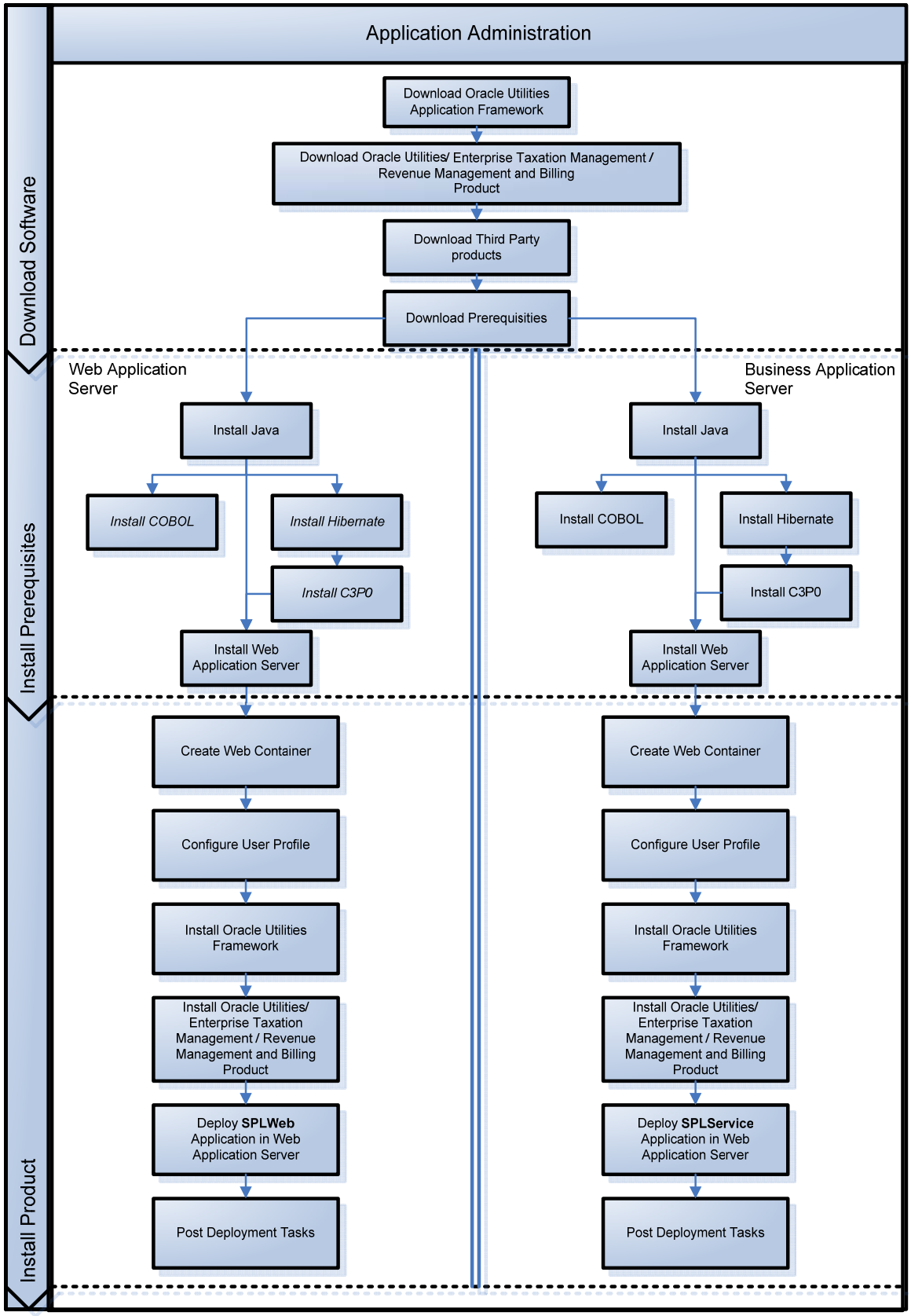
6) To deploy the application refer to the Web Application Server Deployment Process or Business Application Server Deployment Process

Installation of decoupled servers

One of the options available with the V2.2 the product is the ability to split the Business Application Server from the Web Application Server. This facility requires an additional set of installation processes from the normal installation processes. This section will outline these additional steps.

Note: The guidelines below outline a generic process for installing the decoupled servers. Please refer to the Installation Guide for platform specific guidelines for each part of the process.

The following diagram illustrates the process:



The installation of the decouple server configuration is performed in three phases:

- **Download Software** – Download the required software packages from the relevant sources prior to installation. This process is documented in detail in the Installation Guide.
- **Install Prerequisites** – Installation of the prerequisite software required by product. This process is documented in detail in the Installation Guide.
- **Install Product** – Installation of the components of the model on each tier.

