Server Administration Guide

Oracle Utilities Meter Data Management
Version 2.0.0 (OUAF 4.0.2)

E18183-01

July 2010

ORACLE®

Server Administration Guide, Oracle Utilities Meter Data Management, Version 2.0.1 - OUAF 4.0.2

E18183-01

Primary Author: Anthony Shorten, Principal Product Manager, Tax And Utilities Global Business Unit

Table of Contents

Preface

Introduction

Welcome to the Oracle Utilities Meter Data Management Server Administration Guide for Version 2.0.1. 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 Meter Data Management product will be referred to as "product" in this document.

Note: All utilities in this guide are multi-platform (unless otherwise indicated). For publishing purposes the commands will be in the format ***command[.sh]*** *which indicates that the command can be used as* ***command*** *on the Windows platform or* ***command.sh*** *on the Linux/UNIX platforms.*

Note: Any changes to the version of this document from previous releases of the Oracle Utilities Application Framework or as a result of service packs are marked with a NEW *graphic.*

Note: Sections of this manual cover the batch aspects of the Oracle Utilities Application Framework for completeness only. Products that use the batch component of the Oracle Utilities Application Framework should refer to the dedicated Batch Server Administration Guide for specific advice about that component. Sections covering the Batch component are marked with a BATCH *graphic.*

Note: This document now covers aspects of the mobile framework used for mobile based products. The relevant settings for this component are marked with a MWM *graphic.*

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 My Oracle Support (http://support.oracle.com). Please refer to My Oracle Support for more information.

This document is regularly updated and should be re-downloaded on a regular basis. The Service Pack that applies to this document is indicated on the initial page of this document after the product version number. This document currently applies to OUAF 4.0.2.

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:

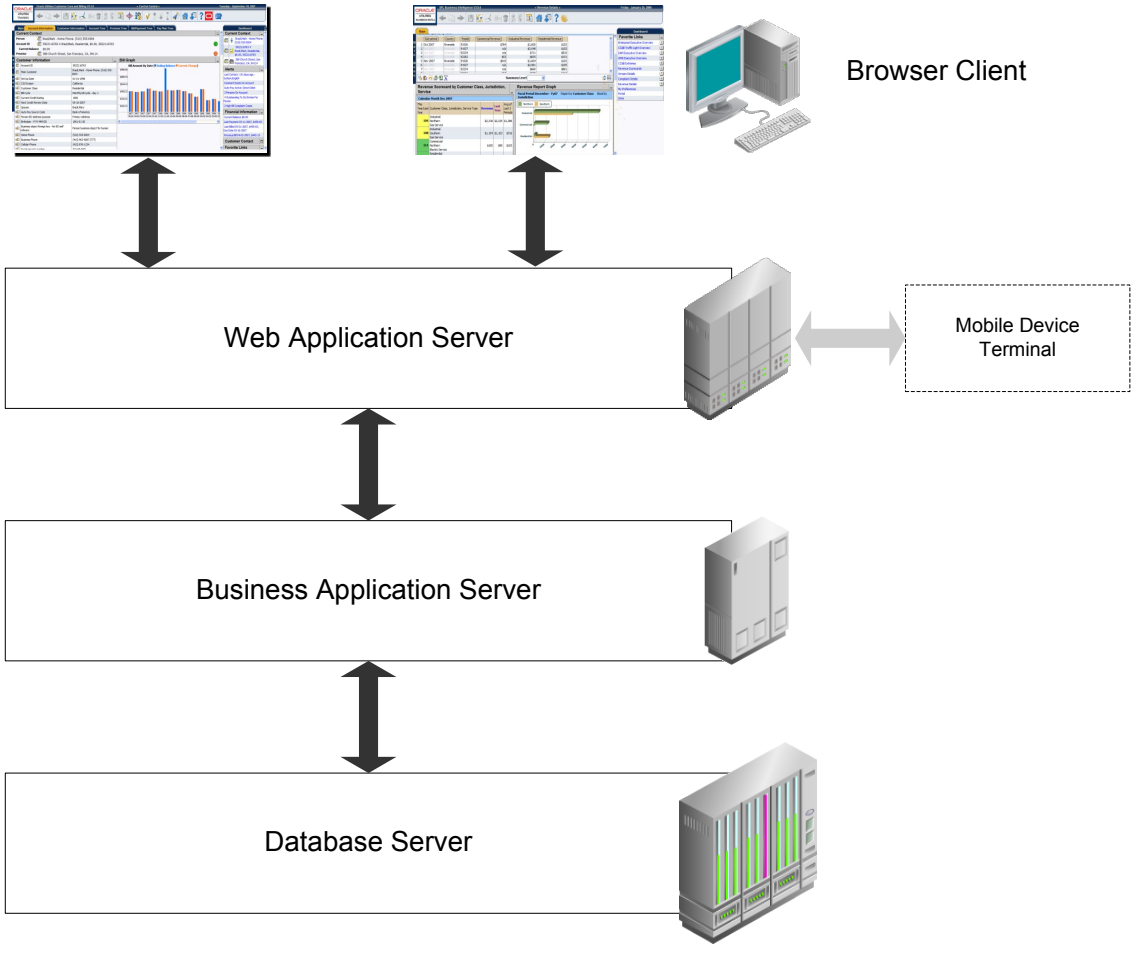
- *Oracle Utilities Meter Data Management Installation Guide*
- *Oracle Utilities Meter Data Management Quick Installation Guide*

- *Oracle Utilities Meter Data Management DBA Guide*

These documents are available from http://edelivery.oracle.com

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:

- **Browser 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. 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.

  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.
- **Mobile Device Terminal** – In some products the Mobile framework is deployed to allow mobile devices to interact with server processes. Refer to the product guides for applicability of the mobile framework to your product.

  ***Note**: This manual has minimal information about the operation of the Mobile component of*

the Oracle Utilities Application Framework.

- **Web Application Server (WAS)** – 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 metadata 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 (BAS)** – The business component of the architecture can be installed as part of the Web application server (default) 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 Java object. It contains all the SQL, programmatic rules and structures to manage the data for the transactions. In some products.
  - **DB Connection Pool** – If any database access is required, we use an industry component called Universal Connection Pool to manage and pool the connections to the database for the batch component and use the Web Server's own native JDBC connection pooling for the online and Web Services component. 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.

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.

Note: Refer to the installation guide for the supported browsers and the supported versions of those browsers.

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 they can be added and used for the implementation.

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.

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 start-up 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.

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.

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 is installed using the guidelines outlined in the Environment Management document in the Software Configuration Management series KB Id: **560401.1** on My Oracle Support.

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 **splsys** (`SPLADMIN` parameter and environment variable) and the group is **splusr** (`SPLADMINGRP` parameter and environment variable). However, alternative values can be used according to your site standards.

The administration userid is responsible for the following:

- It is the owner of the majority of the files installed for the product.
- It is the only userid that should be used to run any of the administration tools provided with the product.
- It is the userid that owns the UNIX resources used by the product. When the product is running, this userid owns the processes associated with running the base software.

The administration userid 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 userid. Instead, an alternative solution may be sought (such as using *sudo* or similar security tools).

The administration userid should not be used for any of the following:

- As a product end user. By default, the administration userid 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 userid as **splsys**. If your site uses an alternative userid as the administration userid, substitute that userid value for **splsys**.

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. 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 **splsys** (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 (denoted in this documentation as ***<SPLDIR>***)
- Application output directory structure / log directory (denoted in this documentation 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 batch) 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.*

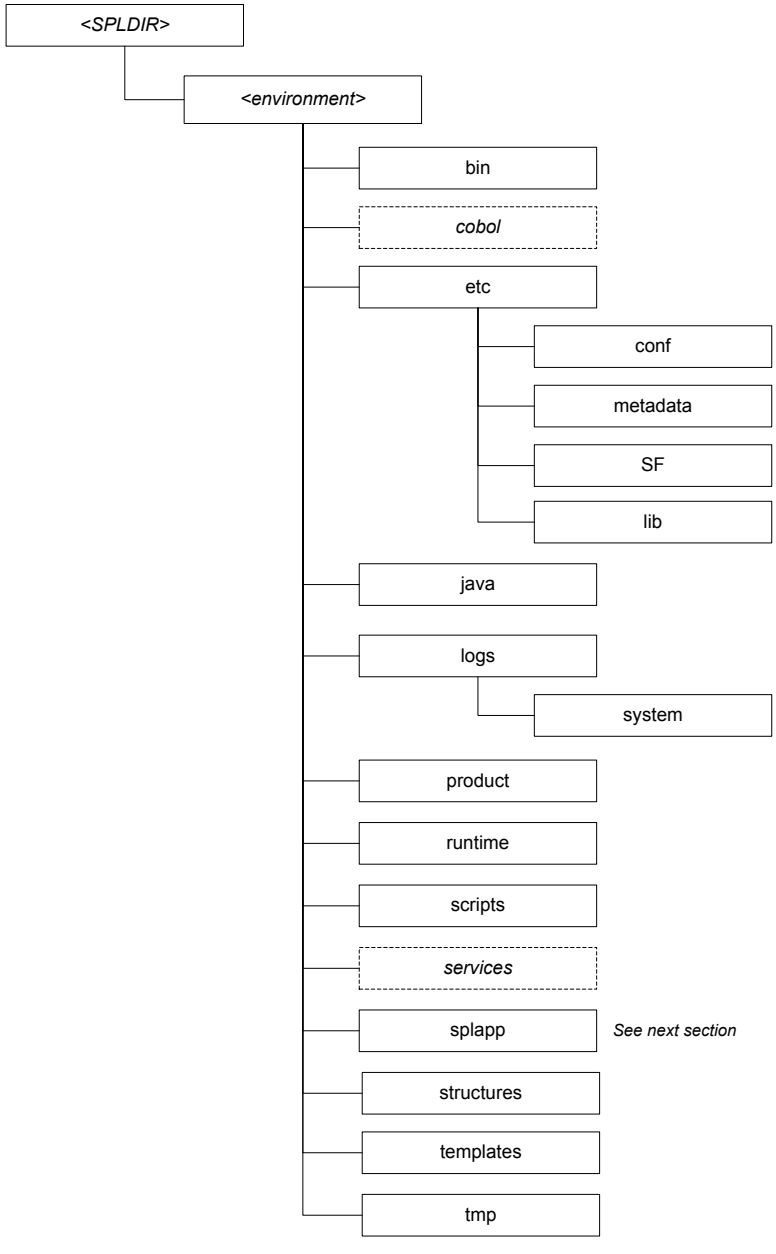
The actual location for the mount points can differ per environment if you want. This is handy if you need to vary the location because you do not have enough space for all your non-production environments. Typically the number of environments during an implementation varies according to the level of 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.

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 figure depicts the layout of where the product code is placed upon installation into the file system (where ***<environment>*** is the environment name chosen during the installation process):

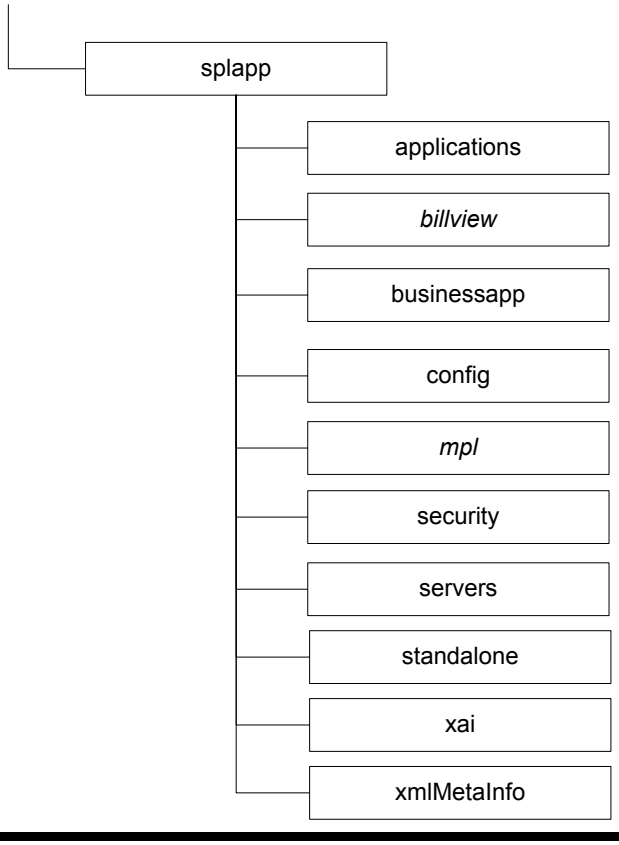


The following table outlines the typical contents of these directories:

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 and base libraries used to generate the configuration files.

Directory	Contents
java NEW	Location of temporary files for java execution
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.
services	For products that support COBOL, directory containing COBOL source service definitions for the development kit and compilation
splapp	Directories containing the J2EE Web Applications (see below)
structures NEW	Internal structures used for configuration utilities
templates NEW	Base templates used to build configuration files
tmp NEW	Directory used to hold intermediary files used for the deployment process

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



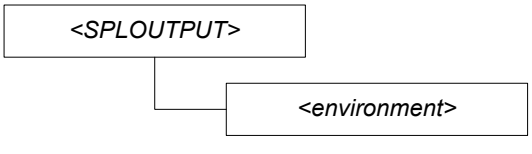
Directory	Contents
applications	Location of the Web application product files

Directory	Contents
billview	Location of the online bill viewing files (Products supporting bill view only)
businessapp	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 (Products supporting mpl only)
myserver	Reserved for future use.
security	Default location of domain security initialization files (Oracle WebLogic only)
servers	Default location of copies of configuration and associated files (Oracle WebLogic only)
standalone	Location of common Java libraries and the batch component of the product. Used for batch component.
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 environment variable **SPLOUTPUT**. By default, this directory is created as **/spl/splapp**, though this can be substituted for an alternative during the installation process. The figure below illustrates the typical directory structure for this location (where ***<environment>*** is the environment name chosen during the installation process):



The implementation may add subdirectories as their site standards and implementation dictates.

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 some of the key environment variables:

Variable	Usage
ADF_HOME	Location of the Oracle ADF files NEW
ANT_ADDITIONAL_OPT	Options for *ant* as per the configuration settings at installation time NEW
ANT_HOME	Location of *ant* build utilities
ANT_OPTS	Options for *ant* for Oracle SDK
ANT_OPT_MAX	Maximum memory settings for *ant* as per the configuration settings at installation time NEW
ANT_OPT_MIN	Minimum memory settings for *ant* as per the configuration settings at installation time NEW
CHILD_JVM_JAVA_HOME	Location of JVM used for COBOL integration *(COBOL based products only)* NEW
CMPDB	Database Type
COBDIR	Location of COBOL runtime *(COBOL based products only)*
COBJVM	Name of JVM for COBOL integration *(COBOL based products only).* NEW
COBMODE	Execution mode of COBOL runtime (32 or 64 bit) *(COBOL based products only)* NEW
CUSTCOBDIR	Location of custom COBOL installation (if used) *(COBOL based products only)*
ENVFILE	Location and name of environment configuration file
HIBERNATE_JAR_DIR	Location of Hibernate Java library
JAVA_HOME	Location of JDK
ONLINEBILLINI	Location of DOC1 configuration files *(DOC1 supported platforms only)*
ONLINEDOCINI	Location of DOC1 configuration files *(DOC1 supported platforms only)*
OPSYS	Operating System
SPLADMIN	Administration user ID
SPLADMINGROUP	Administration group
SPLAPP	Name of root Web application

Variable	Usage
SPLBUILD	Location of COBOL build directory *(COBOL based products only)*
SPLCOBCPY	Location of COBOL copy code libraries *(COBOL based products only)*
SPLCOMP	Name of COBOL compiler vendor *(COBOL based products only)*
SPLEBASE	Location of software for environment
SPLENVIRON	Name of environment
SPLOUTPUT	Location of output for environment
SPLRUN	Location of runtime for environment
SPLSDKROOT	Location of SDK (Development environment only)
SPLSOURCE	Location of COBOL source *(COBOL based products only)*
SPLSYSTEMLOGS	Location of product specific logs
SPLVERSION	Version identifier of product
SPLVERSIONFILE	Location of version file
SPLWEB	Location of installed Web applications
SPLWAS	Web application Server type
WEB_ISEXPANDED	Whether Web application is expanded or not (not = WAR/EAR files)
WEB_SERVER_HOME	Location of Web Application Server software NEW
WL_HOME	Location of Oracle WebLogic installation (WebLogic supported platforms only)

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** is 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 locations for execution.*

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 J2EE Web Application Server instance (i.e. as per a *local install*) 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>

Sample log entries:

```
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 **splsys**.
- 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]*** *utility for the environment by manually changing to the directory where the* ***splenviron[.sh]*** *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) : Vx.x.x
Database Type ............... (SPLDB) : oracle
Oracle_SID ............. (Oracle_SID) : DEMODB
NLS_LANG ................. (NLS_LANG) : AMERICAN_AMERICA.UTF8
Environment Name ....... (SPLENVIRON) : DEMO
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.

splenviron – Set Environment variables

The **splenviron[.sh]** 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]
```

Where:

-e *<environment>*	*<environment>* is the environment id as installed in the **cistab** file.
-c *<command>*	Execute *<command>* after running **splenviron[.sh]**. Command must be enclosed in double quotes (""). 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.

Samples:

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

The **splenviron[.sh]** 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 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]** 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]** utility (for example, in

your logon profile).

- When the **splenviron[.sh]** 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]** 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]** 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]** 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.

