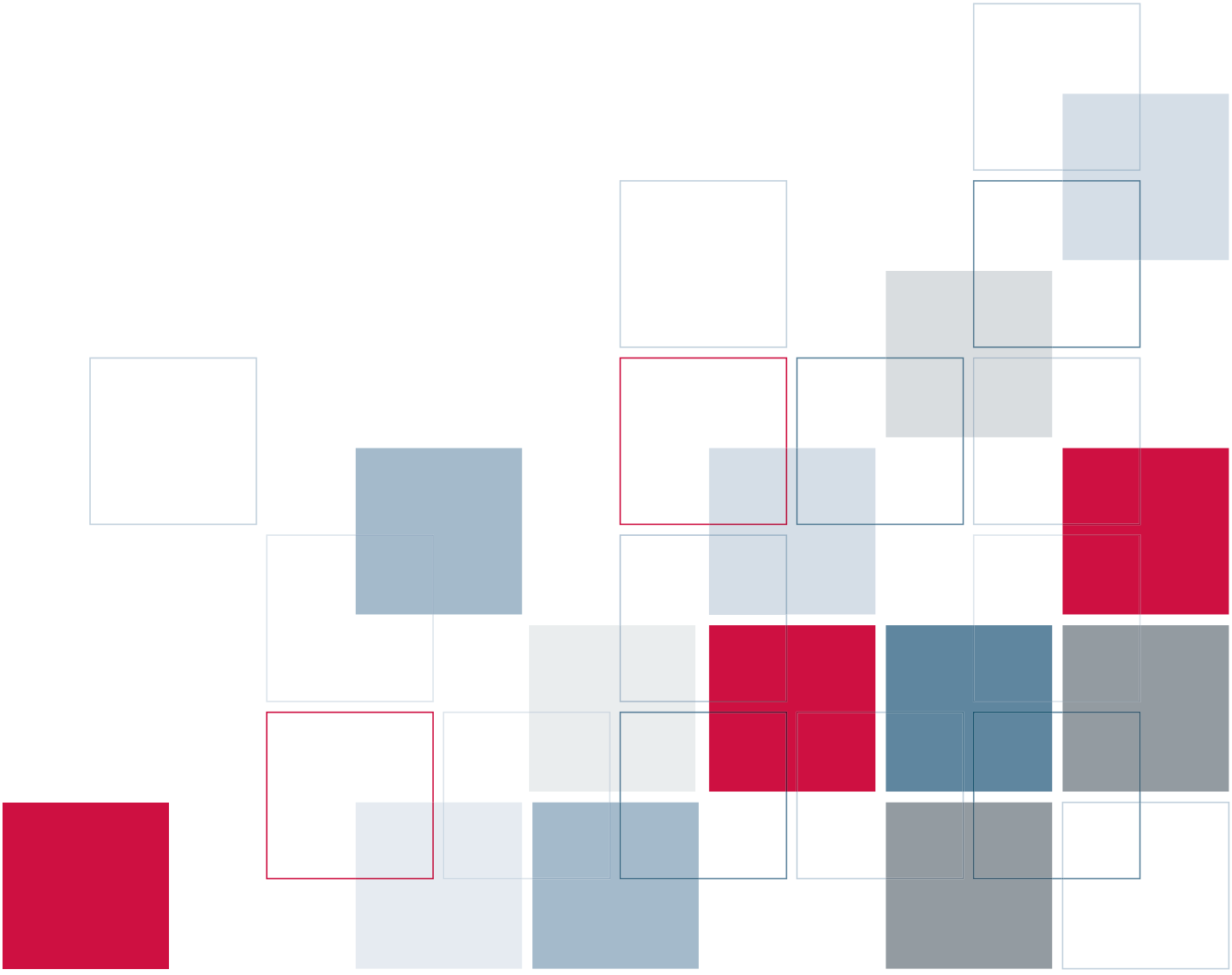
PASW® Collaboration and Deployment Services 4.1 Installation and Configuration Guide (Windows)



SPSS Inc.233 South Wacker Drive, 11th Floor
Chicago, IL 60606-6412
Tel: (312) 651-3000
Fax: (312) 651-3668

Printed in the United States of America.

Preface

PASW Collaboration and Deployment Services enable widespread use and deployment of predictive analytics with features like centralized, secure, and auditable storage of analytical assets, advanced capabilities for management and control of predictive analytic processes, as well as sophisticated mechanisms of delivering the results of analytical processing to the end users.

This manual documents the software and hardware requirements for PASW Collaboration and Deployment Services and the system installation and configuration. Tasks such as setting up content repository server, managing users, auditing the repository, etc. are documented in the *PASW Collaboration and Deployment Services 4.1 Administrator's Guide*. The tasks associated with everyday use of the analytical facilities of PASW Collaboration and Deployment Services are documented in *Deployment Manager 4.1 User's Guide*.

Technical Support

The services of SPSS Inc. Technical Support are available to registered customers of SPSS Inc.. Customers may contact Technical Support for assistance in using SPSS Inc. products or for installation help for one of the supported hardware environments. To reach Technical Support, see the SPSS Inc. Web site at *http://www.spss.com*, or contact your local office, listed on the SPSS Inc. Web site at *http://www.spss.com/worldwide*. Be prepared to identify yourself, your organization, and the serial number of your system.

Tell Us Your Thoughts

Your comments are important. Please let us know about your experiences with SPSS Inc. products. Please send e-mail to *suggest@us.ibm.com*, or write to SPSS Inc., Attn: Director of Product Planning, 233 South Wacker Drive, 11th Floor, Chicago IL 60606-6412.

Contents

Chapter 1

Overview

PASW Collaboration and Deployment Services

PASW Collaboration and Deployment Services is an enterprise-level application that enables widespread use and deployment of predictive analytics. PASW Collaboration and Deployment Services provides centralized, secure, and auditable storage of analytical assets and advanced capabilities for management and control of predictive analytic processes, as well as sophisticated mechanisms for delivering the results of analytical processing to the end users. The benefits of PASW Collaboration and Deployment Services include:

- safeguarding the value of analytical assets
- ensuring compliance with regulatory requirements
- improving the productivity of analysts
- minimizing the IT costs of managing analytics

PASW Collaboration and Deployment Services allows you to securely manage diverse analytical assets and fosters greater collaboration among those developing and using them. Furthermore, the deployment facilities ensure that the right people get the information they need to take timely, appropriate action.

Collaboration

Collaboration refers to the ability to share and reuse analytic assets efficiently, and is the key to developing and implementing analytics across an enterprise. Analysts need a location in which to place files that should be made available to other analysts or business users. That location needs a version control implementation for the files to manage the evolution of the analysis. Security is required to control access to and modification of the files. Finally, a backup and restore mechanism is needed to protect the business from losing these crucial assets.

To address these needs, PASW Collaboration and Deployment Services provides a repository for storing assets using a folder hierarchy similar to most file systems for organization. Files stored in the repository are available to users throughout the enterprise, provided those users have the appropriate permissions for access. To assist users in finding assets, the repository offers a search facility.

Analysts can work with files in the repository from client applications that leverage the service interface of PASW Collaboration and Deployment Services. Products such as PASW Statistics and PASW Modeler allow direct interaction with files in the repository. An analyst can store a version of a file in development, retrieve that version at a later time, and continue to modify it until it is finalized and ready to be moved into a production process. These files can include

custom interfaces that run analytical processes allowing business users to take advantage of an analyst's work.

The use of the repository protects the business by providing a central location for analytical assets that can be easily backed-up and restored. In addition, permissions at the user, file, and version label levels control access to individual assets. Version control and object version labels ensure the right versions of assets are being used in production processes. Finally, logging features provide the ability to track file and system modifications.

Deployment

To realize the full benefit of predictive analytics, the analytic assets need to provide input for business decisions. Deployment bridges the gap between analytics and action by delivering results to people and processes on a schedule or in real time.

In PASW Collaboration and Deployment Services, individual files stored in the repository can be included in processing **jobs** that define an execution sequence for the files. The execution results can be stored in the repository, on a file system, or delivered to specified recipients. Results stored in the repository can be accessed by any user with sufficient permissions using the Deployment Portal interface. The jobs themselves can be triggered according to a defined schedule or in response to system events.

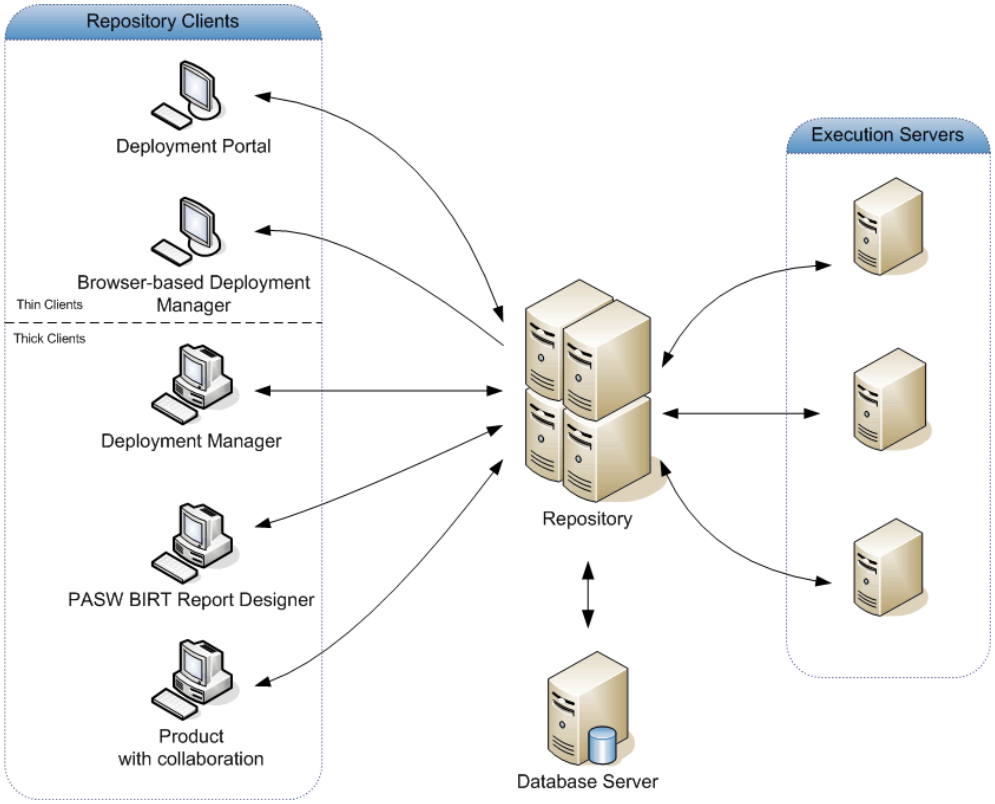
In addition, the scoring service of PASW Collaboration and Deployment Services allows analytical results from deployed models to be delivered in real time when interacting with a customer. An analytical model configured for scoring can combine data collected from a current customer interaction with historical data to produce a score that determines the course of the interaction. The service itself can be leveraged by any client application, allowing the creation of custom interfaces for defining the process.

The deployment facilities of PASW Collaboration and Deployment Services are designed to easily integrate with your enterprise infrastructure. Single sign-on reduces the need to manually provide credentials at various stages of the process. Moreover, the system can be configured to be compliant with Federal Information Processing Standard Publication 140-2.

System Architecture

In general, PASW Collaboration and Deployment Services consists of a single, centralized repository that serves a variety of clients, using execution servers to process analytical assets.

Figure 1-1
PASW Collaboration and Deployment Services Architecture



PASW Collaboration and Deployment Services consists of the following components:

- repository for analytical artifacts
- Product with Collaboration
- Deployment Manager
- Deployment Portal
- browser-based Deployment Manager
- Enterprise View
- PASW BIRT Report Designer

Repository

The repository provides a centralized location for storing analytical assets, such as models and data. The repository includes facilities for:

- Security
- Version control
- Searching
- Auditing

The repository requires an installation of a relational database, such as Oracle, IBM DB2 UDB, or Microsoft SQL Server.