The installation of a decoupled server is similar in installing the default installation with the following differences:

- All the prerequisite software is installed on both Web Application Server and Business Application Server tiers. This also allows for the role of the tier to be decided at deployment time not installation time. The role of the tier can be modified using configuration only instead of requiring installation.
- Both tiers are housed in a J2EE Web Application Server. The failover and clustering facilities of the J2EE Web Application Server can be used to provide high availability. Refer to the J2EE Web Application server documentation supplied by your respective vendor for information about high availability configuration.
- At deployment time the **SPLWeb.EAR** file is deployed to the Web Application Server and the **SPLService.EAR** file to be deployed to the Business Application Server either using the J2EE Web Application provided utilities or the deployment utilities documented in the previous sections of this document.
- The configuration settings for the Web Application Server must refer to the appropriate settings for the Business Application Server. Refer to the Business App Server Configuration in the Installation documentation for the settings applicable for your product.

Refer to the relevant sections of the Installation Guide for all the details for each of the tasks of the process.

Overriding the default Oracle database connection information

Note: Single Fix 7924537 must be applied prior to using this configuration facility. NEW

By default the database connection information configured by the product is of the format:

jdbc:oracle:thin:@*<hostname>*:*<dbport>*:*<database>*

where

<hostname>	Database hostname
<dbport>	Database Listener portname
<database>	Database Name

The URL format is described at
http://www.oracle.com/technology/tech/java/sqlj_jdbc/htdocs/jdbc_faq.html#05_03

This configuration setting is sufficient for the majority of the environments at a site. If your site requires a specialist URL for RAC support then you must override the default URL.

To enable the use of this functionality, you must create a file called **dbconn_override.txt** in the environment under the **$SPLEBASE\etc** for Linux/UNIX (or **%SPLEBASE%/etc** for Windows) folder containing the override string. This setting will replace the database connection URL used by all modes of access at configuration time.

To reflect the change into the product you must:

- Execute **configureEnv[.sh]** to detect the override file when using the "**p**" option to process the configuration file.

Note: The **dbconn_override.txt** must exist each time you run **configureEnv[.sh]**

- Execute **initialSetup[.sh]** to place the override in the appropriate **hibernate.properties** files.

Note: When running **initialSetup[.sh]** any manual customization (outside of the normal configuration process) will need to be manually reapplied.

- Redeploy the Web Applications using the appropriate deploy commands.

Examples of dbconn_override.txt file

These are a few examples of database connection overrides.

Example contents of **dbconn_override.txt** using Oracle JDBC thin client (for Oracle Real Application Clustering):

```
jdbc:oracle:thin:@(DESCRIPTION =(ADDRESS = (PROTOCOL = TCP)(HOST = machine-
name)(PORT = 1251))
    (ADDRESS = (PROTOCOL = TCP)(HOST = machine-name)(PORT = 1251)
    (LOAD_BALANCE = yes)
    (FAILOVER=YES)
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = SID.WORLD)
    )
)
```

Example contents of **dbconn_override.txt** Oracle JDBC thick client:

```
jdbc:oracle:oci:@SID.WORLD
```

Note: For thick client to work, the Oracle client library directory must be added to the library search path. Oracle client libraries are installed under **ORACLE_HOME/lib** and **ORACLE_HOME/lib32** directories. Add this directory to the library search path environment variable. The library search path environment for AIX is **LIBPATH**, for HP-UX is **SH_LIB_PATH** for Linux is **LD_LIBRARY_PATH** and for Windows is **PATH**.

Automatic shunning of Child COBOL JVM's

For products that use COBOL there are a series of COBOL Child JVMs created for products that support COBOL using the Oracle Utilities Application Framework for backward compatibility. This is primarily used to transfer data between the java based framework and any remaining COBOL based business objects.

There are instances when the COBOL processes hosted in "child" Java virtual machines can consume too many resources, e.g. running out of “native” memory. In the event that such a situation obtains, and cannot be resolved by e.g. identifying a problematic COBOL module, it is necessary to shutdown (“shun”) the OS process that hosts COBOL in order to reclaim the resources.

In these situations is possible to configure the system to automatically "shun" a COBOL child JVM in order to forestall a possible situation where the process consumes too many resources. This facility allows both time-based and request-based scheduling for an automated rollover to a standby JVM.

Optionally a facility has been created that allows for an automatic rollover from the active COBOL child JVM to a standby JVM, without disrupting any system processing. In order to allow this, the system must be configured to use at least two (2) child JVMs, to assure a near-instantaneous switchover to the standby JVM.

The feature is activated by placing either, or both, of the following properties into the **spl.properties** that govern the child JVM:

spl.runtime.cobol.remote.jvmMaxLifetimeSecs=*[number of seconds]*

spl.runtime.cobol.remote.jvmMaxRequests=*[number of COBOL requests]*

Set either property to zero (or leave it out) to disable the relevant rollover policy.