In using this override 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]** 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 ([-a]|[-g]) [-i] [-h]
```

Windows:

```
configureEnv.cmd ([-a]|[-g]) [-i] [-h]
```

Where:

blank	Configure basic configuration options
-a	Configure advanced configuration options NEW
-g	Configure all configuration options (basic and advanced). NEW
-h	Show usage.
-i	Configure Installation options (used for initial installation) NEW

Samples:

```
$ configureEnv.sh -i

************************************************************
Environment Installation Options:
************************************************************
1. Environment Mount Point:
2. Log Files Mount Point:
3. Environment Name:
4. Database Type:
5. Web Application Server Type:
6. Install only Web Component:           false

$ configureEnv.sh -a

************************************************************
Advanced Environment Configuration <ENV_NAME>
************************************************************
 1. General Web Server Environment Settings
      This environment will be used for development:          false
      Preload All Pages on startup:                           false
      Maximum age of a cache entry for text (s):              28800
      Maximum age of a cache entry for images (s):            28800
      To switch to basic Login Page enter: BASIC:             FORM
      Preload All Pages on startup:                           false
      Maximum age of a cache entry for text  :                28800
      Maximum age of a cache entry for images  :              28800
      Interval (s) to check recompiling of JSPs:              43200
```

```
      Authentication Login Page Type           :             FORM

 2. Environment Batch Configuration
        Online JVM Batch Server Enabled:                     false
        Online JVM Batch Threads Number:                     5
        Online JVM Batch Scheduler Daemon:                   false

        * Owned by Top Product (Default False: Release Cobol Thread
Memory):

 3. Environment Memory Configuration
       JVM Child Memory Allocation:
       Weblogic Memory Settings:
        < Min >
        < Max >
        < Max Perm Size >
       ANT Memory Settings:
         < Min >
         < Max >
        Thread Pool Worker Memory Settings:
         < Min >
         < Max >
         < Max Perm Size >
4.   JMX Configuration Setting:
       JMX Port Number:

$ configureEnv.sh

************************************************************
Environment Configuration <ENV_NAME>
************************************************************
1. Environment Description
      Environment Description:

2. Business Application Server Configuration
      Business Server Host:
      Weblogic System User ID:
      Weblogic System Password:
      MPL Admin Port:

3. Web Application Server Configuration
      Web Server Host:
      Web Server Port Number:
      Weblogic SSL Port Number:
```

```
      Weblogic System User ID:
      Weblogic System Password:
      Application Viewer Module:                                    true
      Application Admin User ID :
      Application Admin Password:
      Exploded directory (true) or Archive format (false):    false

4. Database Configuration
      Database User ID:
      Database Password:
      Oracle Database Name:
      Host name of database server:
      Port name for database connection:                            1521
      Oracle Client Character Set NLS_LANG:
AMERICAN_AMERICA.UTF8
      Database JDBC Connection:

P. Write Configuration File
X. Exit (without save)

Each item in the above list should be configured for a successful install.
Choose option to configure or (P) to process or confirm configuration:
```

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

spl – Start/Stop Environment

*Note: The **splenviron[.sh]** utility must be executed before this utility can be used. See splenviron – Set Environment variables for details.*

The **spl[.sh]** 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 in sections of this document.

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] [-wsmba] [-q] <action>
```

Windows:

```
spl.cmd [-h] [-wsmba] [-q] <action>
```

Where:

`-h`	Show usage.
`blank`	Perform *`<action>`* on Web application server/Business Application only
`-w`	Perform *`<action>`* on Web application server only
`-s`	Perform *`<action>`* on Business application server only NEW
`-b`	Perform *`<action>`* on batch component only. DEFAULT threadpool only. NEW BATCH
`-a`	Perform *`<action>`* on all components NEW
`-q`	Quiet Mode – Non-critical output goes to log file only
`-m`	Perform *`<action>`* on Multi-Purpose Listener (MPL) only (if used)
`<action>`	`start` – start the component/environment `stop` – stop the component/environment `check` – Check the status of the environment NEW

When executed the script returns the following return codes:

Return Code ($?)	Comments
0	Command executed successfully
1	Command executed unsuccessfully

Note: The command may issue other commands that need to be tracked separately depending on the platform. For Example

Action	Linux/Unix Command	Windows Command
Start Application Server	`spl.sh start`	`spl start`
Stop Application Server	`spl.sh stop`	`spl stop`
Start all components	`spl.sh -a start`	`spl -a start`
To start the MPL only	`spl.sh -m start`	`spl -m start`
Stop DEFAULT threadpool	`spl.sh -b stop`	`spl -b stop`
Start Business Application Server	`spl.sh -s start`	`spl -s start`
Stop Web Application Server	`spl.sh -w start`	`spl -w start`

genappvieweritems – generate AppViewer

Note: The `splenviron[.sh]` utility must be executed before this utility can be used. See splenviron – Set Environment variables for details.

If the environment is used for reference or development then it may be necessary to regenerate the **appViewer** component 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 [-j] <job> [-Dshv]
```

Windows:

```
genappvieweritems.cmd [-j] <job> [-Dshv]
```

Where:

-h	Show usage.
blank	Execute all extract jobs
-v	Display Version
-j *<job>* NEW	Execute specific <job> from the following list: • **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
-s	Silent Mode (logs only)
-D	Debug Mode enabled (development use only).

Samples:

```
$ genappvieweritems.sh
…
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 WAR/EAR files, the **genappvieweritems** utility will automatically rebuild the WAR/EAR 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 option of initialSetup – Maintain Configuration Settings command must be executed to deploy rebuild the WAR file and redeploy the application.

initialSetup – Maintain Configuration Settings

*Note: The **initialsetup[.sh]** script replaces the **gen*[.sh]** script provided with previous releases of the Oracle Utilities Application Framework.* NEW

*Note: The **splenviron[.sh]** utility must be executed before this utility can be used. See splenviron – Set Environment variables for details.*

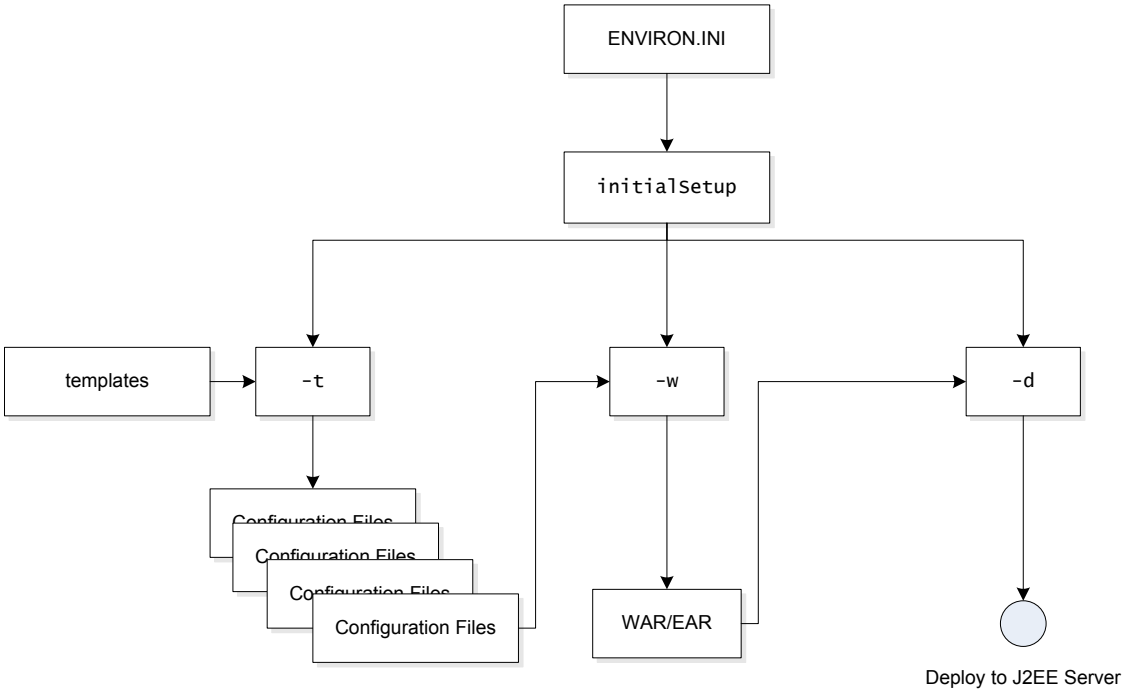
Warning: This command will reset all configuration files to template settings. Any direct customization to configuration files will be lost. Backup configuration files prior to running this script.

During the installation and configuration 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 settings and using a set of provided templates (located in the etc directory), builds the necessary configuration files for the product components.

This utility has three functions:

- Build/Rebuild the configuration files from templates.
- Build/Rebuild the WAR/EAR files used by the product.
- Deploy the WAR/EAR files to the J2EE Web Application Server (optional)

This concept is shown in the diagram below:



While this utility is used at installation time and configuration to reflect configuration settings in the product, 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.

Command Usage:

Linux/Unix:

```
initialSetup.sh [-h] [-t] [-w] [-d] [-v]
```

Windows:

```
initialSetup.cmd  [-h] [-t] [-w] [-d] [-v]
```

Where:

blank	Process Templates, Build WAR/EAR files and Deploy to J2EE Web Application Server in one process.
-h	Show usage.
-t	Process Templates only NEW
-w	Build WAR/EAR files only NEW
-d	Deploy WAR/EAR files only NEW
-v	Display Version