Configuration options for the repository are defined using the Deployment Manager or the browser-based Deployment Manager. The contents of the repository are managed with the Deployment Manager and accessed with Deployment Portal.

Deployment Manager

Deployment Manager is a client application that allows users to schedule, automate, and execute analytical tasks, such as updating models or scores, using the repository. Deployment Manager allows a user to:

- View any existing files within the system, including reports, SAS syntax files, and data files.
- Import files into the repository.
- Schedule jobs to be executed repeatedly using a specified recurrence pattern, such as quarterly or hourly.
- Modify existing job properties in a user-friendly interface.
- Determine the status of a job.
- Specify e-mail notification of job status.

In addition, Deployment Manager allows users to perform administrative tasks for PASW Collaboration and Deployment Services, including:

- user management
- security provider configuration
- role and action assignment

Deployment Portal

Deployment Portal is a thin-client interface for accessing the repository. Unlike the browser-based Deployment Manager, which is intended for PASW Collaboration and Deployment Services administrators, Deployment Portal is a web portal serving a variety of users. Deployment Portal includes the following functionality:

- Browsing the repository content by folder
- Opening published content
- Running jobs and reports
- Generating scores using models stored in the repository
- Searching repository content.
- Viewing content properties.
- Accessing individual user preferences, such as e-mail address and password, general options, subscriptions, and options for output file formats.

Browser-based Deployment Manager

The browser-based Deployment Manager is a thin-client interface for performing setup and system management tasks, including:

- Configuring the system.
- Configuring security providers.
- Managing MIME types.

Non-administrative users can perform any of these tasks provided they have the appropriate actions associated with their login credentials. The actions are assigned by an administrator.

Enterprise View

Enterprise View provides a single, consistent view of enterprise data. Enterprise View allows users to define and maintain a common view of warehoused and transaction data needed to perform analytics, optimization, deployment, and reporting. Underlying data may come from a variety of sources, including a data warehouse, an operational data store, and an online transaction database. Enterprise View ensures a consistent use of enterprise data and hides the complexities of stored data structures from the end user. Enterprise View is the data backbone for the predictive enterprise.

Data discovery requires a major investment of resources from the organizations deploying predictive analytics. The process is labor intensive—it can involve representatives from departments across the organization and often entails resolving differences in data structure and semantics across organizational boundaries. Enterprise View provides a mechanism for recording the outcomes of the data discovery process, versioning and securing the resulting schema, and tracking changes over time.

Enterprise View includes the Enterprise View Driver component designed to provide other applications access to Enterprise View objects stored in the repository. The driver operates similarly to ODBC drivers with the exception that it does not directly query a physical data source but rather references Enterprise View data provider definitions and application views. Note that while Enterprise View is installed as part of Deployment Manager, Enterprise View Driver must be installed separately. For more information, see the installation instructions.

Execution Servers

Execution servers provide the ability to execute resources stored within the repository. When a resource is included in a job for execution, the job step definition includes the specification of the execution server used for processing the step. The execution server type depends on the resource.

Execution servers currently supported by PASW Collaboration and Deployment Services include:

- **SAS**. The SAS execution server is the SAS executable file *sas.exe*, included with Base SAS® Software. Use this execution server to process SAS syntax files.
- **Remote Process**. A remote process execution server allows processes to be initiated and monitored on remote servers. When the process completes, it returns a success or failure message. Any machine acting as a remote process server must have the necessary infrastructure installed for communicating with the repository.

Execution servers that process other specific types of resources can be added to the system by installing the appropriate adapters. For information, consult the documentation for those resource types.

During job creation, assign an execution server to each step included in the job. When the job executes, the repository uses the specified execution servers to perform the corresponding analyses.

PASW BIRT Report Designer

The reporting functionality of PASW Collaboration and Deployment Services is enabled by BIRT (Business Intelligence and Reporting Tools), an open-source package distributed by Eclipse Foundation under the Eclipse Public License. BIRT provides core reporting features, such as report layout, data access, and scripting. For more information about BIRT, see the BIRT project page (*http://www.eclipse.org/birt*).

The PASW Collaboration and Deployment Services installation includes the BIRT reporting engine server components, which enable the execution of BIRT report syntax files as part of the PASW Collaboration and Deployment Services reporting job steps. PASW BIRT Report Designer is a standalone application that can be used in conjunction with PASW Collaboration and Deployment Services. It provides a rich user interface with a number of advanced features for creating reports and must be installed separately.

If a PASW BIRT Report Designer report requires a JDBC-based database connection, a corresponding JDBC driver must be installed with the repository. For application server-specific information on the location of the JDBC drivers, see the corresponding section of the repository installation instructions.

To start PASW BIRT Report Designer, execute the file *BIRT.exe* in the installation directory. For information on using PASW BIRT Report Designer, see the documentation installed with the application.

Products with Collaboration

A product with collaboration allows interaction with the repository from within the native interface. Files can be stored and retrieved directly from the collaborating product.

In addition, some files stored in the repository can be executed as steps within jobs. A job can contain any number of steps, with each step corresponding to a separate file. Relationships defined between the steps determine the processing flow. The job can be scheduled to execute at a specific time, according to a recurrence pattern, or in response to a defined event. Moreover, notifications can be sent to specified recipients to report on individual step and overall job execution status.

Collaboration between PASW Collaboration and Deployment Services and other products is enabled through the use of adapters. These adapters are installed into the PASW Collaboration and Deployment Services environment to add the product-specific features. For more information, consult the collaborating product documentation.

Chapter 2

What's New in This Release?

New in Release 4.1

PASW Collaboration and Deployment Services 4.1 introduces the following new features and enhancements that impact system configuration:

Support has been added for the most current versions of third-party components (operating systems, application servers, databases, etc.) including:

- Windows Server 2008 and Windows Server 2008 R2
- IBM WebSphere 7
- IBM DB2 9.7 for the repository database
- Java 6 (for certain application servers)
- Oracle 11g R2 for the repository database
- Oracle WebLogic 11g
- JBoss 5.1

PASW Collaboration and Deployment Services clustering has been enhanced to be more reliable and scalable. In order to simplify the management of PASW Collaboration and Deployment Services in a clustered environment, the system now allows deployment of PASW Collaboration and Deployment Services applications as a single EAR file. Additionally, PASW Collaboration and Deployment Services 4.1 can be configured for job step failover in the event of cluster node failure.

Chapter
3

Installation and Configuration

This chapter provides the information about the installation and configuration of PASW Collaboration and Deployment Services repository. Configuration of the repository environment may consist of:

Provisioning. Certain prerequisites must be in place before beginning the installation. This includes verifying certain hardware and software requirements are met, setting up database connections and tables, and determining the installation directory of the application server PASW Collaboration and Deployment Services will use for distributed access.

Installing. New users must perform a clean installation of the repository in Windows, UNIX, or IBM i environment.

Upgrading. Users with an existing version of the repository can conveniently upgrade their environment to take advantage of new features and functions.

Uninstalling. In the event that an installation becomes corrupt or the application needs to be reinstalled due to system errors, the repository can be removed and the system restored to its original state.

When finished, verify the installation is successful and install Deployment Manager on client workstations that will connect to the repository.

Provisioning the System

Prior to installing the repository, verify that the necessary application server, database configuration, hardware, software, and permissions requirements have been met.

Hardware Requirements

The following hardware requirements must be met before installing the repository. Note that this does not reflect the hardware requirements of software beyond the repository, such as operating systems and databases.

Table 3-1
Hardware requirements

Component	Requirement
Processor	At least Pentium 1.8 GHz
Hard Drive	At least 5 GB of free space
Memory	At least 4 GB RAM
Optical drive	DVD-ROM

Software Requirements

Server operating systems

The repository can be installed into application servers running on the following operating system(s):

Operating System	Edition	Release	Processor	Word-size
Windows Server		2008 R2	x86	32-bit
Windows Server		2008 R2	EM64T, AMD64	64-bit
Windows Server		2008	x86	32-bit
Windows Server		2008	EM64T, AMD64	64-bit
Windows Server	Standard	2003 R2	EM64T, AMD64	64-bit
Windows Server	Standard	2003	x86	32-bit
Windows Server	Standard	2003	EM64T, AMD64	64-bit

Client operating systems

Repository desktop client applications, such as Deployment Manager, can run on the following operating systems.

OS	Release	Edition	Processor	Required Patch Level
Windows	7	Enterprise	x86	
Windows	7	Professional	x86	
Windows	7	Enterprise	x64 (32-bit code)	
Windows	7	Professional	x64 (32-bit code)	
Windows	7	Enterprise	x64 (64-bit code)	
Windows	7	Professional	x64 (64-bit code)	
Windows	Vista	Enterprise	x86	SP1
Windows	Vista	Business	x86	SP1
Windows	Vista	Enterprise	x64 (32-bit code)	SP1
Windows	Vista	Business	x64 (32-bit code)	SP1
Windows	Vista	Enterprise	x64 (64-bit code)	SP1
Windows	Vista	Business	x64 (64-bit code)	SP1
Windows	XP	Pro	x86	SP3
Windows	XP	Pro	x64 (64-bit code)	SP3
Windows	XP	Pro	x64 (32-bit code)	SP3

Web browsers

PASW Collaboration and Deployment Services 4.1 Web applications can be accessed by the following browsers.

Browser	Release	Windows 7	Vista	XP	Desktop Linux	Mac OSX
Internet Explorer	IE 8	Supported	Supported	Supported	Not Supported	Not Supported

Browser	Release	Windows 7	Vista	XP	Desktop Linux	Mac OSX
Internet Explorer	IE 7	Not supported	Supported	Supported	Not Supported	Not Supported
Mozilla Firefox	3.x	Required	Supported	Supported	Supported	Supported
Mozilla Firefox	2.x	Requirws	Supported	Supported	Supported	Supported
Apple Safari	3.x	Not supported	Not Supported	Not Supported	Not Supported	Required

Other requirements

Other software requirements include:

- A JVM appropriate to the application server selected for the installation. For more information, see application server vendor documentation.

File System Permissions

The user installing PASW Collaboration and Deployment Services must have the following permissions on the host system:

- Write permissions to the PASW Collaboration and Deployment Services installation directory and subdirectories.
- Execute permissions for all library files under *<PASW Collaboration and Deployment Services Installation Directory>/components/setup/jni/[win64|windows]*.
- Write permissions to the deployment and configuration directories and read and execute permissions to other application server directories.

Application Servers

Before installing the repository, a supported J2EE application server or a server cluster must be installed and accessible. The repository installation requires a connection to the application server to deploy the necessary Web services and components. If the repository is reinstalled, it is strongly recommended to use a new instance of the application server. It is also essential to make sure the latest versions of vendor patches have been applied to application server installations.

The following table lists supported application servers.

Application Server	Operating System	Java Environment					
		Sun	JRockit (BEA/Oracle)	HP-UX	IBM	Azul	SAP
JBoss 5.1	Windows 2003 Server	1.6					
	Windows 2008 Server	1.6					
JBoss 4.2.x (AP 4.3)	Windows 2003 Server	1.5	1.5	1.5			
	Windows 2008 Server	1.5	1.5				

Application Server	Operating System	Java Environment					
		Sun	JRockit (BEA/Oracle)	HP-UX	IBM	Azul	SAP
Redhat Enterprise Application Platform 4.3	Windows 2003 Server	1.5, 1.6	1.5				
	Windows 2008 Server	1.5, 1.6	1.5				
IBM WebSphere 7	Windows 2003 Server				1.6		
	Windows 2008 Server				1.6		
IBM WebSphere 6.1	Windows 2003 Server				1.5		
	Windows 2008 Server				1.5		
Oracle WebLogic 11g	Windows 2003 Server	1.6	1.6				
	Windows 2008 Server	1.6	1.6				
Oracle WebLogic 10	Windows 2003 Server	1.5	1.5				
	Windows 2008 Server	1.5	1.5				
SAP NetWeaver 7.1	Windows 2008 Server						1.5
	Windows 2008 Server						1.5

Whether or not the application server should be running during installation depends on the server.