- If the JVM max lifetime seconds parameter is set to e.g. 3600 for one hour, then one hour after the first request is made to that child JVM, it will be automatically "shunned", completing all in-flight requests normally, while transferring all new work to the standby child JVM.
- If the JVM max requests parameter is set to e.g. 50000, then after 50000 COBOL commands have been sent to the child JVM, it will be automatically “shunned” as above.
- When both parameters are provided, the child JVM will be shunned automatically when either condition obtains, e.g. shun after one hour, or 20000 COBOL commands, whichever comes first

It is recommended starting with the values of 3600/50000 if you are encountering problems with the COBOL child JVM exhausting its resources (e.g. COBOL-level memory allocation failures).

These policies are not active in the default configuration as part of the installation process.

The system creates log file entries when a rollover condition has been satisfied.

Cache Management

A great deal of information in the system changes infrequently. In order to avoid accessing the database every time this type of information is required by an end-user, the system maintains a cache of static information on the web server. In addition to the web server’s cache, information is also cached on each user’s browser.

Server Cache

The cache is populated the first time any user accesses a page that contains cached information. For example, consider a control table whose contents appear in a dropdown on various pages. When a user opens one of these pages, the system verifies that the list of records exists in the cache. If so, it uses the values in the cache. If not, it accesses the database to retrieve the records and saves them in the cache. In other words, the records for this control table are put into the cache the first time they are used by any user. The next user who opens one of these pages will have the records for this control table retrieved from the cache (thus obviating the database access).

The following points describe the type of data that is cached on the web server:

- Field labels. This portion of the cache contains the labels that prefix fields on the various pages in the system.
- System information. This portion of the cache contains installation and license key information as well as basic information about the various application services (e.g., the URL's that are associated with the various pages).
- Menu items. This portion of the cache contains the menu items.
- Dropdown contents. This portion of the cache contains the contents of the various dropdowns that appear throughout the system.
- XSL documents. This portion of the cache contains each page's static HTML.
- Portal information. This portion of the cache contains information about which zones are shown on the various pages.

The contents of the cache are cleared whenever the web server is "bounced". This means that fresh values are retrieved from the database after the application server software is restarted.

If you change the database after the cache is built and the information you changed is kept in the cache, users may continue to see the old values. If you don't want to bounce your web server, you can issue one of the following commands in your browser's URL to immediately clear the contents of the cache (a separate command exists for each type of data that is held in the cache):

Command	Effect
`flushAll.jsp?language=<language code>`	This command flushes every cache described below. *Note:* The `language=<language code>` option is optional.
`flushMessageCatalog.jsp`	This command flushes the field labels. Use this whenever you add or change any labels (this is typically only done by implementers who have rights to introduce new pages).
`flushSystemLoginInfo.jsp`	This command flushes the system information cache. Use this whenever you change navigation options, cached information from the installation record, or if you change an application service's URL.
`flushMenu.jsp`	This command flushes the menu cache. Use this command if you change menu data.
`flushDropdownCache.jsp?language=<language code>`	This command flushes ALL objects in the dropdown contents associated with a given language (note, the language code is defined on the Language control table). Use this whenever you change, add or delete values to/from control tables that appear in dropdowns. Unlike the above caches, the

Command	Effect
	objects in the dropdown cache are flushed automatically every 30 minutes from the time they are first built. *Note:* The **language=*<language code>*** option is optional.
flushDropDownField.jsp?*language=<language code>&key=<field>*	If you want to flush the values in a specific drop-down field in the cache, specify the <field> using this command. *Note:* This command is designed primarily for developers.
flushUI_XSLs.jsp	This command flushes the XSL document cache. Use this command if you change user interface meta-data. Also use this command whenever you change or introduce new zones.
flushXMLMetaInfo.jsp	This command flushes the XML repository of the XML Application Interface. Use this command when your site's XAI configuration data needs to be updated, which should be a very rare occurrence.
flushPortalMetaInfo.jsp	This command flushes the portal information cache. Use this command whenever you change any portal zone meta-data

Client Cache

In addition to the web server's cache, information is also cached on each user's browser. After clearing the cache that's maintained on the web server, you must also clear the cache that's maintained on your client's browser. To do this, follow the following steps:

- Select Tools on your browser's menu bar
- Select Internet Options … on the menu that appears.
- Click the Delete Files button on the pop-up that appears.
- Turn on Delete all offline content on the subsequent pop-up that appears and then click OK.
- And then enter the standard URL to re-invoke the system.

Note: Each user's cache is automatically refreshed based on the **maxAge** parameter. We recommend that you set this parameter to 1 second on development / test environments and 28800 seconds (8 hours) on production environments.