Examples:

```
$ initialSetup.sh
100207.02:37:33 <info> Template generation step.
100207.02:37:43 <info> FW template generation step.
100207.02:37:43 <info> Create war file for SPLApp.war.
100207.02:38:14 <info> Create war file for XAIApp.war.
100207.02:38:26 <info> Create war file for appViewer.war.
100207.02:39:14 <info> Create war file for help.war.
```

```
100207.02:41:11 <info> FINISHED INITIAL INSTALL SETUP at Thu Aug  7
02:41:11 EST 2009

100207.02:41:11 <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.

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.

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/or Business application server using the administration account for the desired environment (for example, **splsys**).
- Execute the following command to attach to the desired environment:

  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 software the environment 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 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.

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 OWNER=`perl ${_splebase}/bin/getconfvalue.plx -k SPLUSER`

      # 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 - $OWNER -c "${_startcmd}" ;;
           "stop")  su - $OWNER -c "${_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.

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:

- Log on to the server containing the Web application server and/or Business application server using the administration account for the desired environment (for example, **splsys**).
- Execute the following command to attach to the desired environment:

  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 software the environment and ***<environment>*** is the name of the environment to stop.

- Stop 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[.sh] 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.

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 is 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 Oracle Utilities Application Framework Performance Troubleshooting Guides KB Id: **560382.1** on My Oracle Support.

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.

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 pick up 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

Monitoring The Desktop

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:

- **Desktop vendor 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`

Where:

<host>	Web Application Server hostname
<port>	Port allocated to product installation
<server>	Context for the product at installation time

Note: For the user to have debug access their userid must have "Change" access to service ***F1DEBUG****.*
NEW

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. It is recommended to 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 uses ***spl_web.log*** *and* ***spl_service.log*** *but* ***spl_service.log*** *or may not appear depending on the installation type, therefore the name* ***spl*.log*** *is used.*

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

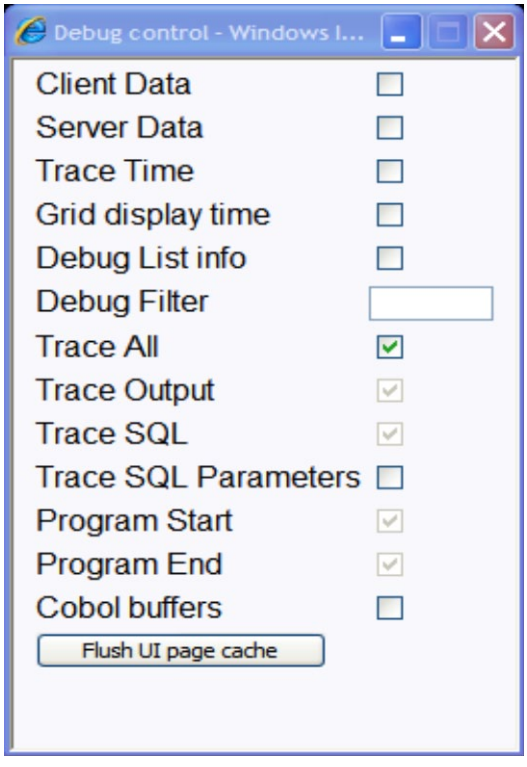
Most tracing in non-development uses *Trace All* unless otherwise instructed by Oracle Support. All debug information is written to the **spl*.log** files.

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 as shown below:

  - **Start Debug** – Start the logging of the transaction. NEW
  - **Stop Debug** – Stop the logging process NEW
  - **Show Trace** – Show trace information (Configuration based objects only) NEW
  - **Clear Trace** – Clear Trace Information NEW
  - **Show User Log** – Show debug information for the user (line limit is configurable).
  - **Validate on DB** – SDK Use only
  - **Global Debug** – Set debug level.
- Select *Global Debug* to specify the level of debug information. 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. A sample of the Debug Control dialog is shown below:



- Navigate to the transaction that you wish to trace as a user would normally operate. Press "Start Debug" to initiate debug.
- 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* or press *Stop Debug* so that debugging is disabled. This will stop debug code writing to the writing to the log. If you select *Show User Log* the log lines output by the debug facility are displayed (*up to the line limit specified*). This will only show lines applicable to the Current User only.

Note: If the userid is shared across multiple physical users then the information may contain debug information from multiple sessions.

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 6 has a predefined set of MBeans that can be enabled automatically.
- **Web application server console** – All Web Application Servers 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 6 (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.

Refer to the Oracle Utilities Application Framework Performance Troubleshooting Guides KB Id: **560382.1** on My Oracle Support for details of monitoring aspects of the product.

JMX Based Monitoring NEW

With the advent of Java Management Extensions (JMX) technology into base java, it is possible to use the technology to monitor and manage java infrastructure from a JSR160 compliant JMX compliant console (or JMX browser). Whilst the J2EE components of the product can use basic JMX statistics such as Memory usage, Threads, Class information and VM summary information, there are application specific JMX classes added to the product to allow greater levels of information to be display and additional operations.

The Oracle Utilities Application Framework has implemented a set of product specific JMX classes on the Web Application Server and Business Application Server tiers of the architecture to allow the following:

- Management of the cache of the Web Application Server. See Server Cache Management for more details of this cache.
- Collection of JVM information and performance statistics for memory, thread usage and operating system level information. Most of these are extensions of java.lang.management classes.
- Collection of service based performance information for SLA tracking on the Business Application Server.

To use this facility the facility must be configured and enabled to allow the collection of the relevant information. This can be done at installation time by using the following configuration settings:

Configuration Setting	Deployment details
`WEB_JMX_RMI_PORT_PERFORMANCE`	Port Number used for JMX based management for Web Application Server.
`ouaf.jmx.splwg.base.support.management.mbean.JVMInfo`	Globally enable or disable `JVMInfo` Mbean (setting in spl.properties). Default is **enabled**.
`ouaf.jmx.com.splwg.base.web.mbeans.FlushBean`	Globally enable or disable `FlushBean` Mbean (setting in spl.properties). Default is **enabled**.
`BSN_JMX_RMI_PORT_PERFORMANCE`	Port Number used for JMX based

Configuration Setting	Deployment details
	management for Business Application Server.
ouaf.jmx.com.splwg.ejb.service.management.PerformanceStatistics	Globally enable or disable **PerformanceStatistics** Mbean (setting in spl.properties). Default is **enabled**
BSN_JMX_SYSUSER	Default JMX Userid for both Web Application Server and Business Application Server
BSN_JMX_SYSPASS	Default JMX Password for both Web Application Server and Business Application Server

These settings are registered in the ENVIRON.INI for setting in the relevant configuration files. It is important that the values used for these port numbers are unique across all environments within a particular machine. The security used for these ports are defined as outlined in the JMX Security section of this document.

Web Application Server JMX Reference

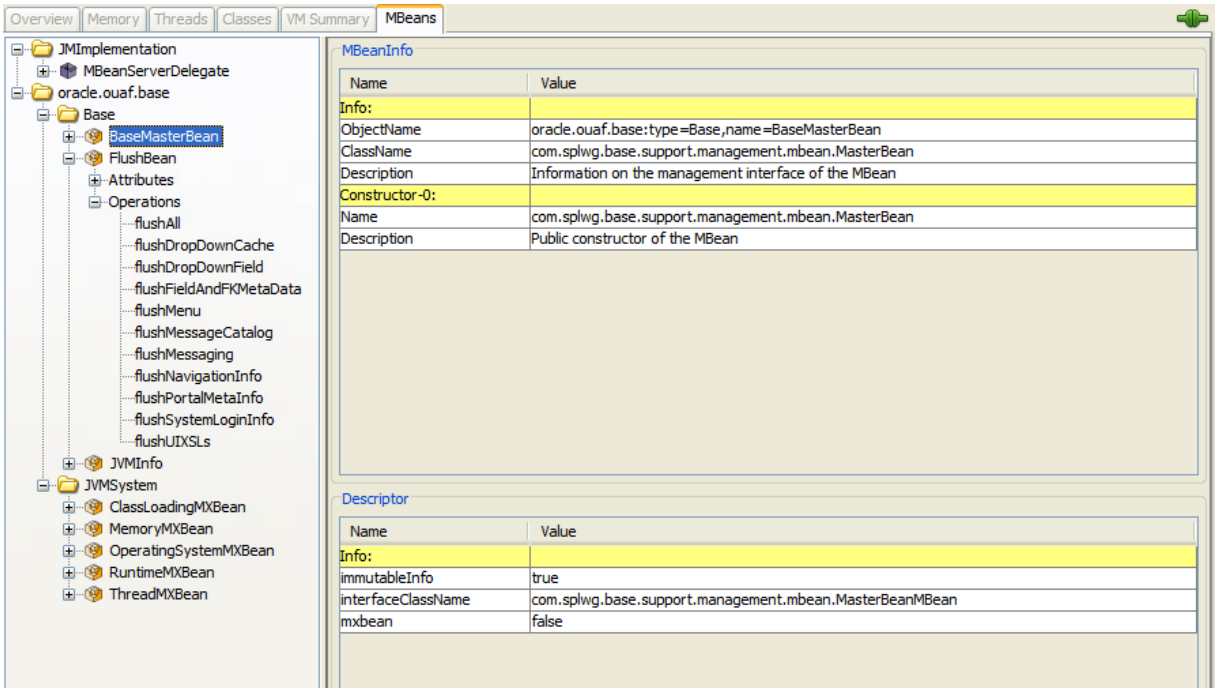
Once configured a JMX client (e.g. **jconsole**) can be used to connect to the JMX information using the following Remote Connection string:

`service:jmx:rmi:///jndi/rmi://<host>:<jmx_port>/oracle/ouaf/webAppConnector`

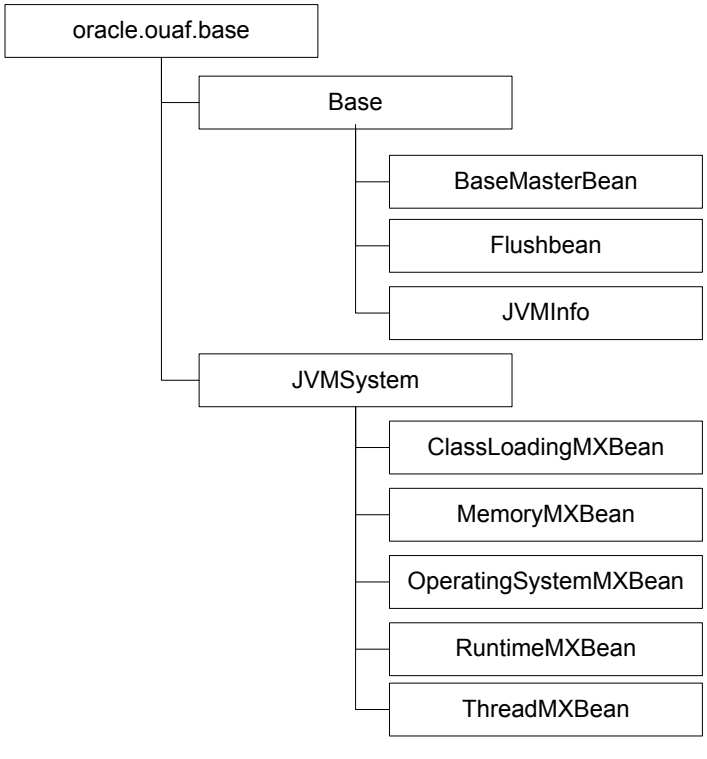
Where:

<host>	The Web Application Server host name
<jmx_port>	The JMX Port specified using **WEB_JMX_RMI_PORT_PERFORMANCE** from the ENVIRON.INI configuration file.

The credentials provided to the JMX console are as configured in JMX Security. Upon successful connection to the JMX port and host with the correct credentials provides access to the Mbean information. The figure below illustrates the successful connection to the JMX Mbeans using **jconsole** (*as an example*):



The structure of the Mbean is shown by the figure below:



The following table summarizes the Mbean attributes and operations for the Web Application Server:

Mbean	Arguments	Usage
BaseMasterBean Attributes		
`NumberOfMbeans`	None	Returns number of active Mbeans
`MBeanList`	None	Returns an array with the list of Mbeans defined to this Master

Mbean	Arguments	Usage
		Mbean
BaseMasterBean Operations		
disableMbean	Mbean Name	Disables Mbean with designated name
enableMbean	Mbean Name	Enables Mbean with designated name
disableMbean	None	Returns a list with the list of Mbeans defined to this BaseMasterBean. Names can be programmatically used to supply parameters to disableMbean and enableMbean.
enableJVMSystemBeans	None	Enables base JVM system Mbeans
disableJVMSystemBeans	None	disables base JVM system Mbeans
FlushBean Attributes		
VersionInfo	None	Returns string of base version number of Flush Mbean.
CompleteClassPath	None	Returns classpath name of Flushbean Mbean.
FlushBean Operations (Refer to Server Cache for details of this cache).		
flushAll	None	Reset all elements in online data cache
flushDropDownCache	None	Resets cached elements of the online drop down lists in online data cache
flushDropDownField	None	Resets drop down fields in online data cache (*Development use only*)
flushFieldAndFKMetaData	None	Resets Field and Foreign Key Meta Data in online data cache
flushMenu	None	Reset Menu items in online data cache
flushMessageCatalog	None	Reset field labels in online data cache
flushMessaging	None	Reset messages in online data cache
flushNavigationInfo	None	Reset navigation information in online data cache
flushPortalMetaInfo	None	Reset portal and zone information in online data cache
flushSystemLoginInfo		Reset security information in online

Mbean	Arguments	Usage
		data cache
flushUIXSLs	None	Reset user interface style sheets in online data cache
JVMInfo Attributes		
CompleteClassPath	None	Displays the class path of the JVMInfo mbean
JVMInfo Operations		
classPath	None	Returns the full classpath used by the online JVM
systemSettings	None	Returns the attributes of the JVM for debugging and support purposes.
ClassLoadingMXBean Attributes		
LoadedClassCount	None	Returns the number of classes that are currently loaded in the JVM
TotalLoadedClassCount	None	Returns the total number of classes that have been loaded since the JVM was last started.
UnloadedClassCount	None	Returns the total number of classes unloaded since the Java virtual machine has started execution.
Verbose	None	Enables or disables the verbose output for the class loading system. Default is **false** (*disabled*)
MemoryMXBean Attributes		
HeapMemoryUsage	None	Returns the current memory usage of the heap that is used for object allocation. Initial, Committed, Maximum and Used memory statistics are provided for Heap memory
NonHeapMemoryUsage	None	Returns the current memory usage of non-heap memory that is used by the JVM. Initial, Committed, Maximum and Used memory statistics are provided for Non-Heap memory
ObjectPendingFinalization	None	Returns the approximate number of objects for which finalization is pending (used for diagnosing

Mbean	Arguments	Usage
		memory leaks).
Verbose	None	Enables or disables the verbose output for the memory system. Default is **false** (*disabled*)
MemoryMXBean Operations		
gc	None	Initiate garbage collection
MemoryMXBean Notifications		
javax.management.Notification	None	Used for low memory notifications. Notification Types supported: (java.management.memory.threshold .exceeded, java.management.memory .collection.threshold.exceeded)
OperatingSystemMXBean		
MaxFileDescriptorCount	None	Returns the File Descriptor Maximum Limit in force on the JVM
OpenFileDescriptorCount	None	Returns the number of Open File Descriptors currently used by JVM
CommittedVirtualMemorySize	None	Returns the amount of committed virtual memory (that is, the amount of virtual memory guaranteed to be available to the running process).
FreePhysicalmemorySize	None	Returns the total amount of free physical memory
FreeSwapSpaceSize	None	Returns the total amount of free swap space
ProcessCpuTime	None	Returns the amount of process CPU time consumed by the JVM
TotalPhysicalMemorySize	None	Returns the total amount of physical memory
TotalSwapSpaceSize	None	Returns the total amount of swap space
Name	None	Returns the operating system name
Version	None	Returns the version of the operating system
Arch	None	Returns the operating system architecture

Mbean	Arguments	Usage
AvailableProcessors	None	Returns the number of available processors to the JVM
SystemLoadAverage	None	Returns the system load average for the last minute.
RuntimeMXBean Attributes		
Name	None	Returns the name representing the running JVM. The returned name string can be any arbitrary string and a JVM implementation can choose to embed platform-specific useful information in the returned name string. Each running virtual machine could have a different name.
ClassPath	None	Returns the Java class path that is used by the system class loader to search for class files.
Starttime	None	Returns the start time of the Java virtual machine in milliseconds. This method returns the approximate time when the JVM started.
ManagementSpecVersion	None	Returns the version of the specification for the management interface implemented by the running JVM
VmName	None	Returns the Java virtual machine implementation name
VmVendor	None	Returns the Java virtual machine implementation vendor
VmVersion	None	Returns the Java virtual machine implementation version
SpecName	None	Returns the Java virtual machine specification name
SpecVendor	None	Returns the Java virtual machine specification vendor
SpecVersion	None	Returns the Java virtual machine specification version
LibraryPath	None	Returns the Java library path
BootClassPath	None	Returns the boot class path that is used by the bootstrap class loader to

Mbean	Arguments	Usage
		search for class files
Uptime	None	Returns the uptime of the Java virtual machine in milliseconds
BootClassPathSupported	None	Tests if the JVM supports the boot class path mechanism used by the bootstrap class loader to search for class files. Returns *false* if not supported; *true* if supported
InputArguments	None	Returns the input arguments passed to the JVM which does not include the arguments to the main method. This method returns an empty list if there is no input argument to the JVM. Typically, not all command-line options to the '**java**' command are passed to the Java virtual machine. Thus, the returned input arguments may not include all command-line options
SystemProperties	None	Returns a map of names and values of all system properties
ThreadMXBean Attributes		
ThreadCount	None	Returns the current number of live threads including both daemon and non-daemon threads
PeakThreadCount	None	Returns the peak live thread count since the JVM started or peak was reset
TotalStartedThreadCount	None	Returns the total number of threads created and also started since the JVM started
DaemonThreadCount	None	Returns the current number of live daemon threads
ThreadContentionMonitoringSupported	None	Tests if the JVM supports thread contention monitoring. Returns *false* if not supported; *true* if supported
ThreadContentionMonitoringEnabled	None	Enables or disables thread contention monitoring. Set to *false* to disable; *true* to enable.
CurrentThreadCpuTime	None	Returns the total CPU time for the

Mbean	Arguments	Usage
		current thread in nanoseconds. The returned value is of nanoseconds precision but not necessarily nanoseconds accuracy. If the implementation distinguishes between user mode time and system mode time, the returned CPU time is the amount of time that the current thread has executed in user mode or system mode
CurrentThreadUserTime	None	Returns the CPU time that the current thread has executed in user mode in nanoseconds. The returned value is of nanoseconds precision but not necessarily nanoseconds accuracy.
ThreadCpuTimeSupported	None	Tests if the JVM supports CPU time measurement for the current thread. Returns *false* if not supported; *true* if supported
ThreadCpuTimeEnabled	None	Enables or disables thread CPU time measurement. The default is platform dependent. Set to *false* to disable; *true* to enable.
CurrentThreadCpuTimeSupported	None	Tests if the Java virtual machine supports CPU time measurement for the current thread. Returns *false* if not supported; *true* if supported
ObjectMonitorUsageSupported	None	Tests if the Java virtual machine supports monitoring of object monitor usage. Returns *false* if not supported; *true* if supported
SynchronizerUsageSupported	None	Tests if the JVM supports monitoring of ownable synchronizer usage. Returns *false* if not supported; *true* if supported
AllThreadIds	None	Returns all live thread IDs. Some threads included in the returned array may have been terminated when this method returns
ThreadMXBean Operations		

Mbean	Arguments	Usage
dumpAllThreads	Locked Monitors, Locked Synchronizers	Returns the thread info for all live threads with stack trace and synchronization information. Some threads included in the returned array may have been terminated when this method returns • Locked Monitors - if *true*, dump all locked monitors • Locked Synchronizers - if *true*, dump all locked ownable synchronizers
findDeadlockedThreads	None	Finds cycles of threads that are in deadlock waiting to acquire object monitors or ownable synchronizers. Threads are deadlocked in a cycle waiting for a lock of these two types if each thread owns one lock while trying to acquire another lock already held by another thread in the cycle
getThreadCpuTime	Thread Id	Returns the total CPU time for a thread of the specified ID in nanoseconds. The returned value is of nanoseconds precision but not necessarily nanoseconds accuracy. If the implementation distinguishes between user mode time and system mode time, the returned CPU time is the amount of time that the thread has executed in user mode or system mode
getThreadInfo	Thread Id	Returns the thread info for a thread of the specified id with no stack trace.
getThreadInfo	Array of Thread Ids	Returns the thread info for each thread whose ID is in the input array ids with no stack trace.
getThreadInfo	Thread Id, maxDepth	Returns thread information for a thread of the specified id, with stack trace of a specified number of stack trace elements. The *maxDepth* parameter indicates the maximum

Mbean	Arguments	Usage
		number of *StackTraceElements* to be retrieved from the stack trace. This method does not obtain the locked monitors and locked synchronizers of the thread
`getThreadInfo`	Array of Thread Ids, maxDepth	Returns the thread information for each thread whose ID is in the input array ids, with stack trace of a specified number of stack trace elements. The *maxDepth* parameter indicates the maximum number of *StackTraceElements* to be retrieved from the stack trace. This method does not obtain the locked monitors and locked synchronizers of the threads
`getThreadInfo`	Array of Thread Ids, locked Monitors, locked Synchronizors	Returns the thread info for each thread whose ID is in the input array ids, with stack trace and synchronization information. This operation obtains a snapshot of the thread information for each thread including: • the entire stack trace, • the object monitors currently locked by the thread if *lockedMonitors* is true, and • the ownable synchronizers currently locked by the thread if *lockedSynchronizers* is true
`getThreadUserTime`	Thread Id	Returns the CPU time that a thread of the specified ID has executed in user mode in nanoseconds
`resetPeakThreadCount`	None	Resets the peak thread count to the current number of live threads

Business Application Server JMX Reference

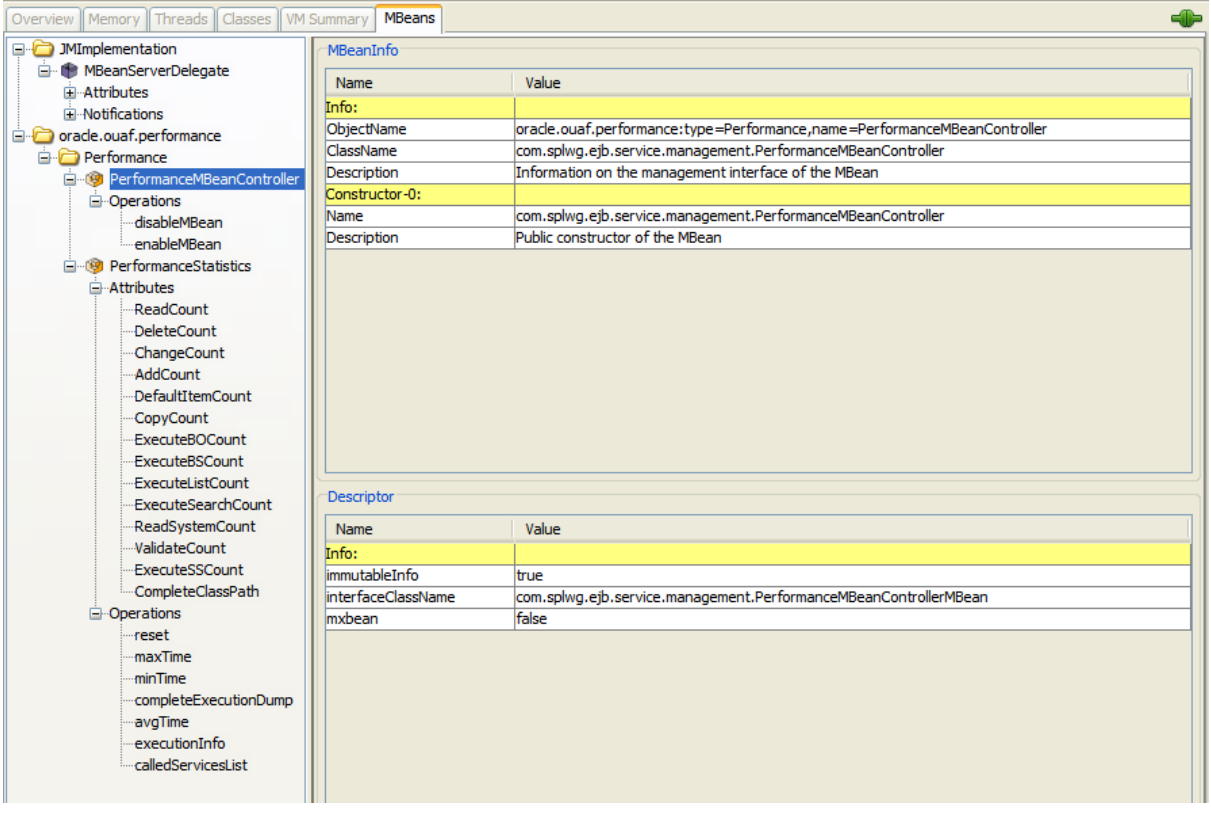
Once configured a JMX client (e.g. `jconsole`) can be used to connect to the JMX information using the following Remote Connection string:

`service:jmx:rmi:///jndi/rmi://<host>:<jmx_port>/oracle/ouaf/ejbAppConnector`

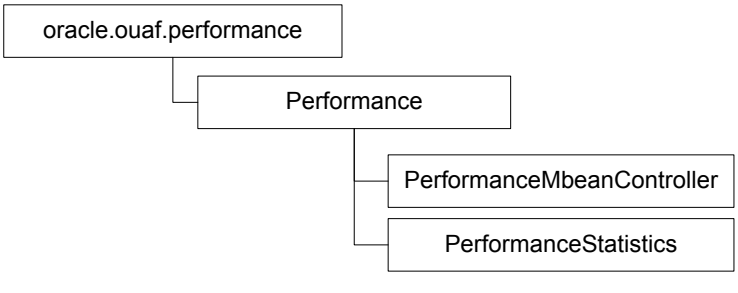
Where:

<host>	The Business Application Server host name
<jmx_port>	The JMX Port specified using **BSN_JMX_RMI_PORT_PERFORMANCE** from the ENVIRON.INI configuration file.

The credentials provided to the JMX console are as configured in JMX Security. Upon successful connection to the JMX port and host with the correct credentials provides access to the Mbean information. The figure below illustrates the successful connection to the JMX Mbeans using **jconsole** (*as an example*):



The structure of the Mbean is shown by the figure below:



The following table outlines the Mbean attributes and operations for the Business Application Server:

Mbean	Arguments	Usage
PerformanceMBeanController Operations		
disableMbean	None	Disable PerformanceStatistics Mbean

Mbean	Arguments	Usage
enableMbean	None	Enable PerformanceStatistics Mbean
PerformanceStatistics Attributes		
ReadCount	None	Returns number of executed *read object* calls since last reset or last time collection enabled
DeleteCount	None	Returns number of executed *delete object* calls since last reset or last time collection enabled
ChangeCount	None	Returns number of executed *change object* calls since last reset or last time collection enabled
AddCount	None	Returns number of executed *add object* calls since last reset or last time collection enabled
DefaultItemCount	None	Returns number of executed calls to *default the object values* since last reset or last time collection enabled
ExecuteBOCount	None	Returns number of calls to *Business Objects* since last reset or last time collection enabled
ExecuteBSCount	None	Returns number of calls to *Business Services* since last reset or last time collection enabled
ExecuteListCount	None	Returns number of calls to *List based services* since last reset or last time collection enabled
ExecuteSearchCount	None	Returns number of calls to *search based services* since last reset or last time collection enabled
ReadSystemCount	None	Returns number of calls to *Oracle Utilities Application Framework system Objects* since last reset or last time collection enabled
ValidateCount	None	Returns number of calls to *Validate objects* since last reset or last time collection enabled
ExecuteSSCount	None	Returns number of calls to *Service Scripts* since last reset or last time collection enabled
CompleteClassPath	None	Returns the class path used for the Mbeans
PerformanceStatistics Operations		
reset	None	Resets statistical values. See Resetting Statistics for more advice on this operation.
maxTime	Service Name	Returns maximum (worst case) time, in ms, for the designated service since the last reset or last time collection enabled.
minTime	Service Name	Returns minimum (best case) time, in ms, for the designated service since the last reset or last

Mbean	Arguments	Usage
		time collection enabled.
`completeExecutionDump`	None	Returns complete statistics for all services executed since the last reset or last time collection enabled. See Execution Dump section for details of format.
`avgTime`	Service Name	Returns average time, in ms, for the designated service since the last reset or last time collection enabled.
`executionInfo`	Service Name	Returns complete statistics for the designated service executed since the last reset or last time collection enabled. See Execution Dump Format section for details of format.
`calledServices`	None	Returns list of services and service types since the last reset or last time collection enabled. See Service Lists for details of format.

Note: The times quoted in the statistics only record times experienced from the Business Application Server down to the data and back. They do not include network time to the Web Application Server, any time spent by the Web Application Server, network time to the browser client or browser rendering times. The Business Application Server time represents the typical majority of the time spent in a transaction.

JMX Security

By default, when JMX is enabled for either the Web Application Server and Business Application Server then a default JMX configuration using simple security is implemented as outlined in http://java.sun.com/javase/6/docs/technotes/guides/management/agent.html.

The simple security system consists of two files that control the access permissions and passwords specified by default for the installation:

Configuration Setting	Location of file	Template
Password File	`scripts/ouaf.jmx.password.file`	`ouaf.jmx.password.file.template`
Access Control File	`scripts/ouaf.jmx.access.file`	`ouaf.jmx.access.file.template`

These files are built by the initialSetup utility using the templates indicated. Refer to the templates or generated files for valid values. The format of these files is dictated by http://java.sun.com/javase/6/docs/technotes/guides/management/agent.html#gdeup.

Note: By default, the passwords stored in these files are in encrypted text. Alternative security schemes are allows as documented in the link above. This will require a custom templates and changes to specific files to implement.

Execution Dump Format

In previous versions (V1.x) of the Oracle Utilities Application Framework based products, it was possible to extract performance information from the Business Application Server using a logging based method using the Oracle Tuxedo `txrpt` utility. This facility was useful in tracking performance of individual services over time to detect non-compliance against Service Level Agreement targets. With the advent of later versions of the Oracle Utilities Application Framework, the need for Oracle Tuxedo was removed but there was a need for performance information to be collated.

In the latest version of the Oracle Utilities Application Framework, it is possible to track performance information using JMX to process externally to check performance and check compliance against Service Level Agreements.

To extract the information from the product the following needs to be done:

- Use a JMX browser (or JMX console) product to connect to the Business Application Server JMX port using the appropriate credentials.
- Invoke the `completeExecutionDump` operation from the `PerformanceStatistics` Mbean. This is will return a Comma separated values, with field names in the header record, containing the performance data which can be transferred to the clipboard (or whatever format supported by the JMX client). The format of the CSV is shown in the table below:

Column	Comment
ServiceName	Name of Service
ServiceType	Type Of Service or Action (see Service Lists for valid values)
MinTime	Minimum Service Time, in ms, since last reset
MaxTime	Minimum Service Time, in ms, since last reset
Avg Time	Average Service Time, in ms, since last reset
# of Calls	Number of Calls to Service since last reset
Latest Time	The service time of the latest call, in ms
Latest Date	The date of the latest service call (in format: *YYYY-MM-DD::hh-mm-ss-sss*)
Latest User	The userid of the user who issued the latest call

- (Optionally) Invoke the `reset` operation from the `PerformanceStatistics` Mbean to reset the statistics for the next collection period. Refer to Resetting Statistics for a discussion of this task.

This information can then be post processed in an appropriate analysis tool to determine appropriate actions.

Note: The statistics are active as long the Mbean is enabled or the system is active. Shutting down the Business Application Server with collection of the data may cause data loss for the statistics.

Service Lists

The JMX Performance Mbeans collect information about application services that have been executed during the collection period. This information can be obtained using the **calledServices** operation which returns a list of called services and their valid actions (summarized actions that have been called) in the format:

<servicename> [<valid action>]

Where

<servicename>	Name of Service
<valid actions>	List of valid actions recorded for the service. The table below lists the valid values

Valid Action	Comment
ADD	Service is attempting adding a new instance of an object to the system. For example, adding a to do record.
CHANGE	Service is attempting changes to an existing object in the system.
DEFAULT_ITEM	Service is resetting its values to defaults. For example, by pressing the *Clear* button on the product UI toolbar
DELETE	Service is attempting to delete an existing object
EXECUTE_BO	Service is a business object
EXECUTE_BS	Service is a business service
EXECUTE_LIST	Service is a list based service
EXECUTE_SEARCH	Service is a search
EXECUTE_SS	Service is a service script (including BPA scripts)
READ	Service is attempting to retrieve an object from the system
READ_SYSTEM	Service is a common Oracle Utilities Application Framework based service.
VALIDATE	Service is issuing a validation action

Resetting Statistics

The performance statistics collected represent values since the application was started or when it has been reset. Collection of statistics, without reset, can adversely influence the effectiveness of the statistics over time. It is therefore recommended to reset the statistics on a regular basis (after they are collected for example).

This can be achieved using the **reset** operation from the **PerformanceStatistics**

Mbean to effectively zero or blank out the collection statistics.

For example, if the statistics are to be collected on an hourly basis then the reset should occur after the data collection happens per hour.

Note: Any statistics collected during the actual reset operation will not be reflected in the statistics. This situation should have minimal impact on overall statistics.

Database Connection Monitoring NEW

Note: The facility described in this section only applies to sites using the Oracle database.

By default, the product uses a common database userid for accessing the information from the connection pools used by the product (via UCP or the JNDI based connection pools). While this sufficient for execution of the product, it can complicate monitoring individual connections and troubleshooting database issues with individual users or transactions.

It is now possible to show the actual user (as per the security system) that is using the database connection from the pool during the execution of the individual transaction. The application userid is propagated from the product to the **CLIENT_IDENTIFIER** on individual sessions.

For example, the following database query will return the session ids and the users using then at any time:

```
SELECT sid, client_identifier FROM V$SESSION;
```

This will not affect execution of the query as the common database userid is used to access the appropriate userid but the **CLIENT_IDENTIFIER** can be used by the database to impose additional security or profiling on the application user. In other words, by default, the application users do not have to be defined to the database but the **CLIENT_IDENTIFER** can be used by the database (and related additional security and auditing products) to impose additional functionality.

Configuration

Global Configuration Files

There are a number of configuration files that are global across an environment and also restricted to an environment.

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`

A sample 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.

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

*Note: For Windows environments it is possible to move the file to alternative drive by setting **%SYSTEMDRIVE%** to an alternative drive prior to running any utilities. For example **set SYSTEMDRIVER=D:** places the **cistab** in **d:\spl\etc**.*

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 utility provided in the installation.

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

Location of **ENVIRON.INI** 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>	Name of configuration parameter
<value>	Value of the configuration parameter

For example:

```
…
appViewer=appViewer
DBCONNECTION=jdbc:oracle:thin:@myserver:1521:train
DBDRIVER=oracle.jdbc.driver.OracleDriver
DBNAME=TRAIN
…
```

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

Tier	Blank = all, WEB = Web application server, BAS = Business Application Server, XAI = Web Services Adapter, DB = Database.
Platform	Blank = all, WLS = Oracle WebLogic, WASND = IBM WebSphere ND, WAS = IBM WebSphere, ORA = Oracle Database, MSSQL = Microsoft SQL Server, DB2 = IBM DB2

Parameter	Description	Tier	Platform
ADDITIONAL_RUNTIME_CLASSPATH NEW	Additional Runtime Classpath for Web Application Server (allows custom jars to be added to path)		
ADF_HOME NEW	Location Of ADF software	WEB	
ANT_ADDITIONAL_OPT NEW	Ant Additional Options		
ANT_HOME NEW	Location of Ant software		
ANT_OPT_MAX NEW	Ant Maximum Heap Size		WLS
ANT_OPT_MIN NEW	Ant Minimum Heap Size		WLS
appViewer	Name of appViewer WAR file	WEB	
BATCH_MEMORY_ADDITIONAL_OPT NEW BATCH	Threadpool Worker JVM additional options	BAS	
BATCH_MEMORY_OPT_MAX NEW BATCH	Threadpool Worker Java Maximum Heap Size	BAS	
BATCH_MEMORY_OPT_MAXPERMSIZE NEW BATCH	Threadpool Worker Java Maximum Perm Size	BAS	
BATCH_MEMORY_OPT_MIN NEW BATCH	Threadpool Worker Java Minimum Heap Size	BAS	
BATCH_MODE NEW BATCH	Default Mode of Batch	BAS	
BATCH_RMI_PORT NEW BATCH	RMI Port for Batch	BAS	
BATCHDAEMON BATCH	Inbuilt Batch Daemon enabled	BAS	
BATCHENABLED BATCH	Inbuilt Batch Server enabled	BAS	
BATCHTHREADS BATCH	Maximum # of threads for Inbuilt Batch Server	BAS	
BSN_APP NEW	Business Server Application Name	BAS	
BSN_JMX_RMI_PORT_PERFORMANCE NEW	RMI port for JMX monitoring of Business Server Application	BAS	
BSN_JMX_SYSPASS	Default Password for BSN JMX Monitoring	BAS	

Parameter	Description	Tier	Platform
BSN_JMX_SYSUSER	Default User for BSN JMX Monitoring	BAS	
BSN_JVMCOUNT	Number of Child JVM's	BAS	
BSN_NODENAME NEW	IBM WebSphere Node Name	BAS	WASND
BSN_RMIPORT	RMI Port for Child JVM	BAS	
BSN_SVRNAME	IBM WebSphere Server Name	BAS	WAS WASND
BSN_WASBOOTSTRAPPORT	Bootstrap Port	BAS	WAS WASND
BSN_WLHOST	Business App Server Host	BAS	
BSN_WLS_SVRNAME	Oracle WebLogic Server Name	BAS	WLS
CHILD_JVM_JAVA_HOME NEW	Location of Java SDK home used for COBOL interface	BAS	
CHILD_JVM_PATH NEW	Location of Java Libraries for COBOL	BAS	
CLEANSE_INTERVAL NEW MWM	ORS Registry cleansing interval in seconds	WEB	
CMPDB	Database Type (ORA, MSSQL or DB2)	DB	
COBDIR NEW	COBOL Home Directory	BAS	
COBDIR_INPUT NEW	COBOL Home Directory (may be compiler)	BAS	
COHERENCE_CLUSTER_ADDRESS NEW BATCH	Multicast IP address for CLUSTERED execution mode.	BAS	
COHERENCE_CLUSTER_MODE NEW BATCH	Mode used for Coherence execution (dev is for development and prod is for all other environments)	BAS	
COHERENCE_CLUSTER_NAME NEW BATCH	Batch cluster name for CLUSTERED execution mode.	BAS	
COHERENCE_CLUSTER_PORT NEW BATCH NEW	Multicast port number for CLUSTERED execution	BAS	

Parameter	Description	Tier	Platform
	mode.		
COLLATE	SQL Server Collation Name	DB	MSSQL
CONTEXTFACTORY NEW MWM	JMS Context Factory used for external ORS integration	WEB	
DATABASE_HOME NEW	Location of Database software	DB	
DB_OVERRIDE_CONNECTION NEW	Custom JDBC URL	DB	ORA
DB2_HOME NEW	Location of DB2 Software	DB	DB2
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
DBCONNECTION	JDBC Connection string (generated)	DB	
DBDRIVER	Hibernate DB driver	DB	
DBPASS NEW BATCH	Database password for Database User used for database component.	DB	
DBPASS_GEOCODE_WLS NEW MWM	Geocode Database Userid Password		
DBPASS_WLS	Internal Database Password for Oracle WebLogic connection pool	BAS	WLS
DBPORT	Port Number for Database	DB	
DBSERVER NEW	Database Server	DB	
DBURL_GEOCODE NEW MWM	JDBC URL for the Geocode database		
DBUSER NEW BATCH	Database User used by product for the batch component.	DB	
DBUSER_GEOCODE NEW MWM	Geocode Database Userid		
DBUSER_WLS NEW	Internal Database User for Oracle WebLogic	BAS	WLS

Parameter	Description	Tier	Platform
	connection pool		
DESC	Environment Description		
DIALECT	Hibernate Dialect	DB	
DIRSEP	Directory separator		
DOC1BILLSCRIPT	DOC1 Bill Script location (*if DOC1 installed*)	BAS	
DOC1HOSTDIR	Location of DOC1 installation	BAS	
DOC1SCRIPT	DOC1 Letter Script location (*if DOC1 installed*)	BAS	
ENCODING	Whether encryption is enabled		
FW_VERSION NEW	Oracle Utilities Application Framework version		
FW_VERSION_NUM NEW	Oracle Utilities Application Framework high level version		
GIS NEW	GIS Running on Same Sever	WEB	
GIS_URL NEW	GIS Service URL	WEB	
GIS_WLSYSPASS NEW	GIS WebLogic System Password	WEB	WLS
GIS_WLSYSUSER NEW	GIS WebLogic System User Id	WEB	WLS
help	Name of online help WAR file	WEB	
HIBERNATE_JAR_DIR NEW	Location of Hibernate JAR files	BAS	
HIGHVALUE	Language specific highvalues	BAS	
HTTP_ISENABLED NEW MWM	Enable HTTP (non secure) for MCP	WEB	
IPCSTARTPORT NEW MWM	Starting Port for ORS	WEB	
JAVA_HOME NEW	Location of Java SDK		
JAVAENCODING	Java Language Encoding	BAS	

Parameter	Description	Tier	Platform
JDBC_NAME NEW	Name of JDBC Web Application Server pools defined within Web Application Server. This is used for the online and Web Application Server only.	BAS	
JNDI_GEOCODE NEW MWM	JNDI name for the Geocode datasource		
JVM_ADDITIONAL_OPT	Child JVM additional Options. These options must be valid for the java vendor and version used.	BAS	
JVMCOMMAND	Generated java command for Child JVM (if COBOL used)	BAS	
JVMMEMORYARG	Child JVM Memory Allocation (if COBOL used). The format is the JVM command line options	BAS	
LD_LIBRARY_PATH NEW	Library Path for Linux and Solaris		
LIBPATH NEW	Library Path for AIX		
MAPDIR NEW MWM	Location of Map files used for ORS	WEB	
MAPVIEWER_EAR NEW MWM	Location of Mapviewer ear file	WEB	
MAPVIEWER_ISLOCAL NEW MWM	Deploy Mapviewer locally on this instance	WEB	
MAXPROCESSINGTIME NEW MWM	Maximum Processing time (in seconds) for transaction in ORS.	WEB	
MINREQUESTS NEW MWM	Minimum Requests to be spawned at startup time	WEB	
MOBILITY_APP_ONLY NEW MWM	Deploy only mobility web application	WEB	
MODULES	Names of Modules installed		

Parameter	Description	Tier	Platform
MPL_DBPASS	Password for database user used for MPL	XAI	
MPL_DBUSER	Database user used for MPL	XAI	
MPLADMINPORT	MPL Administration Port	XAI	
MPLSTART	MPL Automatic Start (Y/N)	XAI	
NLS_LANG	NLS Language setting	DB	ORA
NODEID NEW MWM	Scheduler Node Identifier	WEB	
ONSCONFIG NEW	ONS Configuration (Oracle RAC FCF only)	DB	ORA
OPSYS NEW	Operating System		
ORACLE_CLIENT_HOME NEW	Home directory of Oracle Client. The product uses the perl located under this directory.		ORA
ORACLE_HOME NEW	Location of Oracle Software	DB	ORA
OWNERUSER	Owner of Database Schema	DB	
PERL5LIB	Location of custom PERL libraries		
RAC_INITIALLIMIT NEW	RAC Initial Limit (Oracle RAC FCF only)	DB	ORA
RAC_MAXLIMIT NEW	RAC Maximum Limit (Oracle RAC FCF only)	DB	ORA
RAC_MINLIMIT NEW	RAC Minimum Limit (Oracle RAC FCF only)	DB	ORA
RJVM	Whether Child JVM is considered remote (if COBOL used)	BAS	
SHLIB_PATH NEW	Library Path for HP-UX		
SPLADMIN	OS Administration userid		
SPLADMINGROUP	OS Administration group		
SPLApp	Name of product WAR file	WEB	
SPLDIR NEW	Location of Environment		

Parameter	Description	Tier	Platform
	software		
SPLDIROUT NEW	Location of Environment output		
SPLENVIRON	Environment Identifier		
SPLSERVICEAPP	Name of Business App Server Application	BAS	
SPLWAS	Web application server installed	WEB	
SPLWEBAPP	Name of Web App Server Application	WEB	
STRIP_HTML_COMMENTS NEW	Strip out comments in code	WEB	
TIMEOUT NEW MWM	JMS Timeout in seconds	WEB	
TOP_VERSION	Product Version		
TOP_VERSION_NUM	High level product version		
URL NEW MWM	JMS JNDI based URL	WEB	
WAS_HOME	WebSphere Home	WEB	WAS
WAS_PASSWORD NEW MWM	JMS Access Password	WEB	WAS WASND
WAS_USERID NEW MWM	JMS Access Userid	WEB	WAS WASND
WASND_DMGR_HOST NEW	WebSphere Deployment Manager Host name	WEB	WASND
WASND_HOME	WebSphere ND Home	WEB	WASND
WEB_ADDITIONAL_OPT NEW	Web/Business Application Server JVM Additional Options for java command line. The value must be a valid option for the java vendor and version.		
WEB_CONTEXT_ROOT NEW	Web Context Root	WEB	
WEB_ISAPPVIEWER	Deploy/Install AppViewer to Web Application Server	WEB	
WEB_ISDEVELOPMENT	Environment used for Development if true. Sets other settings optimized for development.		

Parameter	Description	Tier	Platform
WEB_ISEXPANDED	Exploded directory or WAR/EAR files (archive). Value of false means archive format	WEB	
WEB_JMX_RMI_PORT_PERFORMANCE NEW	JMX Port for Web Application Server monitoring	WEB	
WEB_MAXAGE	Length of time for browser cache entry for text in seconds	WEB	
WEB_MAXAGEI	Length of time for browser cache entry for images in seconds	WEB	
WEB_MEMORY_OPT_MAX NEW	Web Application Server JVM Max heap size (*Xmx*)	WEB	WLS
WEB_MEMORY_OPT_MAXPERMSIZE NEW	Web Application Server JVM Max Perm size (*XX:Permsize*)	WEB	
WEB_MEMORY_OPT_MIN NEW	Web Application Server JVM Initial heap size (*Xms*)	WEB	WLS
WEB_NODENAME NEW	IBM WebSphere ND Node Name	WEB	WASND
WEB_PRELOADALL	Preload all pages on startup	WEB	
WEB_SERVER_HOME NEW	Location of Web Application Server software	WEB	
WEB_SPLPASS	Application Administration Password	WEB	
WEB_SPLUSER	Application Administration Userid	WEB	
WEB_SVRNAME	IBM WebSphere Server Name	WEB	WAS WASND
WEB_WASPASS	IBM WebSphere deployment password	WEB	WAS WASND
WEB_WASUSER	IBM WebSphere deployment userid	WEB	WAS WASND
WEB_WLAUTHMETHOD	Authentication Method	WEB	

Parameter	Description	Tier	Platform
	(BASIC or FORM)		
WEB_WLHOST	Web Server Host	WEB	
WEB_WLPAGECHECKSECONDS	Interval for recompilation of JSP	WEB	WLS
WEB_WLPORT	Web Server HTTP Port	WEB	
WEB_WLS_SVRNAME	Oracle WebLogic Server Name	WEB	WLS
WEB_WLSSLPORT	Oracle WebLogic SSL HTTP Port	WEB	WLS
WEB_WLSYSPASS	Oracle WebLogic JNDI System Password.	WEB BAS	WLS
WEB_WLSYSUSER	Oracle WebLogic JNDI System Userid	WEB BAS	WLS
WEBONLY NEW	Install Only Web Component		
WL_HOME	Oracle WebLogic Home	WEB	WLS
WLS_ADMIN_PORT NEW MWM	Admin Console Port Number for MapViewer	WEB	
WLS_PASSWORD NEW MWM	JMS Access Password	WEB	WLS
WLS_USERID NEW MWM	JMS Access Userid	WEB	WLS
WLS_WEB_WLSYSPASS NEW	Password to Oracle WebLogic console user	WEB BAS	WLS
WLS_WEB_WLSYSUSER NEW	Oracle WebLogic console User used to administrate WebLogic for Web Application and Business Application Server	WEB BAS	WLS
XAI_DBPASS	Password for Database User for Web Services component	XAI	
XAI_DBUSER	Database User used for Web Services component	XAI	
XAIApp	Name of Web Services Adapter WAR file	WEB	

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>	Name of configuration parameter from ENVIRON.INI you desire to get the value of.

For example:

ENVIRON.INI content:

```
…
DBNAME=TRAIN
…
```

Example call:

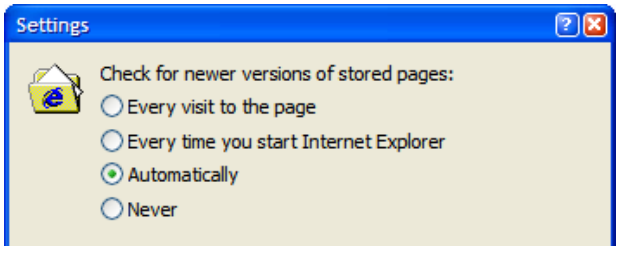
```
$ export DB=`perl $SPLEBASE/bin/getconfvalue.plx -k DBNAME`
$ echo $DB
TRAIN
```

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

Web Browser Configuration

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:

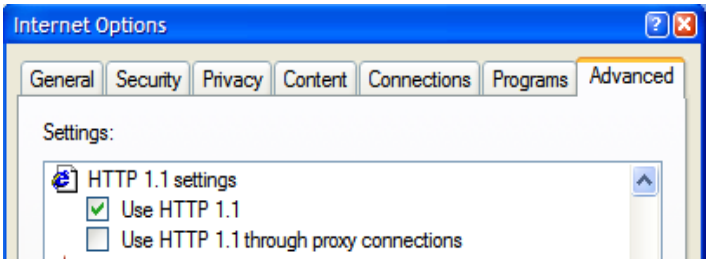
- **Microsoft Internet Explorer** - Cache settings need to be *Every visit to the page* or *Automatically*. For non-production it is recommended to be set to *Every visit to the page* or *Automatically*. For production it is recommended to be set to *Automatically* to fully exploit performance caching.



- **Mozilla Firefox** – Use the default settings with the browser for the browser. NEW

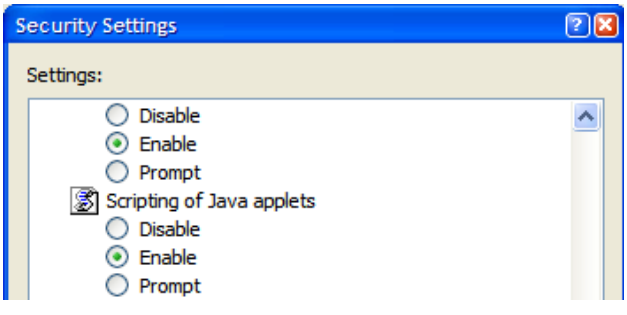
Note: Clearing the cache upon exit will clear the cached screens of the product as well

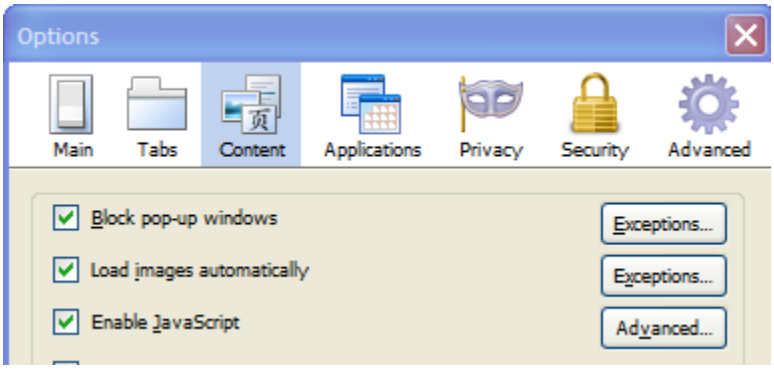
- The product requires support for the HTTP 1.1 protocol to support compression and client cache management.



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* (IE) and "Enable Java Script" (Firefox) must be enabled.





- The product uses popup windows for searches, therefore popup blockers should be configured to allow popups from the product Web application server hosts.
- Set your browser cache size to a reasonable size to hold the cached pages as needed.

Web Application Server Configuration

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 written 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 Application 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 Applications

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

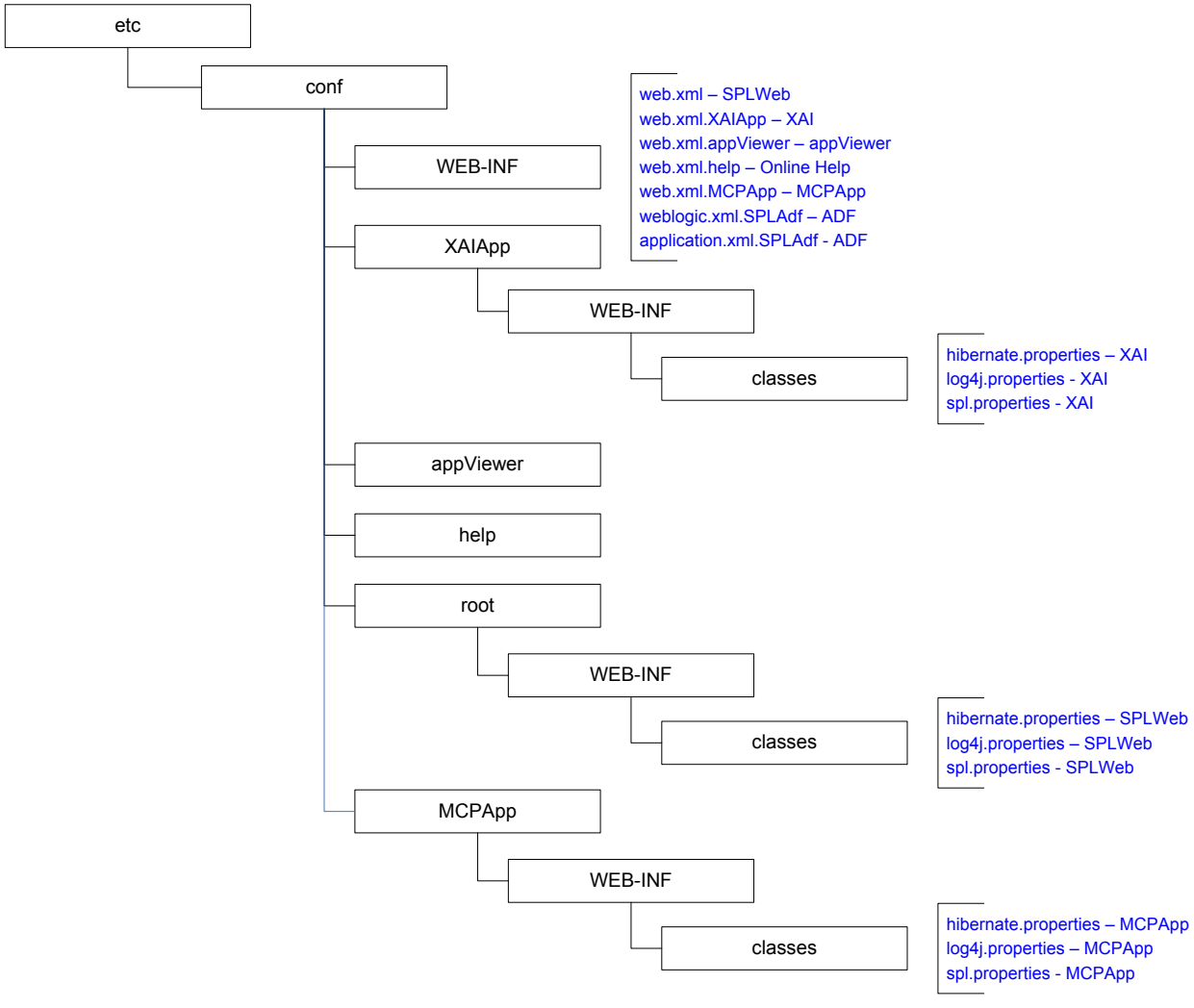
- **root** – This is the product itself is installed.
- **XAIApp** – This is the Web services adapter component.
- **appViewer** – An Application Viewer which contains a data dictionary and source viewer.
- **help** – Online Help.
- **MCPApp** – Mobile Connection Platform (MWM only)

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 initialSetup 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 WAR/EAR files deployed

with the product following the J2EE specification. In terms of configuration, the product structure within the WAR/EAR file looks like the following:



Location	Contents	Configuration Files
WEB-INF	J2EE Application Descriptor for each application	web.xml – J2EE Application Descriptor
root/WEB-INF/classes	Application Configuration files for online application	log4j.properties – Logging Configuration spl.properties – Product configuration settings
XAIApp/WEB-INF/classes	Application Configuration files for Web Services Adapter	log4j.properties – Logging Configuration spl.properties – Product configuration settings©
MCPApp/WEB-INF/classes	MCP Configuration MWM	log4j.properties – Logging Configuration

Location	Contents	Configuration Files
		spl.properties – Product configuration settings©

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 behavioral 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
maxAge (Images)	How long images are stored in the IE cache in seconds?	Derived from **WEB_MAXAGEI** parameter from ENVIRON.INI
auth-method	Security setup for product.	Derived from **WEB_WLAUTHMETHOD** parameter from ENVIRON.INI
maxAge	How long texts are stored in the IE cache in seconds?	Derived from **WEB_MAXAGE** parameter from ENVIRON.INI
fieldValuesAge	How long the static cache is kept on the Web application server in seconds?	Defaulted from 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
disablePreload	Enables or disables preload altogether (true or false). Defaults to false.	Defaulted in template
XAIServerURL	Web Services URL	Derived from Web application server parameters (**WEB_WLHOST**, **WEB_WLPORT**) in ENVIRON.INI
HTTPBasicAuthUser	Default User IDused by Web Services Adapter	Derived from **WEB_SPLUSER** parameter ENVIRON.INI
HTTPBasicAuthPasswordEnc	Encrypted password for HTTPBasicAuthUser	Derived from **WEB_SPLPASS** parameter from ENVIRON.INI NEW

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
ClassicServerURL	URL used for backward compatibility (*XAIApp only*)	Derived from Web application server parameters (WEB_WLHOST, WEB_WLPORT) in ENVIRON.INI

Note: It is highly recommended that you do not change this configuration file by extracting the configuration file from the WAR/EAR file using Java utilities, making the change manually and rebuilding the WAR/EAR file. Use initialSetup – Maintain Configuration Settings to build the WAR/EAR 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
com.splwg.schema.newValidations.F1 NEW	Internal Use Only	Defaulted from template
jmx.remote.x.access.file NEW	Access List for JMX. See JMX Security for more information.	Defaulted from template
jmx.remote.x.password.file NEW	Password file for JMX. See JMX Security for more information.	Defaulted from template
ouaf.jmx.com.splwg.base.support.management.mbean.JVMInfo	Whether JVMInfo JMX call is	Defaulted from template

Parameter	Context	Source
NEW	enabled or not. Default is **enabled** See JMX Based Monitoring for more information.	
ouaf.jmx.com.splwg.base.web.mbeans.FlushBean NEW	Whether flush comments are enabled or not. Default is **enabled** See JMX Based Monitoring for more information.	Defaulted from template
spl.mwm.abr.contextFactory NEW MWM	JMS Context Factory	Derived from **CONTEXTFACTORY**
spl.mwm.abr.password NEW MWM	Password for JMS Access Userid	Derived from **WLS_PASSWORD**
spl.mwm.abr.timeout NEW MWM	Timeout	Derived from **TIMEOUT**
spl.mwm.abr.url NEW MWM	JMS JNDI URL	Derived from **URL**
spl.mwm.abr.userid NEW MWM	JMS Access Userid	Derived from **WLS_USERID**
spl.mwm.scheduler.abr.maxProcessingTime NEW MWM	Maximum Processing time per transaction in ORS	Derived from **MAXPROCESSINGTIME**
spl.mwm.scheduler.abr.minRequests NEW MWM	Minimum number of ORS requests	Derived from **MINREQUESTS**
spl.mwm.scheduler.cleanse.interval NEW MWM	ORS In memory registry cleanse interval	Derived from **CLEANSE_INTERVAL**
spl.mwm.scheduler.mapDir NEW MWM	Map Directory for scheduler	Derived from **MAPDIR**
spl.mwm.scheduler.nodeId NEW MWM	Node Identifier	Derived from **NODEID**
spl.runtime.environ.init.dir	Location of the base configuration files.	Defaulted from template
spl.runtime.environ.isWebExpanded NEW	Whether the environment is expanded or not	Derived from **WEB_ISEXPANDED** parameter from

Parameter	Context	Source
		ENVIRON.INI
spl.runtime.environ.SPLEBASE NEW	Location of software	Derived from **SPLDIR** parameter from ENVIRON.INI
spl.runtime.environ.SPLEBASE NEW	Location of **SPLEBASE**	Defaulted from template
spl.runtime.environ.SPLOUTPUT NEW	Location of output	Derived from **SPLDIROUT** parameter from ENVIRON.INI
spl.runtime.management.connector.url.default NEW	URL used for JMX. See JMX Based Monitoring for more information	Derived from **WEB_WLHOST** and **WEB_JMX_RMI_PORT_PERFORMANCE** parameters from ENVIRON.INI
spl.runtime.management.rmi.port NEW	JMX RMI Port used for monitoring. See JMX Based Monitoring for more information	Derived from **WEB_JMX_RMI_PORT_PERFORMANCE** parameter from ENVIRON.INI
spl.runtime.mwm.scheduler.ipcStartPort NEW MWM	Start Port for ORS	Derived from **IPCSTARTPORT**
spl.runtime.options.isDevelopmentMode	Whether the environment is used for development (true or false).	Derived from **WEB_ISDEVELOPMENT** parameter from ENVIRON.INI
spl.runtime.service.extraInstallationServices	Name of Application service used for installation defaults. Default: CILTINCP	Defaulted from template.
spl.runtime.socket.file.dir	Working directory for workable sockets	Defaulted from template
spl.tools.loaded.applications	List of applications installed. Values are typically	Generated by installation script.

Parameter	Context	Source
	base,*xxx*,cm where *xxx* is the product code	

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
`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
`java-charset-name`	Specifies the Java character set to use.	Defaults from template (`UTF-8`)
`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	Derived from `WEB_WLPAGECHECKSECONDS` parameter from ENVIRON.INI
`prefer-web-inf-classes`	Loading of web classes from the WEB-INF are loaded in preference to system or Oracle WebLogic classes. Defaulted to `false`.	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
`servlet-reload-check-secs`	Defines whether an Oracle WebLogic Server will check to see if a servlet has been modified,	Defaults from template

Parameter	Context	Source
	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.	
`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.	Defaults from template (`false`)

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>
```

application.xml – ADF Application configuration

Note: This configuration file only applies to Oracle WebLogic and Oracle ADF implementations.

To use the Oracle Application Development Framework (ADF) integration the ADF components need to be deployed to a predefined ADF container. The definition of this container is controlled by the J2EE standard `application.xml` file.

Parameter	Context	Source
`context-root`	ADF context root used for calls	Set to `WEB_CONTEXT_ROOT/adf`
`display-name`	Specifies the application display name	Set to `SPLAdf`
`web-uri`	Defines location of WAR file	Set to `SPLAdf`

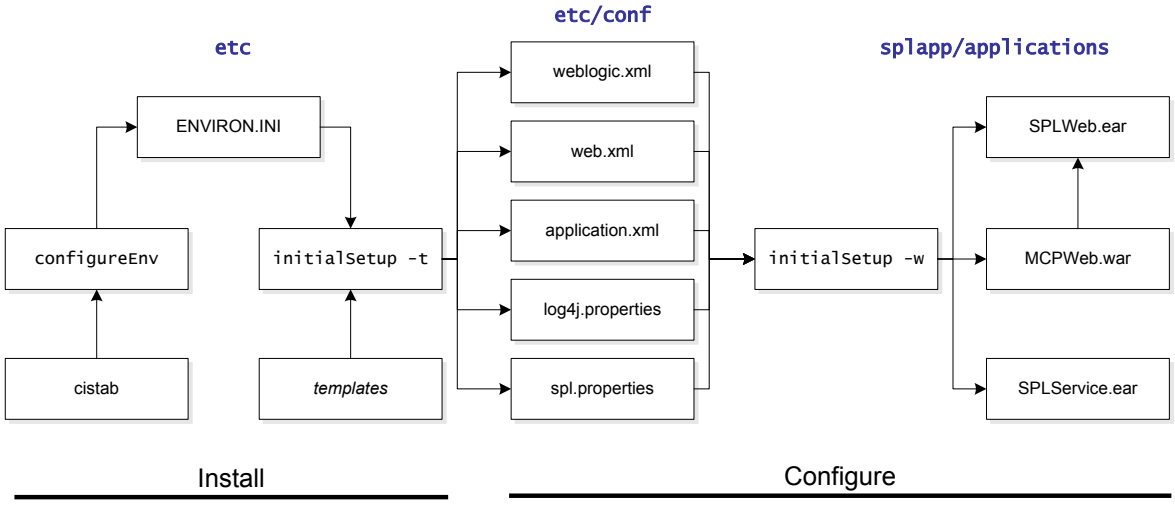
Example:

```
<?xml version = '1.0'?>
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/application_5.xsd" version="5"
xmlns="http://java.sun.com/xml/ns/javaee">
  <display-name>SPLAdf</display-name>
  <module>
    <web>
      <web-uri>SPLAdf</web-uri>
      <context-root>demo/adf</context-root>
    </web>
  </module>
</application>
```

Note: This file should not be altered unless instructed by Oracle Support.

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

  Note: The configureEnv utility should be used to make any changes to the ENVIRON.INI. Manual changes to this configuration file are not recommended.