- For deployment into JBoss, the application server should not be running.
- For deployment into WebLogic, the application server should not be running.
- For deployment into WebSphere, the application server should be running.
- WebSphere 7 requires Fix Pack 5 to be applied.
- Configuring single sign-on for PASW Collaboration and Deployment Services running on WebSphere 6.1 requires Patch 19.

Notes:

- For JBoss application server, it is recommended that only one instance of the server be run. If multiple instances of JBoss application server to be used with the repository must be set up on a single machine, consult vendor documentation.
- To prevent remote attacks on PASW Collaboration and Deployment Services instance running on JBoss through Java Management Extension (JMX) Console, uncomment *security-constraint* bloc in *<JBoss home>/WEB-INF/web-xml*. For more information, see JBoss JMX Console documentation.
- If WebLogic application server is used with JRockit JVM, JAVA_VENDOR parameter in *<BEA_HOME>/user_projects/domains/<domainname>/startWebLogic.bat* must be set to Oracle.

- To avoid errors at PASW Collaboration and Deployment Services startup, it is recommended that JBoss 5.1 application server installation path not contain any spaces, for example, as in *C:\Program Files\jboss5.1.0.GA*.
- For details about NetWeaver configuration following PASW Collaboration and Deployment Services installation, see Appendix A.

For additional information on installing an application server, refer to the vendor documentation.

Databases

Before installing the repository, a database must be running and accessible. Repository installation requires a connection to the database to establish the necessary control tables and infrastructure. The following table lists supported repository databases.

Vendor	Database	Release	Version
IBM	DB2 Enterprise	9.7	32-bit
IBM	DB2 Enterprise	9.7	64-bit
IBM	DB2 Enterprise	9.5	32-bit
IBM	DB2 Enterprise	9.5	64-bit
IBM	DB2 Enterprise	9.1	32-bit
IBM	DB2 Enterprise	9.1	64-bit
IBM	DB2/400	v6r1	(embedded in OS)
IBM	DB2/400	v5r4	(embedded in OS)
Oracle	Oracle Database	11g R2	32-bit
Oracle	Oracle Database	11g R2	64-bit
Oracle	Oracle Database	11g (11.0)	32-bit
Oracle	Oracle Database	11g (11.0)	64-bit
Oracle	Oracle Database	10g (10.2)	32-bit
Oracle	Oracle Database	10g (10.2)	64-bit
Microsoft	SQL Server	2008	32-bit
Microsoft	SQL Server	2008	64-bit
Microsoft	SQL Server	2005	32-bit
Microsoft	SQL Server	2005	64-bit

The database and the repository do not need to be installed on the same server, but some configuration information is necessary to ensure connectivity. During the installation, you will be prompted for the database server name, port number, user name and password, and the name of the database to use for information storage and retrieval.

Important! With databases other than DB2 on IBM i , you must manually create the database prior to installation. Any valid database name can be used, but if a previously created database does not exist, the installation will not continue.

Notes:

- For Oracle database, Oracle XDB (XML database feature) must be installed. You can determine that by verifying that schema (user account) *XDB* exists (SELECT * FROM ALL_USERS;), or that view *RESOURCE_VIEW* exists (DESCRIBE RESOURCE_VIEW).
- For DB2 IBM i , DB2 XML Extender package must be enabled.

Database Permissions

The user must also have the following general permissions to the database to perform the install and initial startup of PASW Collaboration and Deployment Services:

- Create session
- Create table
- Drop table
- Create view
- Drop view
- Create function
- Create procedure
- Select
- Insert
- Update
- Delete
- Execute procedure

The exact names of these permissions vary depending on the database type. Also depending on the database, some additional permissions may be needed. For example, Oracle also requires an explicit CONNECT and CREATE INDEX permissions; MS SQL Server requires a REFERENCES permission.

Oracle Database Configuration

When using an Oracle 10g or 11g database in conjunction with PASW Collaboration and Deployment Services, the following parameters and configurations must be followed. Changes are made to the *init.ora* and *spfile.ora* parameter files.

Table 3-2
Oracle Database Parameters

Parameter	Setting
OPEN_CURSORS	300
NLS_CHARACTERSET	AL32UTF8
NLS_NCHAR_CHARACTERSET	AL16UTF16

Note: Both NLS_CHARACTERSET and NLS_NCHAR_CHARACTERSET should be set when creating the Oracle instance.

DB2 Configuration

When using a non-IBM i DB2 UDB database in conjunction with PASW Collaboration and Deployment Services, the default database creation parameters are not sufficient. The following parameters and configurations must be followed:

- Use a UTF-8 codeset.
- Use an 8K buffer pool (*SPSSEIGHT*).
- Include the two tablespaces, *SPSSEIGHT* and *SPSSLARGE*.
- Create a temporary system tablespace.

An example script for creating a database named *SPSSPLAT* follows:

```
CREATE DATABASE SPSSPLAT ON C: USING CODESET UTF-8 TERRITORY US COLLATE USING SYSTEM;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING APP_CTL_HEAP_SZ 128;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING APPGROUP_MEM_SZ 10715;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING APPLHEAPSZ 32000;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING CATALOGCACHE_SZ 32000;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING CHNGPGS_THRESH 60;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING DBHEAP 600;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING MAXLOCKS 60;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING LOCKLIST 50;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING LOGBUFSZ 131;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING LOGFILSIZ 1024;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING SOFTMAX 300;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING LOGPRIMARY 3;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING USEREXIT YES;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING LOGSECOND -1;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING MAXAPPLS 100;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING MINCOMMIT 1;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING NUM_IOCLEANERS 1;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING NUM_IOSERVERS 2;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING PCKCACHESZ 859;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING STMTHEAP 5000;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING DFT_DEGREE 1;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING DFT_PREFETCH_SZ 32;
UPDATE DATABASE CONFIGURATION FOR SPSSPLAT USING UTIL_HEAP_SZ 37991;
UPDATE DATABASE MANAGER CONFIGURATION USING SHEAPTHRES 14113;
UPDATE DATABASE MANAGER CONFIGURATION USING INTRA_PARALLEL OFF;
UPDATE DATABASE MANAGER CONFIGURATION USING MAX_QUERYDEGREE 1;
UPDATE DATABASE MANAGER CONFIGURATION USING MAXAGENTS 200;
UPDATE DATABASE MANAGER CONFIGURATION USING NUM_POOLAGENTS 200;
UPDATE DATABASE MANAGER CONFIGURATION USING NUM_INITAGENTS 0;
UPDATE DATABASE MANAGER CONFIGURATION USING FCM_NUM_BUFFERS 1024;
UPDATE DATABASE MANAGER CONFIGURATION USING PRIV_MEM_THRESH 32767;
CONNECT TO SPSSPLAT;
ALTER BUFFERPOOL IBMDEFAULTBP SIZE 113974;
SET CURRENT QUERY OPTIMIZATION = 5;
COMMIT;
CONNECT RESET;
BACKUP DATABASE SPSSPLAT TO "C:\Temp" WITH 2 BUFFERS BUFFER
1024 PARALLELISM 1 WITHOUT PROMPTING;
```

```
CONNECT TO SPSSPLAT;
CREATE Bufferpool SPSSEIGHT IMMEDIATE SIZE 250 PAGESIZE 8 K ;
CREATE Bufferpool SPSSTEMP IMMEDIATE SIZE 250 PAGESIZE 32 K ;
CREATE REGULAR TABLESPACE SPSSEIGHT PAGESIZE 8 K MANAGED BY SYSTEM
USING ('C:\DB2\NODE0000\SPSSEIGHT' ) EXTENTSIZE 16 OVERHEAD 10.5 PREFETCHSIZE 16
TRANSFERRATE 0.14 BUFFERPOOL "SPSSEIGHT" DROPPED TABLE RECOVERY OFF;
COMMENT ON TABLESPACE SPSSEIGHT IS '';
CREATE LARGE TABLESPACE SPSSLARGE PAGESIZE 8 K MANAGED BY DATABASE
USING ( FILE 'C:\DB2\NODE0000\SPSSLARGE' 2560 ) EXTENTSIZE 16 OVERHEAD 10.5
PREFETCHSIZE 16 TRANSFERRATE 0.14 BUFFERPOOL "SPSSEIGHT";
COMMENT ON TABLESPACE SPSSLARGE IS '';
CREATE SYSTEM TEMPORARY TABLESPACE SPSSTEMP PAGESIZE 32 K MANAGED BY AUTOMATIC STORAGE
EXTENTSIZE 16 OVERHEAD 10.5 PREFETCHSIZE 16 TRANSFERRATE 0.14 BUFFERPOOL "SPSSTEMP";
COMMENT ON TABLESPACE SPSSTEMP IS '';
CONNECT RESET;
```

The optimal STMTHEAP value, which controls how much memory is available to DB2 to process each submitted SQL statement, depends on the version of DB2 being used. If the system reports problems with long or complex SQL statements, increase the STMTHEAP value to allow proper processing.

Microsoft SQL Server Configuration

Appropriate options must be used when setting up the database for processing non-Latin character sets. For example, it is recommended to use the Kana-sensitive (_KS) option to distinguish between Hiragana and Katakana Japanese characters. For more information about database collation, refer to Microsoft SQL Server documentation.

SPSS Inc. Products Compatibility

The system is compatible with the following versions of SPSS Inc. applications.

Table 3-3
Supported versions of SPSS Inc. applications

SPSS Inc. Product	**Version**
PASW Modeler	13, 14
PASW Statistics	17, 18
PASW Decision Management	6
PASW Data Collection	5.6, 6

PASW Statistics client, PASW Modeler client, and ShowCase Suite client are not required for use of PASW Collaboration and Deployment Services. However, these applications offer interfaces for using the repository to store and retrieve objects. The server versions of these products are required if jobs containing PASW Statistics syntax, PASW Modeler streams, or ShowCase files/sets will be executed.

Notes:

- PASW Collaboration and Deployment Services is backward-compatible with SPSS 16.0 and Clementine 12.0
- By default, the repository is installed without content repository adapter and process manager packages for PASW Modeler and PASW Modeler users must install the content repository adapter packages corresponding to their version of PASW Modeler. PASW Modeler 13.0 packages can be found on the PASW Modeler distribution disk and installed with PASW Collaboration and Deployment Services Package Manager utility. For more information, see the topic Updating the Repository in Chapter 11 on p. 104.

Virtualization

PASW Collaboration and Deployment Services server or client components can be deployed into virtualized environments provided by third-party software. For example, in order to simplify deployment of a PASW Collaboration and Deployment Services development and testing environment, a system administrator can configure a virtual server on which to install the repository. The virtual machines hosting PASW Collaboration and Deployment Services components must meet minimum system requirements. For more information, see the topic Provisioning the System on p. 9.

Table 3-4
Supported virtualized environments

Vendor	Product	Version	Edition	Server or Client Virtualization
VMWare	VSphere	4.0		Server
VMWare	ESXServer	3.5		Server
Microsoft	Windows Terminal Services	Windows 2008 Server		Client
Microsoft	Windows Terminal Services	Windows 2003 R2 Server		Client
Microsoft	Windows Terminal Services	Windows 2003 Server		Client
Citrix	XenApp	5.0	Enterprise	Client
Citrix	XenApp	5.0	Advanced	Client
Citrix	XenApp	5.0	Standard	Client
Citrix	Presentation Server	4.5	Enterprise	Client
Citrix	Presentation Server	4.5	Advanced	Client
Citrix	Presentation Server	4.5	Standard	Client

Assuming that the configured virtualized environment meets the minimum system requirements, no performance degradation PASW Collaboration and Deployment Services server or client installations is expected. It is important to note, however, that virtualized systems might share available physical resources, and resource contention on systems with a heavy processing load can cause performance degradation of the hosted PASW Collaboration and Deployment Services installations.

Installing the Repository

The installation involves:

1. Copying the necessary files from the distribution disk to the target computer.
2. Deploying the repository into an application server for general use and configuring the database. Deployment is performed by the setup utility.

   This can be accomplished by using either the graphical installation wizard or a command line equivalent. Environments without a graphical interface must use the command line approach.

Graphical Installation Wizard

1. Execute the program to start the installation wizard. The file is located in the */PASW/Disk1/InstData/<OS Name>/NoVM/* directory of Disk 1.