- After the ENVIRON.INI 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 utility:
  - web.xml – J2EE Application Descriptor
  - log4j.properties – Logging Configuration
  - spl.properties – Product configuration settings
  - weblogic.xml – WebLogic Extensions
  - application.xml – ADF Application definition
- The utility uses the templates from the *templates* directory to create substituted copies of these files in a standard location. The table below lists the configuration file, the templates used from the *templates* directory and the final configuration built during the initial configuration process:

Configuration File	Destination
Online Application (root)	

web.xml
Template:
`web.xml.template`
	Linux/Unix:
	`$SPLEBASE/etc/conf/WEB-INF`
	Windows:
	`%SPLEBASE%\etc\conf\WEB-INF`

spl.properties Template: `spl.properties.template`	Linux/Unix: `$SPLEBASE/etc/conf/root/WEB-INF/classes`

Configuration File	Destination
	Windows: `%SPLEBASE%\etc\conf\root\WEB-INF\classes`

log4j.properties
Template:
`log4j.properties.template`
	Linux/Unix:
	`$SPLEBASE/etc/conf/root/WEB-INF/classes`
	Windows:
	`%SPLEBASE%\etc\conf\root\WEB-INF\classes`

weblogic.xml
Template:
`weblogic.xml.template`
	Linux/Unix:
	`$SPLEBASE/etc/conf/root/WEB-INF`
	Windows:
	`%SPLEBASE%\etc\conf\root\WEB-INF`

Web Services Adapter (XAIApp)	

web.xml (*web.xml.XAIApp*)
Template:
`web.xml.XAIApp.template`
	Linux/Unix:
	`$SPLEBASE/etc/conf/WEB-INF`
	Windows:
	`%SPLEBASE%\etc\conf\WEB-INF`

spl.properties
Template:
`spl.properties.XAIApp.template`
	Linux/Unix:
	`$SPLEBASE/etc/conf/XAIApp/WEB-INF/classes`
	Windows:
	`%SPLEBASE%\etc\conf\XAIApp\WEB-INF\classes`

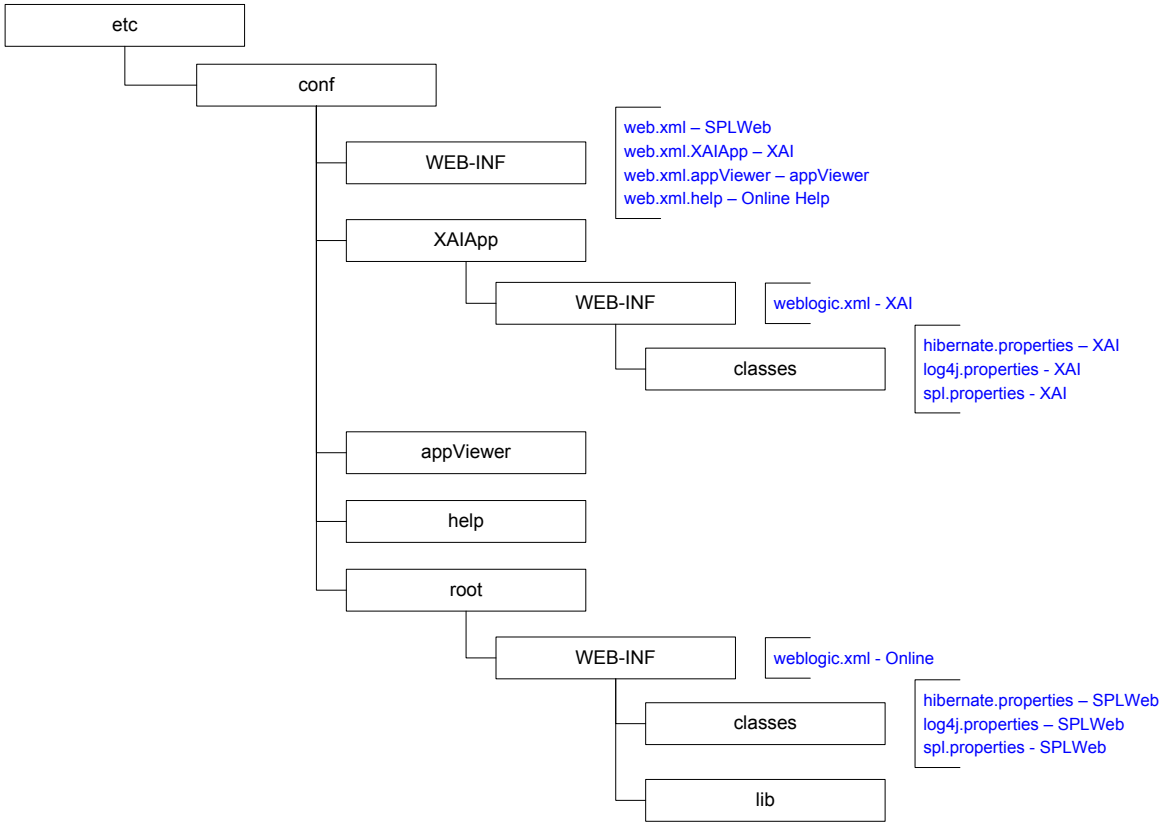
log4j.properties
Template:
`log4j.properties.XAIApp.template`
	Linux/Unix:
	`$SPLEBASE/etc/conf/XAIApp/WEB-INF/classes`
	Windows:
	`%SPLEBASE%\etc\conf\XAIApp\WEB-INF\classes`

weblogic.xml
Template:
`weblogic.xml.XAIApp.template`
	Linux/Unix:
	`$SPLEBASE/etc/conf/XAIApp/WEB-INF`
	Windows:
	`%SPLEBASE%\etc\conf\XAIApp\WEB-INF`

Application Viewer (appViewer)	
web.xml (*web.xml.appViewer*) Template:	Linux/Unix: `$SPLEBASE/etc/conf/WEB-INF`

Configuration File	Destination
web.xml.appViewer.template	Windows: %SPLEBASE%\etc\conf\WEB-INF
Help Application (help)	
web.xml (*web.xml.help*) Template: web.xml.help.template	Linux/Unix: $SPLEBASE/etc/conf/WEB-INF Windows: %SPLEBASE%\etc\conf\WEB-INF
MCP Application (MCPApp) MWM	
web.xml *(web.xml.MCPApp)* Template: MWM_web.xml.MCPApp.template	Linux/Unix: $SPLEBASE/etc/conf/WEB-INF Windows: %SPLEBASE%\etc\conf\WEB-INF
spl.properties Template: MWM_spl.properties.MCPApp.template	Linux/Unix: $SPLEBASE/etc/conf/MCPApp/WEB-INF/classes Windows: %SPLEBASE%\etc\conf\MCPApp\WEB-INF\classes
log4j.properties Template: MWM_log4j.properties.MCPApp.template	Linux/Unix: $SPLEBASE/etc/conf/MCPApp/WEB-INF/classes Windows: %SPLEBASE%\etc\conf\MCPApp\WEB-INF\classes
SPLAdf Application (ADF Integration)	
application.xml (*application.xml.SPLAdf*) Template: MWM_application.xml.SPLAdf.template	Linux/Unix: $SPLEBASE/etc/conf/WEB-INF Windows: %SPLEBASE%\etc\conf\WEB-INF
weblogic.xml (*weblogic.xml.SPLAdf*) Template: MWM_weblogic.xml.SPLAdf.template	Linux/Unix: $SPLEBASE/etc/conf/WEB-INF Windows: %SPLEBASE%\etc\conf\WEB-INF

The locations of the configuration files can be summarized in the following figure:



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

  Note: Any manual changes are overwritten after running the initialSetup utility unless the change is reflected in the appropriate template (see Implementing Custom Templates for more information). 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 initialSetup utility with the **–w** option must be executed. This will build the necessary WAR/EAR files to be deployed into the J2EE Web application server. This step is optional if configuration overrides are in use.

Depending on the architecture, the initialSetup 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 for the Web Application Server then you must run the following utilities in the order indicated:

  1. Execute the configureEnv utility to reflect the parameter change in the ENVIRON.INI.

2. Execute the initialSetup utility (with the **–t** option) to rebuild the configuration files using the ENVIRON.INI and provided template files. This will reset the configuration to the contents of the base template files or custom template (if used).
3. Any configuration changes that are overridden by templates (base or custom) must be manually reapplied (if necessary).
4. Execute the initialSetup utility (with the **–w** option) to implement the configuration files in the product Web Application Server files. This step is not necessary of you are using configuration overrides.

- If the change is to any setting not contained in the ENVIRON.INI 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. Make any manual changes to the relevant configuration files.
    2. Execute the initialSetup (with the **-w** option) utility to implement the configuration files in the product Web Application Server files. This step is not necessary of you are using configuration overrides.

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:

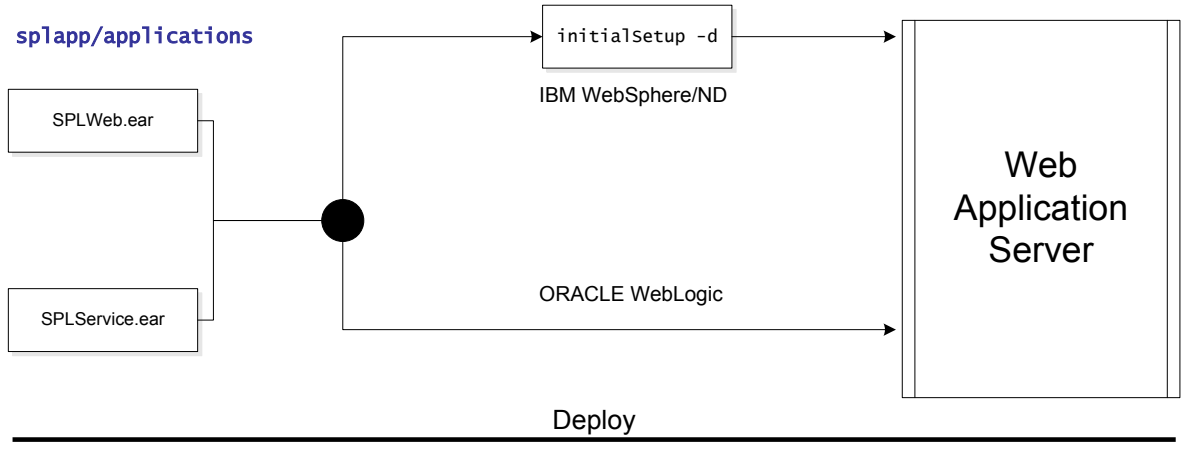
1. Use the deployment utilities provided on the console of the J2EE Web application server. The WAR/EAR files that are available under `$SPLEBASE/splapp/applications` (or `%SPLEBASE%\splapp\applications` for Windows) can be manually deployed using the console. Refer to the Installation Guide for specific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

Note: 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 IBM WebSphere or IBM WebSphere ND platform, use the initialSetup utility (with the **–d** option) utility. This will call the relevant IBM WebSphere utility to perform the deployment.
- For Oracle WebLogic, no additional deployment is necessary as the product automatically detects Oracle WebLogic and allows Oracle WebLogic to read the WAR/EAR 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 **WEB_CONTEXT_ROOT** application by default using **WEB_WLSYSUSER** and **WEB_WLSYSPASS** from the ENVIRON.INI as administration credentials.
IBM WebSphere	Deployed to **WEB_APP** Application on **WEB_SVRNAME** server by default using **WEB_WASUSER** and **WEB_WASPASS** from ENVIRON.INI as administration credentials.
IBM WebSphere ND NEW	Deployed to **WEB_APP** Application on **WEB_SVRNAME** server on **WEB_NODENAME** by default using **WEB_WASUSER** and **WEB_WASPASS** from ENVIRON.INI as administration credentials.

The Web Application should be available from the Web Application Server.

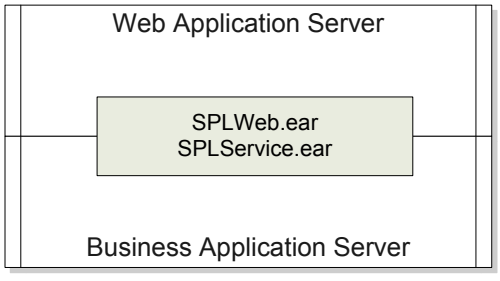
Business Application Server Configuration

It is possible for the Business Application Server logic to be separated from the Web Application Server component. Essentially the product has been split into TWO distinct EAR files:

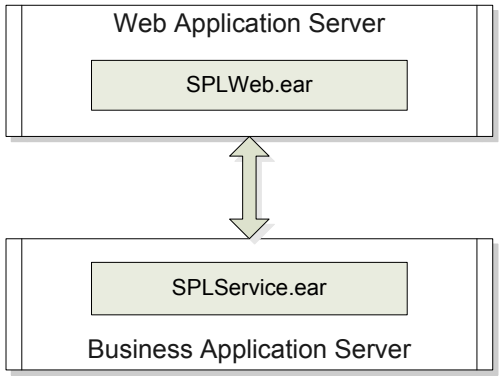
- **SPLWeb.ear** – This contains the Web application server component for the product.
- **SPLService.ear** - This contains the Business Application Server component for the

product.

There are two modes of installation:



Local install



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 WAR/EAR files generated are deployed separately with the **SPLWeb.ear** EAR file deployed to the Web application server as outlined in Web Application Server Deployment Process and **SPLService.ear** 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. Refer to the discussion of the

Business Application Server Concepts

As mentioned previous 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.

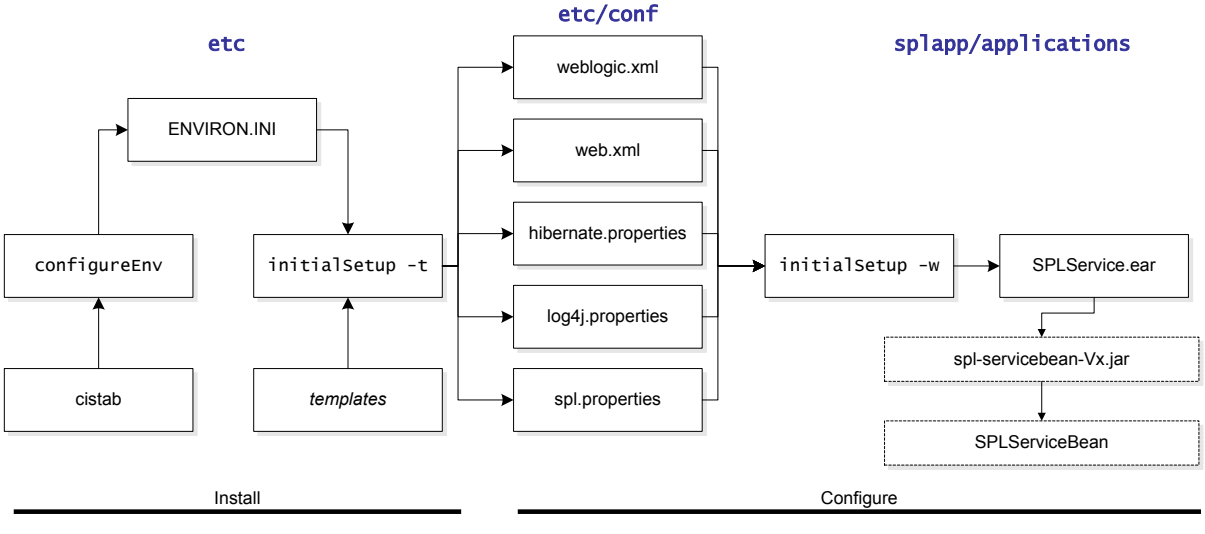
The Business Application Server was designed to fit within the same concepts as the Web Application Server. The main differences are:

- 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. 2.0.0).
- 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 utility is used during installation time and can be used post implementation to set parameters in the ENVIRON.INI. If any parameters are derived or set from the ENVIRON.INI (see "Source" column in the relevant section) then the configureEnv utility should be used to maintain them.

Note: The configureEnv utility should be used to make ANY changes to the ENVIRON.INI.

Manual changes to this configuration file are not recommended.

- After the ENVIRON.INI 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:
  - log4j.properties – Logging Configuration
  - spl.properties – Product configuration settings
  - hibernate.properties – Database connectivity properties
  - 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	Destination
Service Bean	

web.xml
Template:
`web.xml.template`
	Linux/Unix:
	`$SPLEBASE/etc/conf/WEB-INF`
	Windows:
	`%SPLEBASE%\etc\conf\WEB-INF`

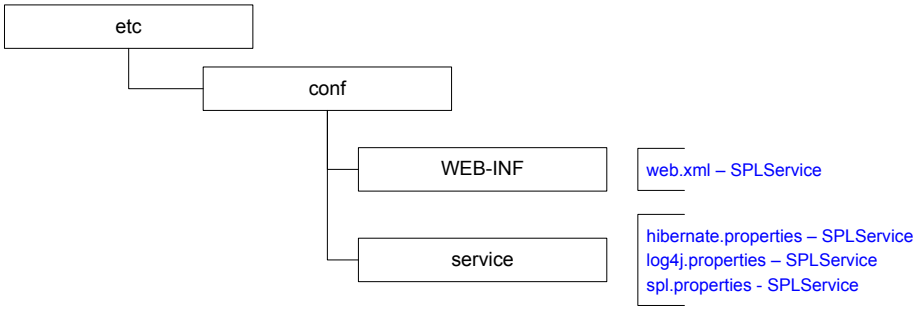
spl.properties
Template:
`spl.properties.service.template`
	Linux/Unix:
	`$SPLEBASE/etc/conf/service`
	`$SPLEBASE/splapp/businessapp/properties`
	Windows:
	`%SPLEBASE%\etc\conf\service`
	`%SPLEBASE%\splapp\businessapp\properties`

log4j.properties
Template:
`log4j.properties.service.template`
	Linux/Unix:
	`$SPLEBASE/etc/conf/service`
	`$SPLEBASE/splapp/businessapp/properties`
	Windows:
	`%SPLEBASE%\etc\conf\service`
	`%SPLEBASE%\splapp\businessapp\properties`

hibernate.properties
Template:
`hibernate.properties.web.template`
	Linux/Unix:
	`$SPLEBASE/etc/conf/root/WEB-INF/classes`
	`$SPLEBASE/etc/conf/XAIApp/WEB-INF/classes`
	`$SPLEBASE/etc/conf/service`
	Windows:
	`%SPLEBASE%\etc\conf\root\WEB-INF\classes`

Configuration File	Destination
	%SPLEBASE%\etc\conf\XAIApp\WEB-INF/classes %SPLEBASE%\etc\conf\service

The locations of the configuration files can be summarized in the following figure:



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

  Note: Any manual changes are overwritten after running the initialSetup utility unless the change is reflected in the appropriate template (see custom templates for more information). Backups should be made of any changes and then manually reapplied to reinstate all manual changes.

- To reflect configuration changes into the product Business EJB Applications the initialSetup utility, with the **–w** option, must be executed. This will build the necessary **spl-servicebean-<*version*>.jar** (where **<*version*>** is the version of the product used) and the **SPLService.ear** EAR file to be deployed into the J2EE Web application server. This step is optional if configuration overrides are in use (refer the discussion of allowing the externalization of configuration settings for alternative methods).

Depending on the architecture used, the initialSetup will generate one or more EAR files.

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 for the Business Application Server then you must run the following utilities in the order indicated:

  1. Execute the configureEnv utility to reflect the parameter change in the ENVIRON.INI.
  2. Execute the initialSetup utility (with the **–t** option) to rebuild the configuration files using the ENVIRON.INI and provided template files. This will reset the configuration to the contents of the base template files or custom template (if used.

3. Any configuration changes that are overridden by templates (base or custom) must be manually reapplied (if necessary).

4. Execute the initialSetup utility (with the **–w** option) to implement the configuration files in the product Business Application files. This step is not necessary of you are using configuration overrides

- If the change is to any setting not contained in the ENVIRON.INI 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. Make any manual changes to the relevant configuration files.

2. Execute the initialSetup, with the **-w** option, utility to implement the configuration files in the product Business Application Server files. This step is not necessary of you are using configuration overrides.

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:

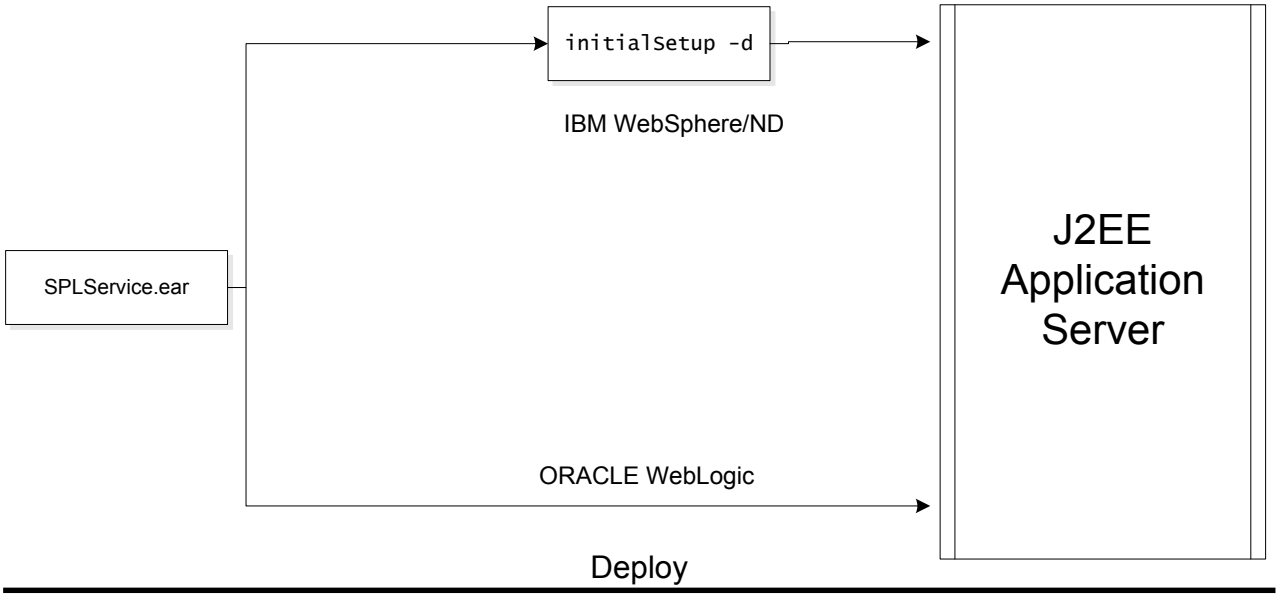
- Use the deployment utilities provided on the console of the J2EE Web application server. The WAR/EAR files that are available under `$SPLEBASE/splapp/applications` (or `%SPLEBASE%\splapp\applications` for Windows) can be manually deployed using the console. Refer to the Installation Guide for specific 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 WAR/EAR 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 EJB 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 IBM WebSphere/WebSphere ND platform, use the initialSetup utility with the **–d** option. This will call the relevant IBM provided utility to deploy the WAR/EAR files into the IBM WebSphere instance.
- For Oracle WebLogic, no additional deployment is necessary as the product automatically detects WebLogic and allows WebLogic to read the WAR/EAR 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 **WEB_CONTEXT_ROOT** application by default using **BSN_WLSYSUSER** and **BSN_WLSYSPASS** from the ENVIRON.INI as administration credentials.
IBM WebSphere	Deployed to **BSN_APP** Application on **BSN_SVRNAME** server by default using **BSN_WASUSER** and **BSN_WASPASS** from ENVIRON.INI as administration credentials.
IBM WebSphere ND NEW	Deployed to **BSN_APP** Application on **BSN_SVRNAME** server on **BSN_NODENAME** by default using **BSN_WASUSER** and **BSN_WASPASS** from ENVIRON.INI 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 WAR/EAR files deployed with the product following the J2EE specification (refer the discussion of allowing the externalization of configuration settings for alternative methods). In terms of configuration, the product structure within the WAR/EAR file looks like the following:

Location	Contents	Configuration Files
root/WEB-INF	J2EE Application Descriptor for Business Application Server	web.xml – J2EE Application Descriptor
root/WEB-INF/classes	Application Configuration files for Business Application Server	log4j.properties – Logging Configuration hibernate.properties – Database connectivity properties spl.properties – Product configuration settings

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 behavioral 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:

Parameter	Context	Source
fieldValuesAge	How long the static cache is kept on the Web application server in seconds?	Defaults from template

Note: It is highly recommended that you do not change this configuration file by extracting the configuration file from the WAR/EAR file using java utilities, making the change manually and

rebuilding the WAR/EAR. Use the initialSetup utility, with the –w option, to build the WAR/EAR 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 Server 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
com.splwg.batch.scheduler.daemon	Whether the online daemon is used or not.	Derived from **BATCHDAEMON** parameter from ENVIRON.INI
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
com.splwg.grid.online.enabled	Whether the online daemon is uses the lightweight batch framework or not.	Derived from **BATCHENABLED** parameter from ENVIRON.INI
jmx.remote.x.access.file NEW	Access List for JMX. See JMX Based Monitoring for more information.	Defaulted from template
jmx.remote.x.password.file NEW	Password file for JMX. See JMX Based Monitoring	Defaulted from template

Parameter	Context	Source
	for more information.	
ouaf.jmx.com.splwg.ejb.service.management.PerformanceStatistics NEW	Whether JMX is enabled or not. See JMX Based Monitoring for more information.	Derived from **BSN_JMX_RMI_PORT_PERFORMANCE** parameter from ENVIRON.INI
spl.geocodeDatasource.contextFactory NEW	GIS component Datasource	Defaulted from template
spl.geocodeDatasource.password NEW	Default password to access GIS component	Derived from **GIS_WLSYSPASS** parameter from ENVIRON.INI
spl.geocodeDatasource.url NEW	URL to GIS component	Derived from **GIS_URL** parameter from ENVIRON.INI
spl.geocodeDatasource.user NEW	Default User to access GIS component	Derived from **GIS_WLSYSUSER** parameter from ENVIRON.INI
spl.runtime.cobol.cobrcall	If COBOL is used, whether remote calls are supported. (true or false). Defaults to ***false***.	Generated from template.
spl.runtime.cobol.encoding	If COBOL is used, the character set supported by the Business Application Server	Derived from **COLLATE**, **NLS_LANG** or **DB2CODEPAGE** parameters from ENVIRON.INI
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.jvmcommand	If COBOL is used, the Java executable	Derived from **JVMCOMMAND**

Parameter	Context	Source
	to be used for COBOL Child JVMs.	parameter from ENVIRON.INI
spl.runtime.cobol.remote.jvmcount	If COBOL is used, the Number of COBOL Child JVM's. Default is 2.	Derived from **BSN_JVMCOUNT** parameter from ENVIRON.INI
spl.runtime.cobol.remote.jvmMaxLifetimeSecs	If COBOL used, how long in seconds Child JVM will live before being reset.	Defaulted from template
spl.runtime.cobol.remote.jvmMaxRequests	If COBOL used, how many requests processed before Child JVM before being reset.	Defaulted from template
spl.runtime.cobol.remote.jvmoptions	If COBOL is used, the Java memory footprint to be used for COBOL Child JVMs.	Derived from **JVMCHILD_OPTIONS** parameter from ENVIRON.INI
spl.runtime.cobol.remote.rmiStartPort	If COBOL is used, Starting port range for COBOL Child JVM's. Defaults to 6503	Derived from **BSN_RMIPORT** parameter from ENVIRON.INI
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	If COBOL used, number of cursors cached. Defaults to 10.	Defaulted from template
spl.runtime.cobol.sql.disableQueryCache	If COBOL used, whether the query cache is disabled. Defaults to **false**.	Defaulted from template
spl.runtime.cobol.sql.fetchSize	If COBOL used,	Defaulted from

Parameter	Context	Source
	size of fetch buffers for SQL statements. Defaults to 150.	template
spl.runtime.environ.init.dir	Location of the base configuration files.	Defaulted from template
spl.runtime.environ.SPLEBASE NEW	Location of **SPLEBASE**	Defaulted from template
spl.runtime.management.connector.url.default NEW	URL used for JMX. See JMX Based Monitoring for more information	Derived from **BSN_WLHOST** and **BSN_JMX_RMI_PORT_PERFORMANCE** parameters from ENVIRON.INI
spl.runtime.management.rmi.port NEW	JMX RMI Port used for monitoring. See JMX Based Monitoring for more information	Derived from **BSN_JMX_RMI_PORT_PERFORMANCE** parameter from ENVIRON.INI
spl.runtime.oracle.statementCacheSize	The SQL cache size allocation for SQL statements. Defaults to 300.	Defaulted from template
spl.runtime.service.extraInstallationServices	Name of Application service used for installation defaults.	Defaulted by template.
spl.runtime.socket.file.dir	Working directory for workable sockets	Defaulted from template
spl.runtime.sql.highValue	High Value used for processing	Derived from **HIGHVALUE** parameter from ENVIRON.INI
spl.runtime.utf8Database	Whether the database supports the UTF8 characterset. (true or false).	Derived from **COLLATE**, **NLS_LANG** or **DB2CODEPAGE** parameters from ENVIRON.INI

Parameter	Context	Source
spl.tools.loaded.applications	List of applications installed. Values are typically base,*xxx*,cm where *xxx* is the product code.	Generated by installation script.
com.splwg.schema.newValidations.F1 NEW	Internal Use Only	Defaulted from template

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 Oracle Universal Connection Pool (UCP) connection pooling or JNDI based connection pooling.

The online and Web Service components of the product use JNDI based connection pools and the batch component uses UCP based connection pools.

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 UCP or JNDI connection pooler, it provides a comprehensive connectivity tool for the COBOL/java to operate effectively against the database.

The product uses the Hibernate and either JNDI or UCP 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://www.oracle.com/technology/software/tech/java/sqlj_jdbc/htdocs/ucp.html for more information on the technology aspects of Hibernate and UCP.

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.cache.use_second_level_cache	May be used to completely disable the

Setting	Usage
	second level cache, which is enabled by default for classes which specifies a cache mapping. Defaults to false
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.connection.datasource NEW	JNDI data source.
hibernate.connection.driver_class	This is the JDBC driver class used by Hibernate.
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. If the value is prefixed by "ENC" then the password is encrypted.
hibernate.connection.provider_class BATCH	The classname of a custom Connection Provider which provides JDBC connections to Hibernate. The product uses the UCP Connection provider. Other providers are not supported.
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.
hibernate.connection.username	This is the user ID used to connect to the database. This value is sourced from the **DBUSER** parameter from the ENVIRON.INI©
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.
hibernate.jdbc.batch_size	A non-zero value enables use of JDBC2 batch updates by Hibernate. Defaults to 30

Setting	Usage
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.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.query.factory_class	Chooses the HQL parser implementation.
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.show_sql	Write all SQL statements to console. Defaults to false.
hibernate.transaction.factory_class	The classname of a Transaction Factory to use with Hibernate Transaction API.
hibernate.ucp.connection_wait_timeout NEW BATCH	Specifies how long, in seconds, an application request waits to obtain a connection if there are no longer any connections in the pool
hibernate.ucp.inactive_connection_timeout NEW BATCH	Specifies how long, in seconds, an available connection can remain idle before it is closed and removed from the pool.
hibernate.ucp.max_idle_time NEW BATCH	Not used
hibernate.ucp.max_size NEW BATCH	Maximum Pool Size
hibernate.ucp.max_statements NEW BATCH	SQL Buffer size
hibernate.ucp.min_size NEW BATCH	Minimum Pool Size

For a more indepth description of these parameters and others not included with the product see http://www.hibernate.org and http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/htdocs/ucp.html .

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 may overwrite these changes unless template overrides are used. 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
   - Business Application Server - log4j.properties
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
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) Execute the initialSetup utility, with the **–w** option, to reflect the changes in the WAR/EAR file.

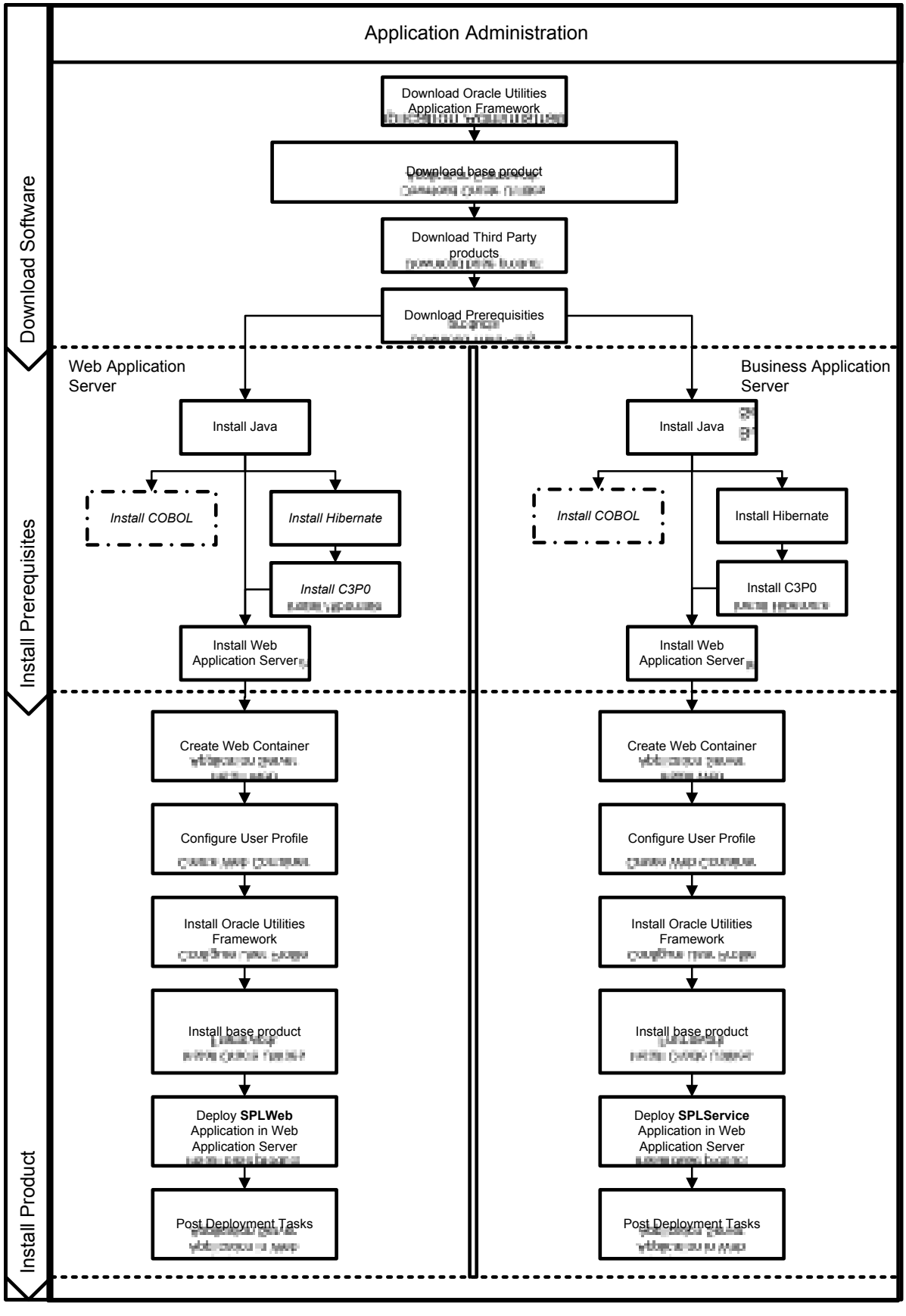
5) To deploy the application refer to the Web Application Server Deployment Process or Business Application Server Deployment Process

Installation of decoupled servers

One of the options available with the 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** EAR file is deployed to the Web Application Server and the **SPLService.ear** 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 Web Application Server section of this document. For the configuration settings for the Business Application Server, refer to the Business Application Server section of this document.

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

By default the database connection for Oracle databases 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 override the default URL specify the following:

- Log on to the server containing the Business application server using the

administration account for the desired environment (for example, **splsys**).

- Execute the splenviron utility, with the **–e** option, to attach to the desired environment to change.
- Execute the configureEnv utility and choose to change menu block 4 (Database).
- Change the *Database Override Connection String* to the desired custom JDBC url.
- Press **p** to save the change to the ENVIRON.INI.
- Execute initialSetup, with the **–t** option, to reflect the change in the hibernate.properties files. This may overwrite custom changes if custom templates are not used.
- Execute initialSetup, with the **–w** option, to include the configuration changes in the WAR/EAR files. This option is not required if externalization of configuration is implemented.
- For selected platforms redeployment of the WAR/EAR files is required as per Business Application Server deployment process.