   ```
   install.exe
   ```

   Note: When performing the installation on Windows Server 2008 R2, you must specify the path to the Java executable as LAX_VM parameter, for example:

   ```
   install.exe LAX_VM "C:\Program Files\Java\jre1.6.0_05\bin\java.exe"
   ```

2. After the installation wizard is launched, follow the instructions that appear on the screen. Setup utility for deploying the repository will be launched automatically after the initial installation tasks are completed. For more information, see Setup on p. 19

   Notes:

   - The path of the installation directory cannot contain extended ASCII characters.
   - The of the JVM for the repository installation must point to JVM used by the application server.

Command Line Installation

Execute the program with the console command line switch to start the command line installation wizard. The program file is located in the */PASW/Disk1/InstData/<OS Name>/NoVM* directory of Disk 1.

```
install.exe -i console
```

Follow the instructions that appear on the screen. When the initial installation is completed, the setup utility must be launched to deploy PASW Collaboration and Deployment Services files into the application server and configure the repository database. For more information, see the topic Setup on p. 19.

Notes:

- The path of the installation directory cannot contain extended ASCII characters.
- The of the JVM for the repository installation must point to JVM used by the application server.

Setup

The repository setup deploys the installation files into the application server, modifies the application server settings, and configures the repository database after the initial installation is completed.

Setup must be run in the following cases:

- Initial PASW Collaboration and Deployment Services installation.
- Migration to different hardware. For more information, see the topic Migration in Chapter 4 on p. 33.
- Migration to a different application server or database.
- Upgrade to a different version of PASW Collaboration and Deployment Services. For more information, see the topic Upgrading Repository on p. 31.
- Master database password change. For more information, see the topic Changing Master Database Password on p. 30.

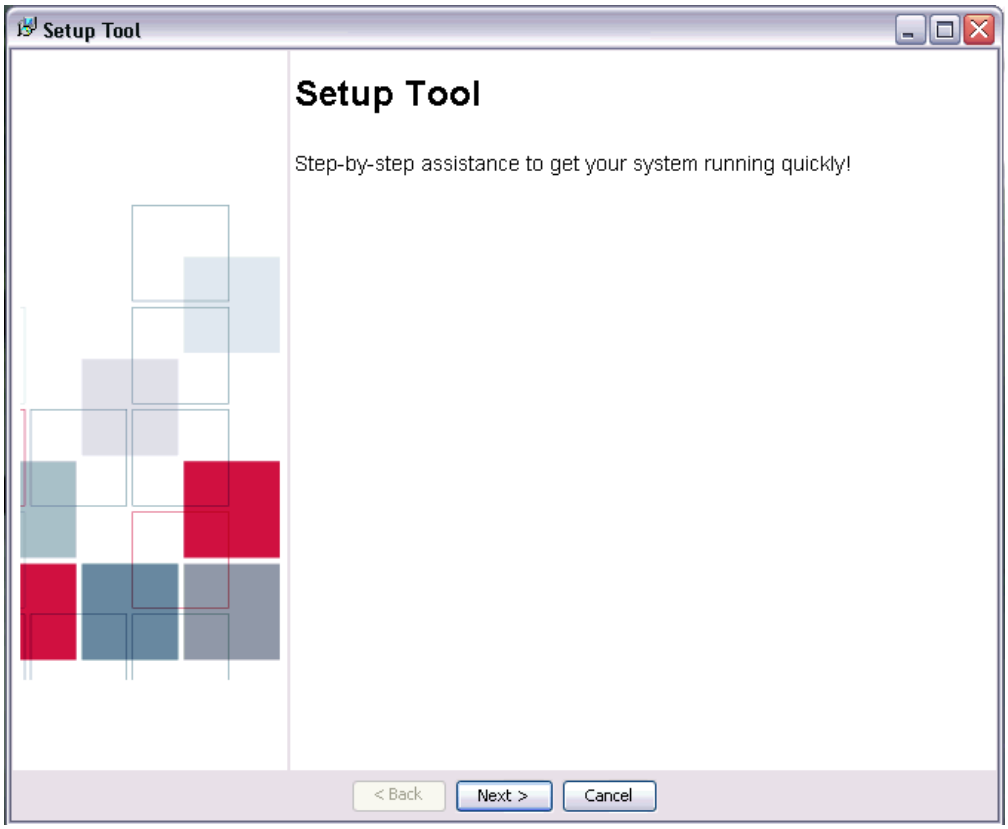
For an already existing PASW Collaboration and Deployment Services installation, any customizations made to the application server settings, such as Java option, memory setting, etc., will be overwritten by setup. In order to preserve customizations, the application server configuration files must be backed up.

The setup utility is launched automatically when the repository installation is run in GUI mode. When the repository is installed in command line mode, setup must be launched manually.

Setup with a Graphical User Interface

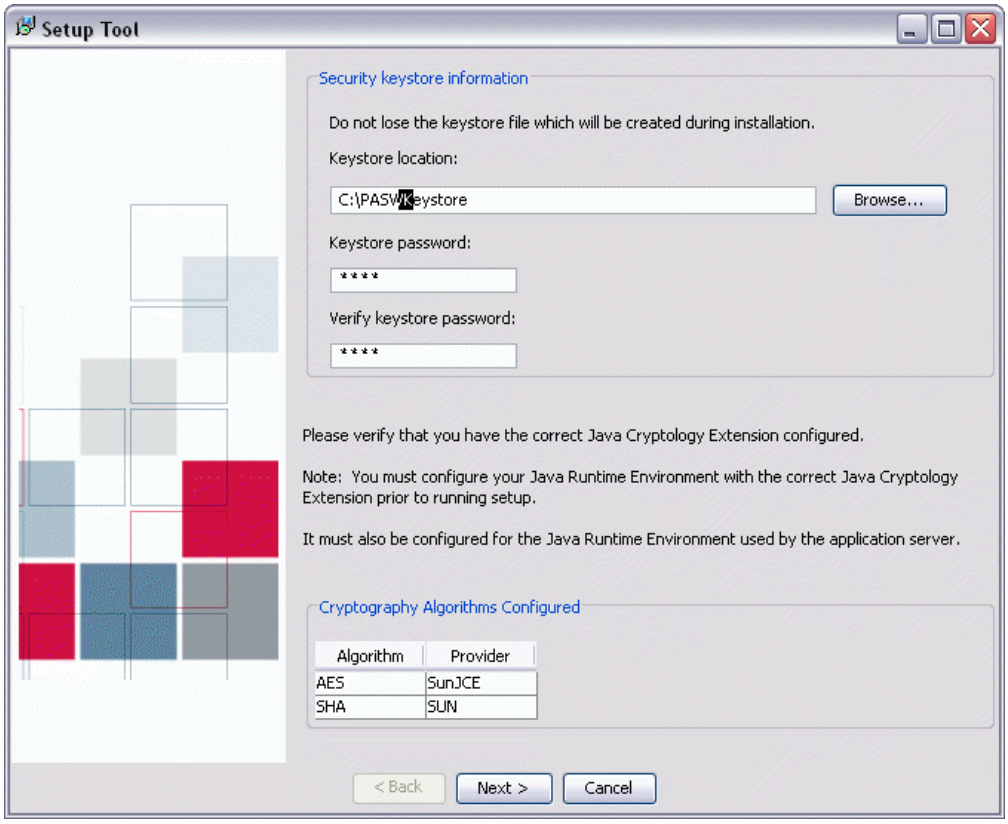
1. To manually launch setup, execute the script in *<PASW Collaboration and Deployment Services installation directory>/setup/*: setup.bat.
   The welcome screen is displayed.

Figure 3-1
Setup welcome screen



To proceed with the setup, click Next. Security keystore information screen appears.

Figure 3-2
Specify keystore location and password and FIPS 140-2 compliance level



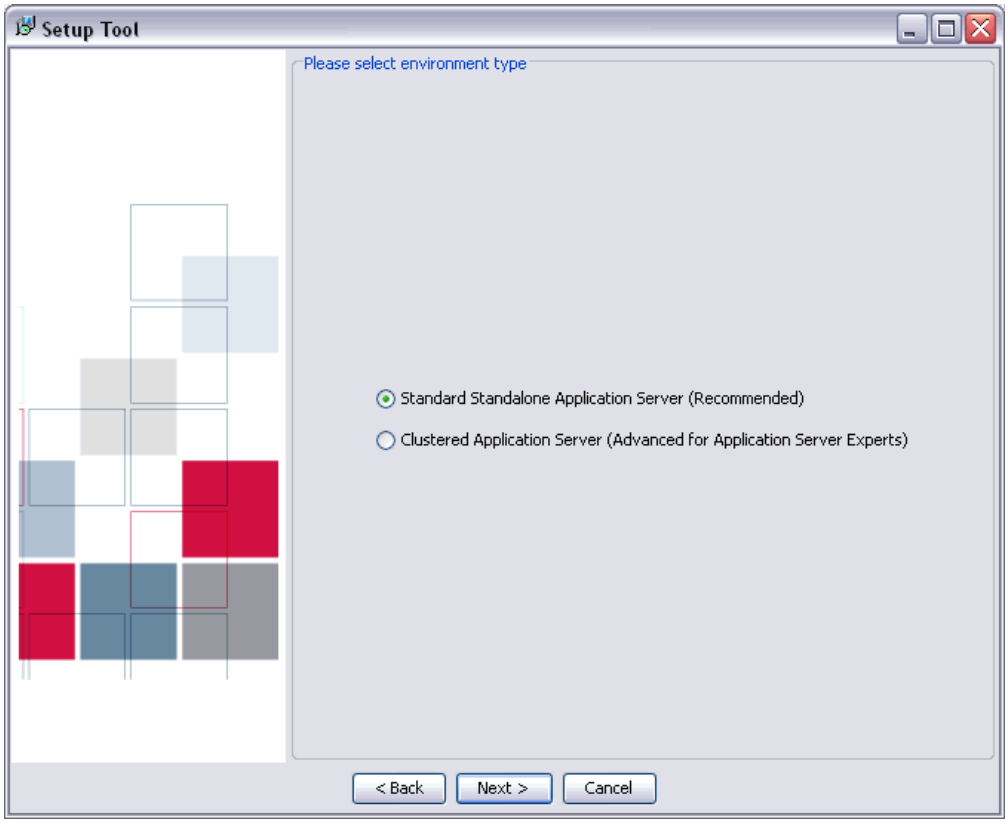
2. Specify the keystore location and then specify and confirm the password for accessing the keystore. The keystore is an encrypted file that contains the key for decrypting the passwords used by the repository, such as the repository administration password, the database access password, etc.

   Important! If the keystore file is lost, none of the passwords can be decrypted and the system becomes unusable and must be reinstalled. Therefore, it is recommended that backup copies of the keystore file be maintained.

   The available encryption algorithm will be listed in the table. If no algorithms are listed, you must exit the setup, configure the encryption modules for your Java runtime environment, and then restart the setup. For more information, see your JVM vendor documentation.

3. Click Next. Select Environment Type screen appears.

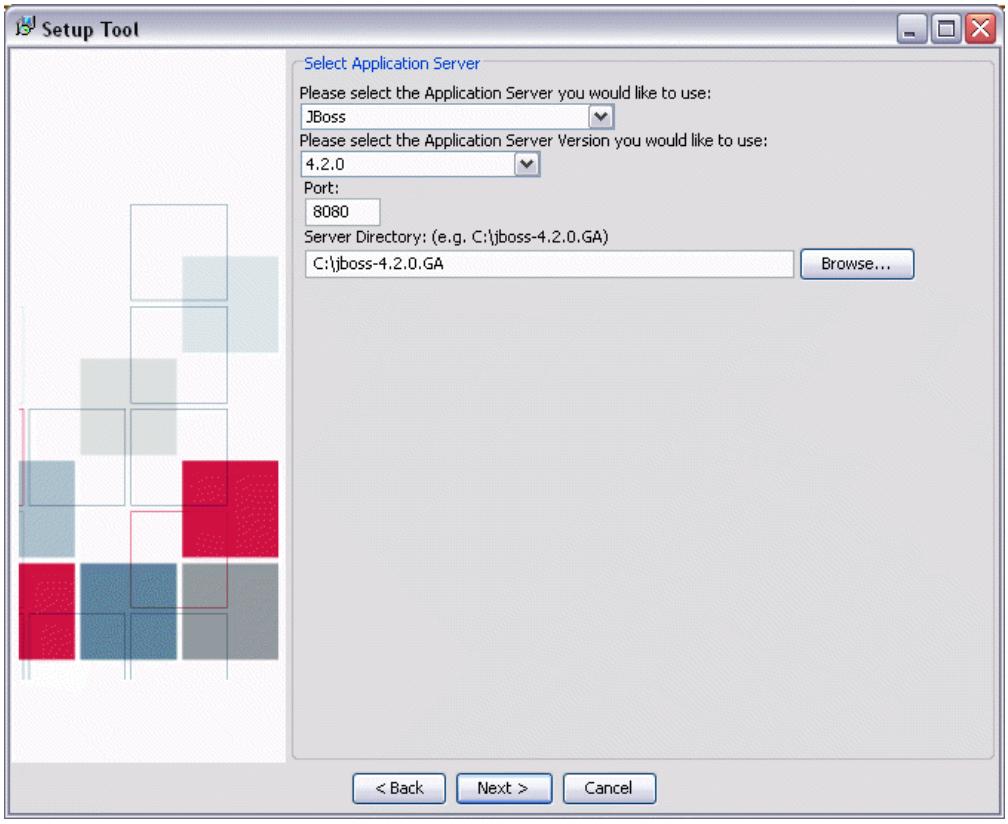
Figure 3-3
Select Environment Type



Select the environment type for PASW Collaboration and Deployment Services installation: standalone application server or application server cluster.

4. Click Next. Select Application Server screen appears.

Figure 3-4
Select Application Server



If you are installing PASW Collaboration and Deployment Services into a an application server cluster, the following parameters must be specified:

- **Application Server.** The application servers in the cluster, for example, WebSphere or WebLogic.
- **Version.** The version of the application servers in the cluster.
- **Output Directory.** The directory on the local file system where PASW Collaboration and Deployment Services file will be installed. The location must be accessible to all servers in the cluster, for example as a mapped or mounted disk drive.
- **Output Directory.** The directory where PASW Collaboration and Deployment Services file will be installed. The location must be accessible to all servers in the cluster, for example as a mapped or mounted disk drive.
- **Load Balancer URL.** The address of the load balancer. Users will be accessing PASW Collaboration and Deployment Services at this address.
- **Cluster Name.** The name of the application server cluster.
- **Secure HTTP/SOAP Communication Between Components.** Specifies that communication between nodes in the cluster will be secure.

Important! Deploying PASW Collaboration and Deployment Services into an application server cluster includes a number of additional configuration steps. For more information, see the topic Clustering in Chapter 6 on p. 46.

If you have chosen a standalone application server installation, specify configuration parameters for the application server. The parameters needed depend on the application server. Select Manual to deploy the repository into the application server yourself. For this option, the installation creates an output directory in the specified location containing the files to be deployed and a *readme.txt* file containing instructions for manual deployment. Manual deployment should only be attempted by J2EE application server experts.

JBoss

- **Port.** The port number on which the application server runs.
- **Server Directory.** The installation location of the application server.

WebLogic

- **Port.** The port number on which the application server runs.
- **Server Directory.** The installation location of the application server.
- **Domain Location.** The directory location of the WebLogic domain.
- **Domain Name.** The name of the domain.
- **Server Name.** The name of WebLogic server.
- **Server Admin User ID.** Administrative login for the application server.
- **Server Admin Password.** Password associated with the specified application server administrative login.

Note: The domain and the server must be created prior to PASW Collaboration and Deployment Services installation.

WebSphere

- **Port.** The port number on which the application server runs.
- **Profile Directory.** The directory where WebSphere profile is stored, for example, *C:\Program Files\IBM\WebSphere\AppServer\profiles\ProfileName*.
- **Server Admin User ID.** Administrative login for the application server.
- **Server Admin Password.** Password associated with the specified application server administrative login.
- **WebSphere SOAP Connector Address Port.** The port number used by WebSphere for incoming SOAP requests via HTTP.
- **Server Name** The name of the WebSphere server.
- **Node.** The name of the WebSphere node on which to install.
- **Cell.** The WebSphere cell containing the node.

NetWeaver

- **Port.** The port number on which the application server runs.

- **Server Directory.** The installation location of the application server.
- **Server Admin User ID.** Administrative login for the application server.
- **Server Admin Password.** Password associated with the specified application server administrative login.
- **SID.** System identifier of the SAP server.
- **Instance Number** System identifier of the SAP instance.
- **P4 Port.** The port number for P4 (RMI) access.

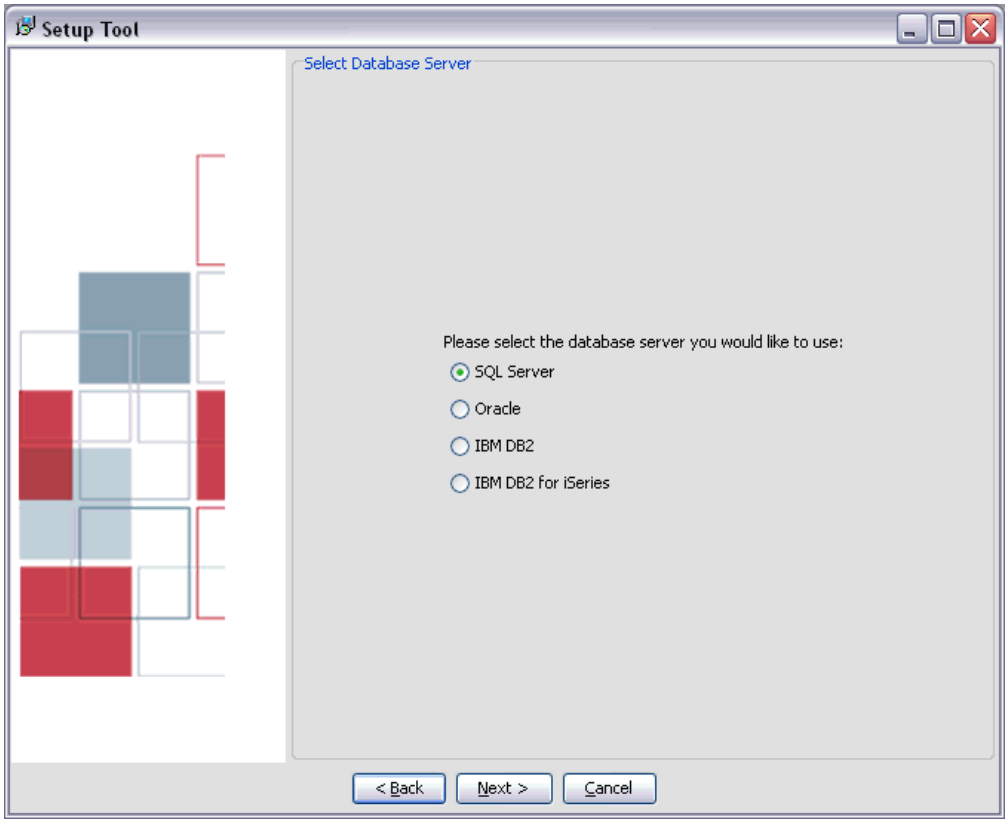
Manual (for expert J2EE server users)

- **Port.** The port number on which the application server runs.
- **Output Directory.** The directory where PASW Collaboration and Deployment Services file will be installed. The location must be accessible to all servers in the cluster, for example as a mapped or mounted disk drive.

For more information about the parameters, consult application server vendor documentation.

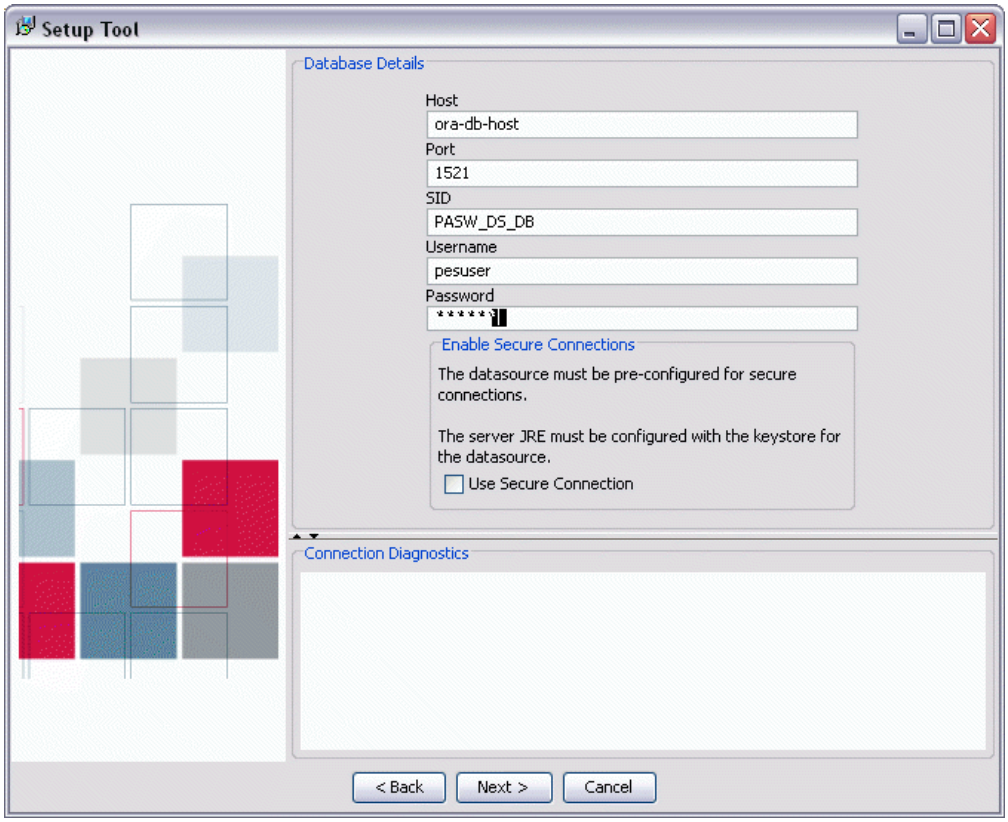
5. Click Next. The Select Database Server screen appears.

Figure 3-5
Select Database Server



6. Select the type of database used for the installation and click Next. The Database Details screen appears.

Figure 3-6
Database Details



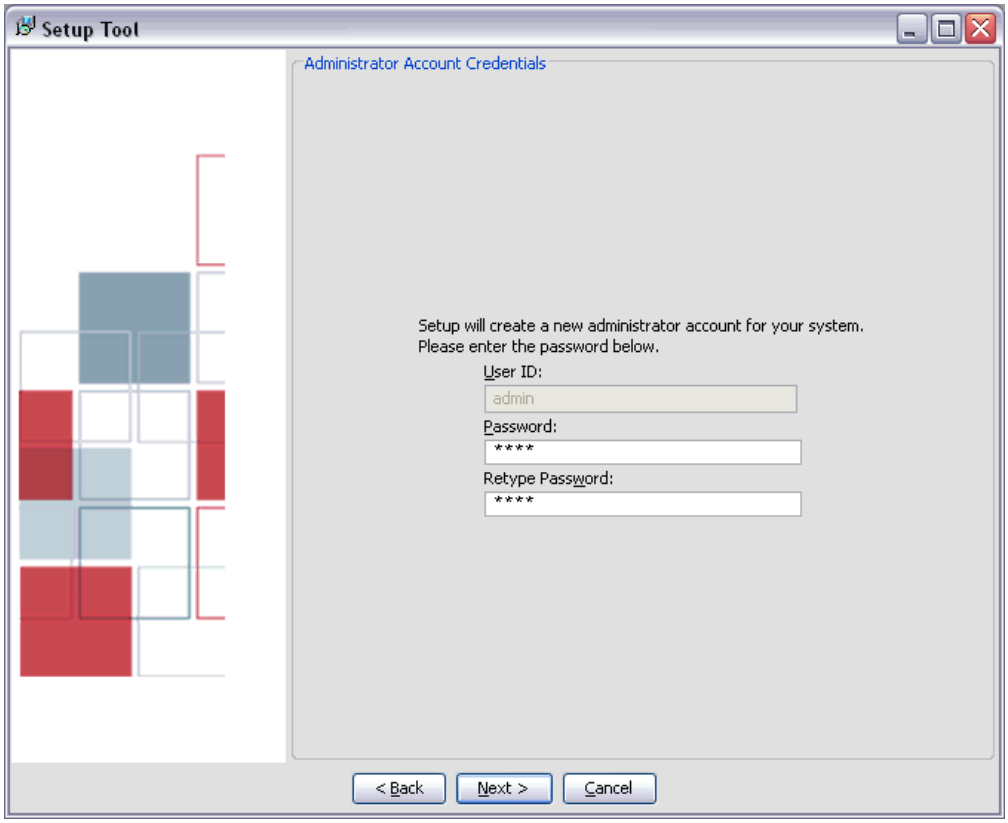
7. Supply the necessary parameters for connecting to the database. The parameters needed depend on the database and include:
   - **Host.** Host name or IP address of the database server.
   - **Port.** Port number on which the database server is running.
   - **Database/SID.** For databases other than DB2 on IBM i, name of an existing database to which to connect.
   - **Username.** Account used to connect to the database. This user must have rights to modify the selected database.
   - **Password.** Password associated with the user name.
   - **Library.** For DB2 on IBM i, the name of the library collection to be used. If the library does not exist, it will be created.