The following example uses the 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)
    )
)
```

Refer to Oracle RAC support for alternatives.

Example URL using the 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

Note: These policies are not active in the default configuration as part of the installation process there must be manually added to online spl.properties files or added to a custom template version of spl.properties.services.template.

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 Application Server. In addition to the Web Application Server cache, information is also cached on each client browser.

Server Cache

Note: Maintenance of the cache is performed automatically by the product. Whilst there are commands

to force refreshes of the cache, these are designed for administrator and developer use only. Additional security setup is required to enable individual users to access to the facilities below.

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).

Typically, this information

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 Application Server is restarted or as automatically refreshed as controlled by the `fieldValuesAge` parameter on the Web Application Server web.xml configuration file. This means that fresh values are retrieved from the database upon first use by end users.

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 restart your Web Application Server, you can either use the relevant operation on the JMX FlushBean Mbean available on the Web Application Server or issue a custom browser URL to issue the appropriate command (see below).

Note: To use the browser URL for the resetting of the cache the user must be logged on to the product browser interface and have access to the ***F1ADMIN*** *application service.* NEW

Function	JSP	MBean Operation
Refresh all cache	`flushAll.jsp`	`flushAll`
Refresh all drop down data	`flushDropdownCache.jsp`	`flushDropDownCache`
Refresh field labels	`flushMessageCatalog.jsp`	`flushMessageCatalog`

Function	JSP	MBean Operation
Refresh Fields and FK information NEW	flushFieldAndFKMetaData.jsp	flushFieldAndFKMetaData
Refresh menu items	flushMenu.jsp	flushMenu
Refresh messages NEW	flushMessaging.jsp	flushMessaging
Refresh navigation keys NEW	flushNavigationInfo.jsp	flushNavigationInfo
Refresh portals and zones	flushPortalMetaInfo.jsp	flushPortalMetaInfo
Refresh screen style sheets	flushUI_XSLs.jsp	flushUIXSLs
Refresh security	flushSystemLoginInfo.jsp	flushSystemLoginInfo
Refresh specific drop down data	flushDropDownField.jsp	flushDropDownField

Note: It is recommended that the "Refresh all cache" is used for non-production and production systems. The other commands are designed for primarily for development use only. Refer to the Oracle Utilities SDK documentation for more information about the options available with the commands.

Note: When using these commands the cache will be reloaded over time with fresh data. As the data is loaded there is a negligible delay in each transaction that reloads data into the cache for the first time. Therefore it is recommended not to execute this command frequently.

Client Cache

In addition to the server cache, information is cached on each user browser. After clearing the cache that's maintained on the Web Application Server, it is recommended to also clear the cache that is maintained on the client browser (if possible). To do this, follow the following steps:

Browser	Steps
Microsoft Internet Explorer	• Select *Tools* on your browser menu bar • Select *Internet Options* on the menu that appears. • Click the *Delete Files* button on the pop-up that appears. • Click the *Delete all…* button on the subsequent pop-up that appears and then click OK. • Enter the standard product URL to re-invoke the product.
Mozilla Firefox NEW	• Select *Tools* from your browser menu bar. • Click *Options* on the Tools menu. • Select the *Advanced* tab from the Options dialog.

Browser	Steps
	• Select the *Network* tab from the Advanced tab. • Click on the *Clear Now* button. • Enter the standard product URL to re-invoke the product.

Note: Each user's cache is automatically refreshed as controlled by the ***maxAge*** *and* ***maxAgeI*** *parameters in the Web Application Server web.xml configuration file. . We recommend that you set these parameter to 1 second on development / test environments and 28800 seconds (8 hours) on production environments.*

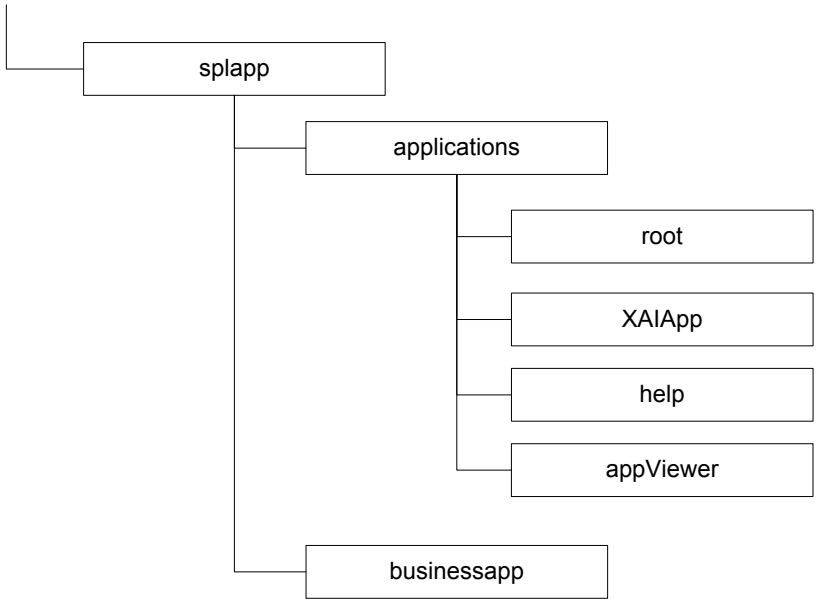
Oracle WebLogic: Expanded or Archive Format NEW

By default, the product is built into a set of WAR/EAR files and deployed in this format on Oracle WebLogic and IBM WebSphere/ND to operate. For Oracle WebLogic it is possible to use *expanded* mode rather than the WAR/EAR format. This mode allows the Oracle WebLogic instance directories access to the directories and files used by the J2EE components of the product without the need for WAR/EAR files. This has a number of key advantages:

- Changes to the individual files in the product (such as JSP's or graphics) do not require a rebuild of the WAR/EAR file.
- Outage time to deploy and execute the WAR/EAR file is reduced as Oracle WebLogic reads the files directly. In the deployment process, Oracle WebLogic loads the WAR/EAR file and uncompressed it to a staging or temporary location for actual execution. This is greatly reduced under *expanded* mode as the files are already uncompressed.
- Application of patches and service packs is faster as the patch installer does not need to rebuild the WAR/EAR files after applying patches.

This expanded mode is suggested for non-production and demonstration environments and is not recommended for production (the default is *Archive* [non-expanded] mode) as the during the WAR/EAR process additional integrity checks are performed and security control of individual application files adds higher security requirements to production.

The figure below illustrates the expanded mode main directories:



- Under the **root** directory are the product specific subdirectories for each subsystem or part of the online component of the product.
- Under the **XAIApp** directory are the product specific subdirectories for each subsystem or part of the Web Services component of the product.
- Under the **busienssapp** directory are the business object specific files for each subsystem or part of the online component of the product.
- The help and AppViewer contain an expanded mode version of the **help** HTML (and related files) and **appViewer** generated files (after running genappvieweritems).

Implementing Custom Templates NEW

As described in the Web Application Server Configuration Process and Business Application Server Configuration Process the configuration files used in the product are built from templates. These templates are typically located in the **$SPLEBASE/templates** (or **%SPLEBASE%\templates** on Windows) subdirectory of each environment.

*Note: The file **FW_template_structure.xml** in the **structures** subdirectory lists all the templates and their destination paths. This file should not be altered unless instructed by Oracle Support.*

By default the product uses the base produce provided templates to build the configuration files. These configuration files are usually adequate for most needs in non-production but usually require some customization for production or site specific standards not covered by the base templates. In the past the site had two options:

- **Make custom changes to the configuration file directly** – This can performed against the **$SPLEBASE/etc/conf** (**%SPLEBASE%\etc\conf** on Windows) copies of the configuration files. The issue here is that if the configuration files are reset back to the templates intentionally or unintentionally, using the initialSetup utility, custom manual changes may be lost if not reapplied manually.

- **Make custom changes to base configuration templates** – In extreme conditions it was possible to make manual changes to the base product templates to reflect your site standards and customizations. The issue is that new releases of the templates for new features would overwrite any customizations if not reapplied manually.

To address this it is now possible to override base product templates with a copy of the template (a custom template). This can be achieved by copying the desired base template in the templates directory to the same name prefixed with "**cm.**". The initialSetup utility will use the custom template instead of the base template.

The process to implement this is as follows:

- Identify the template in the that is used by the desired configuration file. Use the information in the Web Application Server Configuration Process and Business Application Server Configuration Process sections of this document to help identify the templates used for each configuration file.
- Copy the desired template in the **$SPLEBASE/templates** (or **%SPLEBASE%\templates** on Windows) subdirectory to the same name but prefixed with a "**cm.**". This will be the override custom template. To disable the custom template at any time either rename the template to another name or remove it from the subdirectory.
- Make the necessary adjustments to the custom template as per your site standards. Please follow any conventions used in the template including use of environment variables or configuration settings from ENVIRON.INI.
- Use initialSetup as per Web Application Server Configuration Process and Business Application Server Configuration Process sections of this document to use the template to generate the new configuration files and incorporate the changes in the product.

Note: If custom templates are implemented, it is the sites responsibility to maintain the custom templates to reflect any changes in the base templates for new, changed or removed functionality.

Additional templates

The templates mentioned in previously in this document are the main configuration file based templates. There are additional configuration files that are built and used for various purposes. Most of these configuration files are used internally for management of the infrastructure and generation of utilities.

*Note: The file **FW_template_structure.xml** in the **structures** subdirectory lists all the templates and their destination paths. This file should not be altered unless instructed by Oracle Support.*

There are a number of areas the templates cover:

- **Configuration Files for Oracle WebLogic** – Oracle WebLogic has specific requirements for configuration settings and files. Refer to Oracle WebLogic Configuration Support for more specific details.
- **Configuration Files for other software** – Third party software has specific

requirements for configuration files.

- **Utilities for deployment** – Additional configuration files are built to use in the deployment process to define the product applications to the relevant runtime software.
- **Internal ANT build configuration files** – Configuration and build files are built to support the configuration build process.

Note: The latter two categories of templates and configurations (utilities and ANT build files) should not be altered unless instructed by Oracle Support.

The table below lists the templates in the template directory not covered by other sections of this document applicable to the online, service and XAI components:

Templates	Configuration File	Usage
application.xml.template	applicaton_web.xml	J2EE global application configuration file, which contains common settings for the Web Application Server
application_service.xml.template	application_service.xml	J2EE global application configuration file, which contains common settings for the Business Application Server
billdirfile.ini.template	billdirfile.ini	Bill Print extract configuration file
config.xml.template config.xml.win.template	config.xml	Oracle WebLogic main configuration file. The **win.template** is used for the Windows environments.
doc1dirfile.ini.template	doc1dirfile.ini	Bill Print extract configuration file
earServiceBuild.xml.template	earServiceBuild.xml	ANT Build file for EAR file for Business Application Server
earWebBuild.xml.template	earWebBuild.xml	ANT Build file for EAR file for Web Application Server
ejb-jar.xml.template	ejb-jar.xml	Generic Business Application Server descriptor for EJB's
ibm-application-bnd.xmi.template	ibm-application-bnd.xmi	Deployment descriptor

Templates	Configuration File	Usage
		for IBM WebSphere/ND.
jarservice.xml.template	jarservice.xml	ANT Build file for jar files.
java.login.config.template	java.login.config	JAAS Login file used for XAI servlet. Refer to XAI Best Practices whitepaper KB Id: **942074.1** on My Oracle Support for more details.
jps-config.xml.template	jps-config.xml	ADF security configuration.
MPLIsUp.cmd.template	MPLIsUp.cmd	Utility to check status of MPL (if used) as called by spl[.sh] on Windows. Refer to XAI Best Practices whitepaper KB Id: **942074.1** on My Oracle Support for more details.
MPLIsUp.sh.template	MPLIsUp.sh	Utility to check status of MPL (if used) as called by spl[.sh] on Linux/UNIX. Refer to XAI Best Practices whitepaper KB Id: **942074.1** on My Oracle Support for more details.
MPLParameterInfo.xml.template	MPLParameterInfo.xml	MPL Configuration file. Refer to XAI Best Practices whitepaper KB Id: **942074.1** on My Oracle Support for more details.
setDomainEnv.cmd.template	setDomainEnv.cmd	Utility to set Domain configuration for Oracle WebLogic on Windows.
setEnv.sh.template	setEnv.sh	Utility to set Oracle WebLogic environment variables.
splcobjrun.cmd.template	splcobjrun.cmd	COBOL runtime command (if COBOL

Templates	Configuration File	Usage
		used) for Windows.
splcobjrun.sh.template	splcobjrun.sh	COBOL runtime command (if COBOL used) for Linux/Unix.
startMPL.cmd.template	startMPL.cmd	Utility to start MPL (if used) as called by spl[.sh] on Windows. Refer to XAI Best Practices whitepaper KB Id: **942074.1** on My Oracle Support for more details.
startMPL.sh.template	startMPL.sh	Utility to start MPL (if used) as called by spl[.sh] on Linux/UNIX. Refer to XAI Best Practices whitepaper KB Id: **942074.1** on My Oracle Support for more details.
startWebLogic.cmd.template	startWebLogic.cmd	Utility to start Oracle WebLogic on Windows.
startWebLogic.sh.template	startWebLogic.sh	Utility to start Oracle WebLogic on Linux/UNIX.
startWLS.sh.template	startWLS.sh	Utility invoking JVM for Oracle WebLogic.
stopMPL.cmd.template	stopMPL.cmd	Utility to stop MPL (if used) as called by spl[.sh] on Windows. Refer to XAI Best Practices whitepaper KB Id: **942074.1** on My Oracle Support for more details.
stopMPL.sh.template	stopMPL.sh	Utility to stop MPL (if used) as called by spl[.sh] on Linux/UNIX. Refer to XAI Best Practices whitepaper KB Id: **942074.1** on My Oracle Support for more details.
stopWebLogic.cmd.template	stopWebLogic.cmd	Utility to stop Oracle WebLogic on Windows.

Templates	Configuration File	Usage
system-jazn-data.xml.template	system-jazn-data.xml	ADF security store definitions.
warbuild.xml.template	warbuild.xml	ANT WAR Build file
warupdate.xml.template	warupdate.xml	ANT WAR file for updates
weblogic.policy.template	weblogic.policy	Java Security file used by Oracle WebLogic to protect the product files.
weblogic-ejb-jar.xml.template	weblogic-ejb-jar.xml	Deployment descriptor for Business Application Server for Oracle WebLogic.
XAIParameterInfo.xml.template	XAIParameterInfo.xml	XAI Configuration file. Refer to XAI Best Practices whitepaper KB Id: **942074.1** on My Oracle Support for more details.

Note: Templates not mentioned in this document that exist in the templates directory are included in one or more templates above depending on the configuration requirements. Templates relating to the Batch component of the architecture are covered in the Batch Server Administration Guide.

Oracle WebLogic Configuration Support NEW

Whilst the product supports multiple J2EE Web Application Server vendors, the product has *native* support for Oracle WebLogic. Normally the J2EE Web Application is installed and the J2EE Web Application Server components are embedded in the directories controlled by the Web Application Server software during the deployment process. The deployment process usually transfers the WAR/EAR files to the J2EE Web Application Server directories (varies according to J2EE Web Application Server software).

For Oracle WebLogic, the Oracle WebLogic software is effectively *pointed* to directories as in the product installation. This avoids Oracle WebLogic having additional copies of its configuration and WAR/EAR files under its own directory structure.

In this case the following configuration aspects of Oracle WebLogic apply:

- The **$SPLEBASE/splapp** (or **%SPLEBASE%\splapp** in Windows) subdirectory is referenced directly in the configuration files.
- In non-expanded mode (see Oracle WebLogic: Expanded or Archive Format for details), the WAR/EAR files are directly referenced from the config.xml file.
- In expanded mode (see Oracle WebLogic: Expanded or Archive Format for details),

the application files are directly reference in the **splapp** subdirectories from the config.xml file.

- The config.xml file is located under **splapp/config** rather than using the Oracle WebLogic location. Any changes made from the Oracle WebLogic console are stored in this file.
- The utilities to start and stop the Oracle WebLogic instance are located under the **splapp** subdirectory.
- The security configuration files for the Oracle WebLogic instance are located under the **splapp** subdirectory. The security repository configured is configured in the location supplied with the Oracle WebLogic instance.

Thus facility allows one installation of Oracle WebLogic to be used across many environments with each environment being independent.

Using Configuration Files outside the WAR/EAR file NEW

Typically, the configuration files specified Web Application Server Configuration Process and Business Application Server Configuration Process are embedded into the WAR/EAR files, as per the J2EE specification, ready for deployment for use at runtime. While this is generally acceptable for most sites, it also means that any configuration change requires rebuilding of the WAR/EAR files and redeployment to fully implement the configuration changes. This may add outage time to implement configuration changes.

It is possible to allow the product to use versions of the certain configuration files outside the WAR/EAR files to minimize outage time to implement changes. In most cases, a restart of the product components is necessary to implement the configuration change.

The table below outlines the configuration files that can be *externalized* from the WAR/EAR file by product component:

Component	Configuration File	Externalized
Web Application Server (root and XAIApp)	web.xml	✕
	spl.properties	✓
	weblogic.xml	✕
	log4j.properties	✓
Business Application Server	web.xml	✕
	spl.properties	✓
	hibernate.properties	✓
	log4j.properties	✓

By default, the externalization works on the following principles:

- The **SPLEBASE** environment variable must be set to the home location of the software prior to execution of the Web Application Server or Business Application Server. This must match the value configured for the environment in the cistab configuration file

on the machine.

- The external versions of the configuration files should be in their default locations (as supplied) in the **$SPLEBASE/etc/conf** (or **%SPLEBASE%\etc\conf** for Windows) subdirectories.
- The product use the external configuration file versions instead of the versions embedded in the WAR/EAR files. If you wish to revert to the embedded versions then the site can either rename the **conf** subdirectories to prevent the external configuration files being detected or ensuring the **SPLEBASE** environment is not set.

*Warning: If the **conf** subdirectories are renamed they should be reverted to their original names before ANY single fix, service pack or upgrade is performed to prevent configuration reset to base templates or installation failure.*

This facility is useful for a number of situations:

- If any passwords are changed that are used by the product on a regular basis, reflecting changes in the configuration files directly or using templates is easier using externalized configuration files. The WAR/EAR files do not need to be rebuilt and redeployed and this can save time.
- During the initial phases of production or when traffic volumes fluctuate, it may be necessary to tune specific settings. This allows experimentation of the changes before committing to specific values. It allows greater level of *flexibility* in configuration change.

Note: It is recommended to ensure that in the long term that both the external versions and embedded versions are kept in synch on a regular basis to prevent configuration issues. This can be done using standard maintenance windows as necessary.

Oracle RAC Support NEW

As pointed out in previous sections of this document it is possible to override the standard JDBC URL with a custom URL for advanced database facilities. One of the most common uses of the custom JDBC URL facility is to support the Oracle Real Application Clustering (RAC) facility of the database. Refer to the Oracle RAC documentation for a full description of the RAC facility.

To support Oracle RAC a few configuration settings must be configured:

- The relevant Oracle RAC aware JDBC URL must be specified for "*Database Override Connection String*" setting in the installation using the configureEnv utility to set the **DB_OVERRIDE_CONNECTION** setting.

  Example RAC URL:

```
jdbc:oracle:thin:@(DESCRIPTION= \
    (LOAD_BALANCE=yes) \
       (ADDRESS=(PROTOCOL=TCP)(HOST=node1)(PORT=1234)) \
       (ADDRESS=(PROTOCOL=TCP)(HOST=node2)(PORT=5678))\
 (CONNECT_DATA=(service_name=orcl)))
```

- The Oracle Notification Service (ONS) configuration must be specified for the "*ONS Server Configuration*" setting in the installation using the configureEnv utility to set the **ONSCONFIG** setting. This is required for correct configuration of the RAC Fast Connection Failover facility.

Example ONS setting:

```
onsconfig=nodes=node1:1345,node2:1678
```

- The RAC Initial Limit, RAC Minimum Limit and RAC Maximum Limits must be set using the advanced section of the configureEnv utility to set the **RAC_INITIALLIMIT**, **RAC_MINLIMIT** and **RAC_MAXLIMIT** settings respectively. This is required for correct configuration of the RAC Fast Connection Failover facility.

These settings apply to the UCP and JNDI based data sources.

Using JNDI Based Data Sources NEW

By default, the Oracle Utilities Application Framework requires database connection pooling for performance reasons. Typically connection pool is established using a number of means to create and manage a shared pool of connections per component to the database. The size of the pool reacts to the fluctuations in database traffic according to its configuration. This information is stored in the hibernate.properties file per component.

For the online and Web Services component of the product, it is possible to utilize a connection pool defined in your J2EE Web Application Server. This allows the J2EE Web Application Server itself to control the connection pools and may offers alternative facilities that are more appropriate for your site. For example, using the connection pool within the J2EE Web Application Server allows the console of the J2EE Web Application Server to be used for administration of database connections rather than the hibernate.properties file. This allows flexibility in configuration and it is controlled from a central point for the Web Application Server.

To use this facility the following needs to be done:

- The JDBC connection pool needs to be defined to the Java Naming And Directory Interface (JNDI) of the J2EE Web Application Server instances running the product Web Application Server. This task can be performed as per the administration documentation supplied with the J2EE Web Application Server. The connection pool will be given a distinct name at definition time.

Note: This definition MUST be created prior to the use of this facility. See note below.

- At product installation time (or post installation) the configureEnv utility allows for the definition of **JDBC_NAME** which is either set to the name of the JNDI based connection pool created earlier.

Note: For Oracle WebLogic customers, specification of the JDBC pool will allow the installer to create the JDBC pool within Oracle WebLogic automatically.

Note: This configuration only affects the online and XAI components of the product. Other components will use the UCP connection pooling configuration regardless of this configuration.