8. Specify whether secure (SSL) database connections must be used.

   Note: To enable SSL connection to the database, the database must be pre-configured for SSL access. Consult vendor documentation for more information. Also, the application server JRE must have the certificates installed. For information on managing certificates, see *http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html*.

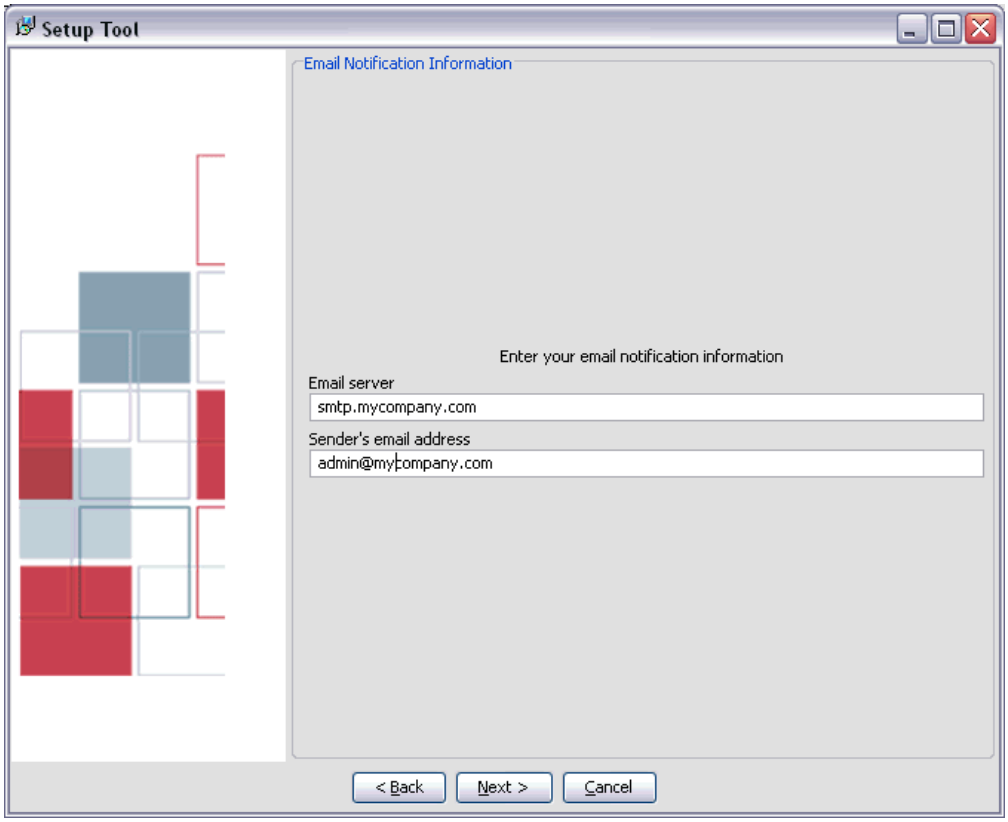
9. Verify the information entered is correct and click Next. The Administrator Account Credentials screen appears.

Figure 3-7
Administrator Account Credentials



10. Type a user ID and password for the administrator account and click Next. The administrator account is used when logging in for the first time; additional users are created after logging in to the system using this account. The E-mail Notification Information screen appears.

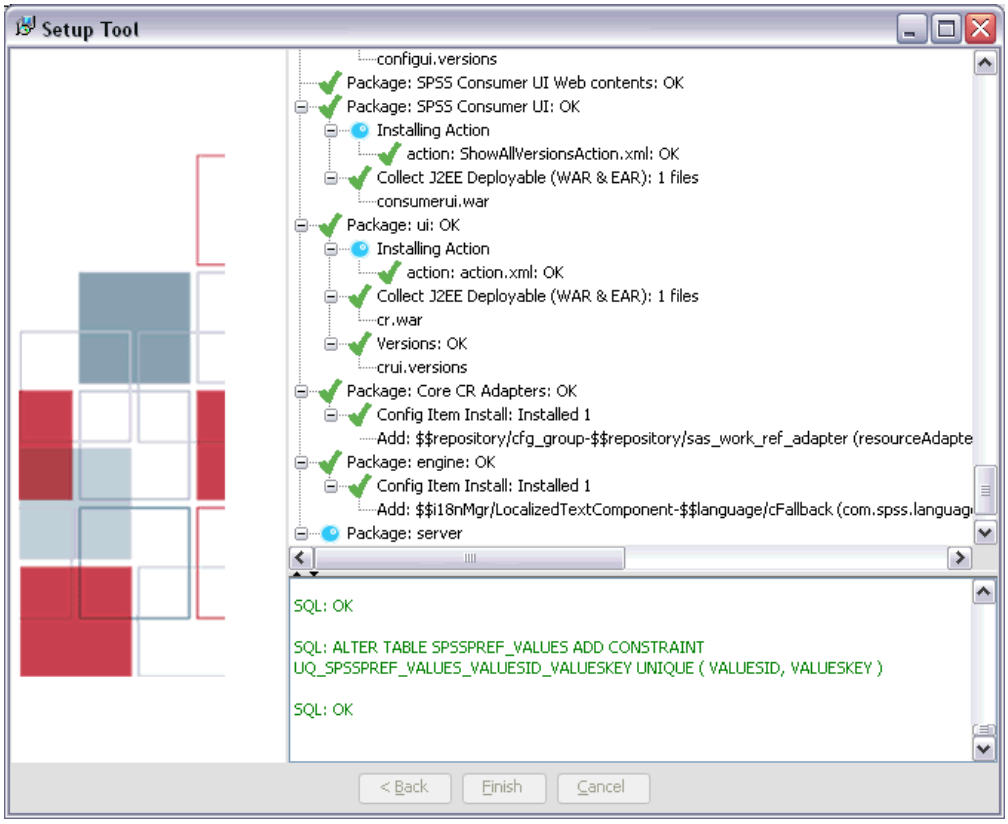
Figure 3-8
E-mail Notification Information



11. Type the name or IP address of the server used for outgoing e-mail and a valid address for the e-mail sender. Click Next. The Deployment screen appears.

12. Click Next to begin deploying the components. When complete, the Deployment Summary screen appears.

Figure 3-9
Deployment Summary



13. Review the deployment log. A green check mark icon indicates a component has been successfully deployed. In the event of a system or deployment error, a red X icon appears, along with a diagnostic report of the error.

14. Click Finish to complete the installation.

Command Line Setup

To launch setup, execute the script in *<PASW Collaboration and Deployment Services installation directory>/setup/*.

clisetup.bat

Command line setup prompts for the same information as the graphical setup wizard (see above). Most fields have default values shown in square brackets. Pressing Enter will accept the default value. Although passwords are echoed on-screen as typed, they are saved in encrypted form. At any time, typing \restart and pressing Enter (or Return) will return to the initial installation screen.

Setup Notes

- The setup progress is recorded in *<Installation directory>/setup/log/setup.log*. If you are deploying the repository into BEA WebLogic application server, for security reasons this file must be deleted after you have verified that the installation completed successfully.
- The parameter values specified during setup are saved in *<PASW Collaboration and Deployment Services Installation Directory>/platform/setupinfo.xml* and will be used if setup is rerun.

Changing Master Database Password

For security reasons, it may be necessary to change the master database password following repository installation. In such cases the password used by the repository for database access must also be changed. PASW Collaboration and Deployment Services provides a utility for changing the database password which can be used in GUI or command line mode.

Note: If WebLogic application server is used with the repository, the password must be changed in PASW Collaboration and Deployment Services before it is changed in the database.

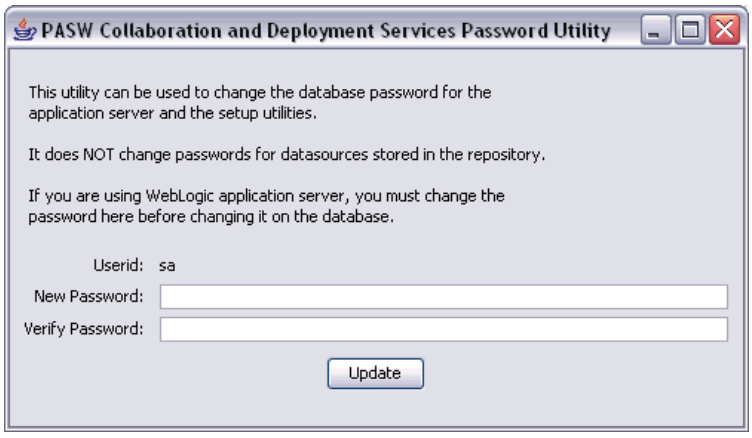
To run the password change utility in GUI mode:

1. Execute

   <PASW Collaboration and Deployment Services Installation Directory>/setup/dbpassword.bat

   Password Utility dialog opens.

Figure 3-10
Password Utility



2. Specify and confirm the new password.
3. Click Update. The password used by the repository for database access is changed.
4. Run PASW Collaboration and Deployment Services setup utility. For more information, see Setup on p. 19

To run the password change utility in command line mode:

1. Execute

   <PASW Collaboration and Deployment Services Installation Directory>/setup/clidbpassword.bat

2. Specify and confirm the new password using the command prompt.
3. Run PASW Collaboration and Deployment Services setup utility.

The password can also be changed by modifying the application server settings. Note that the password is stored in encrypted form, therefore the new password must be converted to an encrypted string by running *encrypt.bat* with the password as command line argument.

Upgrading Repository

Users with an existing version of the repository can conveniently upgrade their environment to take advantage of new features and functions. To upgrade to the current version:

1. Verify that hardware and software requirements are met and determine an installation directory for the application.
2. Reinstall the application server. The old instance of the application server cannot be used with the upgraded repository installation.
3. Install the latest version of the repository. It is recommended to use the already existing installation directory.
4. When prompted, specify the path of the application server.
5. When prompted, preserve the existing data in the existing database.

For detailed information about PASW Collaboration and Deployment Services migration, see Chapter 4

Uninstalling Repository

In the event that an installation becomes corrupt or the repository needs to be reinstalled due to system errors, the current version must be uninstalled.

Note: Back up the database before continuing. Uninstalling removes any tables it has created in the database. You will not be prompted to save this data.

To uninstall the repository:

1. Stop the repository.
2. Back up any data you wish to save in the repository. These tables are removed during the uninstallation process.
3. From the installation path, navigate to the *setup* directory.
4. On Windows, run *uninstall.bat*.

5. When prompted, confirm that the repository should be removed from the system. The uninstall script then undeploys services and deletes tables from the database.

6. When the script is complete, manually delete the root installation directory for the application.

JDBC Drivers

The reporting functionality of PASW Collaboration and Deployment Services is enabled by BIRT (Business Intelligence and Reporting Tools), an open-source package distributed by Eclipse Foundation under the Eclipse Public License. BIRT provides core reporting features, such as report layout, data access, and scripting. For more information about BIRT, see the BIRT project page (*http://www.eclipse.org/birt*). The PASW Collaboration and Deployment Services installation includes the BIRT reporting engine server components, which enable the execution of BIRT report syntax files as part of the PASW Collaboration and Deployment Services reporting job steps. PASW BIRT Report Designer is a standalone application that can be used in conjunction with PASW Collaboration and Deployment Services. It provides a rich user interface with a number of advanced features for creating reports and must be installed separately.

PASW BIRT Report Designer installation contains a set of SPSS Inc. JDBC drivers for all major database systems: Oracle, DB2, and SQL Server. These JDBC drivers are also installed by default with the repository. If a BIRT report uses a JDBC driver other than the ones installed by default, the driver must be installed in the repository. Depending on the application server, the directory location of the JDBC drivers is as follows:

JBoss. *<JBoss Installation Directory>/server/default/lib*

Oracle WebLogic. *<Repository Installation Directory>/SPSSDomain/lib*

WebSphere. *<WebSphere Installation Directory>/lib/ext*

Note that for Netezza, the version 5.0 driver should be used to access both version 4.5 and 5.0 databases.

To access Netezza from PASW Collaboration and Deployment Services running on Windows with JBoss application server, modify *<JBOSS_HOME>\wrapper.wrapper.conf* to include *nzjdbc.jar* in the wrapper classpath, for example:

```
wrapper.java.classpath.4=D:/nzjdbc.jar
```

Chapter
4

Migration

The following migration scenarios are supported for PASW Collaboration and Deployment Services 4.1:

- Repository migration from SPSS Predictive Enterprise Services 3.5 to PASW Collaboration and Deployment Services 4.1.
- Repository migration from PASW Collaboration and Deployment Services 4.0 to PASW Collaboration and Deployment Services 4.1.
- Migration of an existing installation to a different server.

Migration Paths

The following paths can be used for migrating to PASW Collaboration and Deployment Services 4.1 from an earlier version of the system:

- Saving and restoring the repository. In most environments, saving and restoring the repository is recommended.
- "Over-the-top" installation. Installation of PASW Collaboration and Deployment Services 4.1 with an existing repository database is generally more resource-intensive because additional backups of the operational environment may be required.

Important! Regardless of the selected migration path, it is recommended that the latest patches be applied to the existing installation before migration is performed. To obtain the patches, contact SPSS Inc. product support.

Saving and Restoring the Repository

Save and Restore Utility can be used to preserve the configuration and the contents of existing SPSS Predictive Enterprise Services 3.5 and PASW Collaboration and Deployment Services 4 and 4.1 repositories including the following:

- Content repository files and folder structure
- Scheduling and notification components
- Local users
- Locally defined overrides of remote directory user lists and groups
- Role definitions and membership
- User preferences
- Notification templates

- Icons
- Deployed packages

The repository is saved in a compressed archive file which can later be used to restore the content a configuration settings.

Migration process does not automatically add label security actions, such as *Show All Versions* and *Show Latest* to role definitions, and non-administrator users may not be able to see labeled versions and latest versions of objects. PASW Collaboration and Deployment Services administrator must manually assign the actions to non-administrator roles after migration. Also, migration from SPSS Predictive Enterprise Services 3.5 to PASW Collaboration and Deployment Services 4.1 does not preserve configured external security providers, such as Microsoft Active Directory or IBM i. For more information, see the corresponding sections of *PASW Collaboration and Deployment Services 4.1 Administrator's Guide*.

Important! Save and restore utility preserves the package configuration, but PASW Collaboration and Deployment Services may require updated versions of packages. For example, it may be necessary to install newer versions of PASW Modeler adaptors.

The following table presents the use cases for Save and Restore Utility.

Source	**Target**		
	SPSS Predictive Enterprise Services 3.5	**PASW Collaboration and Deployment Services 4.0**	**PASW Collaboration and Deployment Services 4.1**
SPSS Predictive Enterprise Services 3.5	Supported	Supported	Supported
PASW Collaboration and Deployment Services 4.0		Supported	Supported
PASW Collaboration and Deployment Services 4.1			Supported

Important! The save and restore mechanism in PASW Collaboration and Deployment Services is intended primarily for migration purposes and is not a substitute for database backup. A regular backup of the repository database outside of PASW Collaboration and Deployment Services is strongly recommended.

When Save and Restore Utility is used as a migration tool, the following prerequisites must be in place before migration is performed:

- Existing SPSS Predictive Enterprise Repository database must be backed up.
- PASW Collaboration and Deployment Services 4.1 must be installed.

The following steps must be completed for a successful migration:

- Save the existing repository.
- Restore saved data to PASW Collaboration and Deployment Services 4.1 repository.

- Rerun setup tool to update system configuration values.
- Reinstall packages. For more information, refer to product-specific adaptor documentation. For example, for information about reinstalling PASW Modeler adaptor, see PASW Modeler documentation.

Saving the Repository

The SPSS Save Tool can be used as a GUI application or as a command line utility. On systems without a GUI interface, it must be used as a command-line application. It can also be called in batch mode by other applications. The user must be assigned the Administrator role in PASW Collaboration and Deployment Services to perform the save operation. It is strongly recommended to stop PASW Collaboration and Deployment Services before saving.

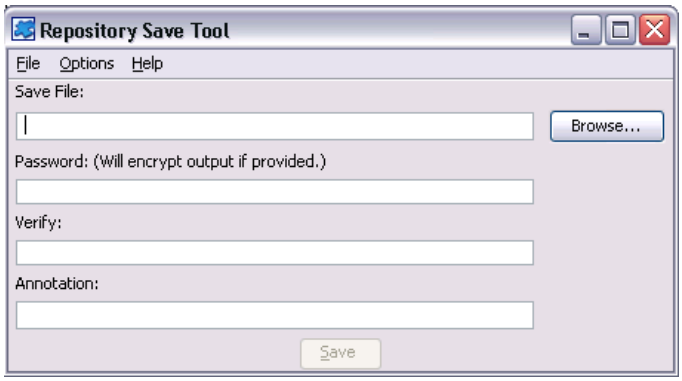
To save a repository using the GUI application:

1. Navigate to *<PASW Collaboration and Deployment Services Installation Directory>/setup/*.
2. Execute *save.bat*.
3. When prompted, enter the username and password.

Figure 4-1
Local Administrator Logon dialog box for the Save Tool

4. Click OK to log in. The Save Tool dialog box opens.

Figure 4-2
Save Tool dialog box



5. Select the save format.

- To save the data as a compressed archive, from the menus choose:

Options
Single .PESsave

- To save the data as a collection of files, from the menus choose:

Options
Directory with Files

6. Enter the file/directory path or click the Browse button to navigate to the location where the data will be saved.

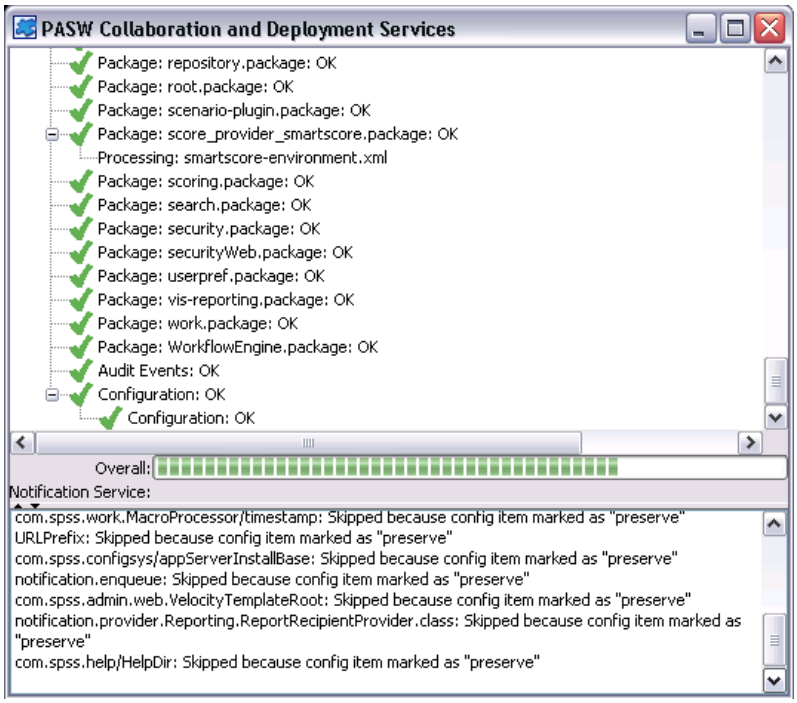
   Note: If the archive file has been selected as the save option, the *.PESsave* extension will be automatically appended to the specified filename. If the directory has been selected as the save option, the target directory cannot already contain PASW Collaboration and Deployment Services save data.

7. To encrypt the data, enter and verify the password. Any alphanumeric string can be used as the password.

8. Add the annotation to the saved data if necessary. An annotation is a descriptive string that will be displayed when the data source (archive file or directory) is selected for system restore.

9. Click Save. The status panel appears.

Figure 4-3
Save operation progress



If errors occur during the save operation, they are displayed in red in the bottom pane. The installation log can be found in *<PASW Collaboration and Deployment Services Installation Directory>/setup/logs/saverestore.log*. At the end of the operation, a message indicating the duration is also displayed.

10. Close the status panel. This will also close the Save Tool.

To save the repository using the command line utility:

1. Navigate to *<PASW Collaboration and Deployment Services Installation Directory>/setup/*.
2. Execute the saverestore -headless command with the following required arguments:
   - -userid <user ID>. The user under whose credential the save operation is being performed.
   - -userpassword <password>. The password of the user.
   - -save <data location path>. The path of the saved data.

   Optional arguments include:
   - -explode. The option to save the data as a directory.
   - -filepassword <file password>. Encryption password.
   - -annotation <annotation>. The annotation string. If the annotation contains spaces, it must be enclosed in quotation marks.
   - -lang <language code>. The language code for localized instances of PASW Collaboration and Deployment Services.

   The following example illustrates saving the contents of the repository in a password-protected file with an annotation.

```
saverestore -headless -userid admin -userpassword pass1234 -save c:/temp/saveFile -filepassword secret
-annotation "Preparing data for migration 1/09/2009"
```

Restoring the Repository

The Restore Tool can be used as a GUI application or as a command line application. On systems without a GUI interface, it must be used as a command-line application. It can also be called in batch mode by other applications. The user must be assigned the Administrator role in PASW Collaboration and Deployment Services to perform the restore operation.

Note: If you experience problems with the Restore Tool graphical user interface in Java 1.5 environment, it may be necessary to upgrade to Java 6. Alternatively, you can run Restore Tool as command line application.

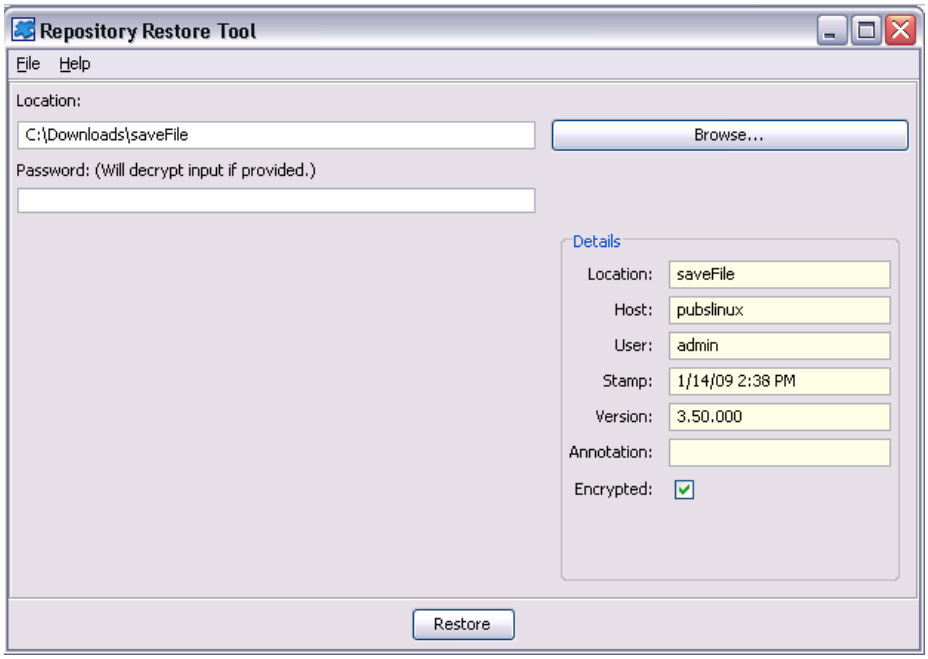
If PASW Collaboration and Deployment Services is being restored over an existing instance, the existing content will be overwritten. In such cases, it is strongly recommended to stop PASW Collaboration and Deployment Services before restoring.

If PASW Collaboration and Deployment Services is being migrated to another server, application components must already be in place. Therefore, the installation must be run prior to restoring. After the repository has been restored, it must be reindexed. For information about reindexing, see administrator documentation.

To restore the repository using the GUI application:

1. Stop the repository server.
2. Navigate to *<PASW Collaboration and Deployment Services Installation Directory>/setup/*.
3. Execute *restore.bat*.
4. When prompted, enter the username and password.
5. Click OK to log in. The Restore Tool dialog box opens.

Figure 4-4
Restore Tool dialog box: Restore tab



6. Enter the file/directory path or click the Browse button to navigate to the location where the data were previously saved. After the data source has been selected, the corresponding information is displayed in the Details group box.

   Note: If the restore utility is run, the specified data source path is retained and will be displayed by default the next time the restore utility is opened.

7. If the data have been encrypted, enter the password. The field is unavailable for unencrypted files.
8. Click Restore. The status panel appears. If errors occur during the restore operation, they are displayed in red in the bottom pane. The installation log can be found in *<PASW Collaboration and Deployment Services Installation Directory>/Enterprise Repository/setup/logs/saverestore.log*. At the end of the operation, a message specifying duration is also displayed.
9. Close the status panel. This will also close the Restore Tool.

To restore the repository using the command line utility:

1. Stop the repository server.
2. Navigate to *<PASW Collaboration and Deployment Services Installation Directory>/setup/*.
1. Execute the saverestore -headless command with the following required arguments:
   - -userid <user ID>. The user under whose credential the restore operation is being performed.
   - -userpassword <password>. The password of the user.
   - -restore <data location path>. The path of the restored data.

   Optional arguments include:
   - -filepassword <file password>. For encrypted files, the password.
   - -setupdir <path>. Option to indicate that the setup directory is different from the current directory.

   The following example illustrates restoring the contents of the repository from password-protected file.

   ```
   saverestore -headless -userid admin -userpassword pass1234 -restore c:/paswuser/saveFile -filepassword secret
   ```

Rerunning Setup

If a repository from an earlier version of the system has been migrated to PASW Collaboration and Deployment Services 4.1 or if an existing installation has been migrated to a different server environment, system configuration values in the repository must be reset by rerunning the setup utility. The utility is initially run as part of PASW Collaboration and Deployment Services 4.1 installation.

To rerun setup:

1. Start the setup utility.

   ```
   <PASW CADS 4 installation directory>/setup/setup.bat
   ```

2. Specify setup parameters as prompted by the wizard or command line. The parameters include keystore location, application server, database, administrator password, and email settings for notifications. For more information about the setup utility, see Setup on p. 19

Overwriting an Existing PASW Collaboration and Deployment Services Installation

You can also upgrade to PASW Collaboration and Deployment Services 4.1 by installing the system over an older version. In that case, during the setup you must point to the existing repository database.

Important! Full database backup is strongly recommended prior to over-the-top installation because it is impossible to revert to the old version of the repository once PASW Collaboration and Deployment Services 4.1 have been installed.

Adaptor packages, such as PASW Modeler adaptor, must be reinstalled. For more information, refer to product-specific adaptor documentation. For example, for information about reinstalling PASW Modeler adaptor, see PASW Modeler documentation. The repository must also be reindexed. For information about reindexing, see administrator documentation.

Note: If Java encryption used while installing PASW Collaboration and Deployment Services over an existing database is different from the encryption used by the original instance (for example, IBM Java encryption versus Sun Java encryption), credentials passwords will not be migrated and setup will report failure. However, PASW Collaboration and Deployment Services 4.1 can be still started, and you can use Deployment Manager to manually change credentials passwords. For information about PASW Collaboration and Deployment Services credentials, see Deployment Manager documentation.

Chapter
5

Optional Components

This chapter provides the information about the installation and configuration of the following optional components of PASW Collaboration and Deployment Services:

- Web installation modules for PASW BIRT Report Designer and Enterprise View Driver
- Remote Process Server
- Python Scripting

For information about installing Enterprise View Driver, see *Enterprise View Driver 4.1 Guide*.

Web Installations from the Repository

In order to enable Web installations of PASW BIRT Report Designer and Enterprise View Driver, the following optional packages must be deployed into the repository:

- PASW BIRT Report Designer—*birtdesignerinstall.package*
- Enterprise View Driver—*pevdriverinstall.package*

The packages can be found in the */PASW/Web/* directory of PASW Collaboration and Deployment Services distribution Disk 1. The packages are deployed using the Package Manager utility. For more information, see the topic Updating the Repository in Chapter 11 on p. 104.

Note: If you are enabling Web installations in a repository running on WebSphere 6.1 application server on Windows 2003, it may be necessary to increase the value of com.ibm.SOAP.requestTimeout attribute in the *IBM\WebSphere\AppServer\profiles\<profile>\properties\soap.client.props* configuration file.

Remote Process Server

In order to enable remote process execution in PASW Collaboration and Deployment Services, the remote process component must be deployed on the machine that is to be configured as a remote server. The installation involves:

1. Copying the necessary files from the distribution media to the target computer.
2. Configuring the remote process server.
3. Starting the remote process server.

   This can be accomplished by using either the graphical installation wizard or the command line equivalent. Environments without a graphical interface must use the command line approach. When executing the Windows batch file or executable shell scripts provided on the installation

media, the user installing the application must have permissions to install software under the operating system.

Installation Notes

- After the component has been copied, the repository database connection information must be provided. Select the database type and then specify database host, database name, user name, and password.
- For remote process server configuration, server name, access port, and whether a secure connection is to be used must be specified.
- Clustering can be enabled for a remote process server. If clustering is enabled for a specific instance of a repository, it will be possible to include the remote server in a cluster defined in that repository. If you choose not to enable clustering, the installation will proceed to completion. Otherwise, you must specify the host, port, and login credentials of the repository for which clustering is to be enabled.

Graphical Installation Wizard

1. When the disk menu opens, click Install Remote Process Server, or execute the program to start the installation wizard in the */RPS/Disk1/InstData/<OS Name>/NoVM/* directory of Disk 2. For Windows, this is *install.exe*. For Unix-based systems, the setup file is named *install.bin*.
2. After the installation wizard is launched, follow the instructions on the screen.

Command Line Installation

Command line installation must be used on systems without a graphical interface. After verifying that a database server exists for the repository to connect to, execute the program in the */RPS/Disk1/InstData/<OS Name>/NoVM/* directory of Disk 2 with the console command line switch.

- On Windows:

```
install.exe -console
```

- On UNIX:

```
./install.bin -console
```

- On IBM i, in QShell environment copy *setupi5.sh* script and the installation JAR files to a temporary directory and then run setup using commands similar to the following:

```
cp /qopt//OPT_DVD/RPS/setupi5.sh /temp
cp /qopt//OPT_CD/RPS/*.jar /temp
cp /qopt//OPT_CD/RPS/SETUP.JAR /temp
/temp/setupi5.sh
```

Note: Remote process server installation on IBM i requires classic JVM 1.5 to be enabled.

After the installation wizard is launched, follow the instructions on the screen. Many items have default values, which are always shown in square brackets. Pressing Enter will accept the default value. Although passwords are echoed on-screen as typed, they are saved in encrypted form.

Starting and Stopping Remote Process Server

After the remote process server has been installed on the target host system, it must be started.

- To start the server, execute the following command:

  (Windows)
  <Remote Process Server Installation directory>/startserver

  (UNIX and IBM i)
  <Remote Process Server Installation directory>/startserver.sh

- To enable remote process server over a secure connection additional parameters must be specified:

  (Windows)
  <Remote Process Server Installation directory>/startserver "-Djavax.net.ssl.keyStore=./keystore" "-Djavax.net.ssl.keyStorePassword=remote"

  (UNIX and IBM i)
  <Remote Process Server Installation directory>/startserver.sh "-Djavax.net.ssl.keyStore=./keystore" "-Djavax.net.ssl.keyStorePassword=remote"

- To stop remote process server, execute the following command:

  (Windows)
  <Remote Process Server Installation directory>/shutdown

  (UNIX and IBM i)
  <Remote Process Server Installation directory>/shutdown.sh

PASW Collaboration and Deployment Services Scripting

PASW Collaboration and Deployment Services provides a scripting framework with a set of Content Repository and Process Management APIs that advanced users and administrators can use to write independent routines or batch jobs that combine a set of routines. This can greatly simplify bulk tasks such as changing security permissions for a large group of users, labeling or removing a label from a large number of folders/files, or uploading/downloading a large number of folders/files. The framework includes the ability to perform tasks from the command line, as well as a rich API for interacting with PASW Collaboration and Deployment Services within your own Python code.

For general information about Python, a dynamic object-oriented programming language, see the Python site (*http://www.python.org*).

Installing Scripting on Windows

1. If Python is already installed on your system, uninstall it.
2. Insert the installation media.

3. Open the *PYTHON\Disk1\InstData\NoVM* directory of Disk 2 and double-click *install.exe*. Follow the screen instructions to complete the installation. Install to the default location. This installs the required Python, ZSI, and PyXML technologies.
4. Open the *PYTHON* directory on the installation media and extract the contents of *cads-scripting-1.0.zip* to a temporary directory.
5. Add the Python scripting location to your PC's **Path** system environment variable.
6. At a command prompt, change the current directory to the folder where you extracted *cads-scripting-1.0.zip*. Type the following command and press Enter.

```
python setup.py install
```

Installing Scripting on UNIX

1. If Python 2.4.3, ZSI 2.0 rc3, and PyXML 0.8.4 are not already installed on your system, install after downloading from their respective web sites before proceeding to step 2.
   - Python 2.4.3: *http://www.python.org/download/releases/2.4.3/*
   - ZSI 2.0 rc3: *http://sourceforge.net/projects/pywebsvcs*
   - PyXML 0.8.4: *http://sourceforge.net/project/showfiles.php?group_id=6473*
2. Insert Disk 2.
3. Open the *PYTHON* directory and extract the contents of *cads-scripting-1.0.tar.gz* to a temporary directory.
4. In the temporary directory, edit *setup.cfg*. Replace `<PythonInstallDir>` with PASW Collaboration and Deployment Services scripting installation path. If no value is specified, the path will default to Python library, for example */usr/lib/python2.4*.

```
[install]
install-base = <PythonInstallDir>
install-data = <PythonInstallDir>
install-purelib = <PythonInstallDir>
install-scripts = <PythonInstallDir>
install_headers = <PythonInstallDir>
```

5. At a command prompt, change the current directory to the folder where you extracted *cads-scripting-1.0.tar.gz*. Execute the following command:

```
python setup.py install
```

Installing Scripting on IBM i

1. Log into your IBM i system using a Telnet terminal.
2. Insert Disk 1.
3. Start QShell with the following command

```
QSH
```

4. Change the directory to */qopt/PASW/IBMi/Python*.
5. Copy the content of the directory to a temporary location.

6. Run the installation script by executing the following command:

   ```
   ./PyInst.scr
   ```

   Python is installed as */QOpenSys/usr/local/bin/python2.4* and PASW Collaboration and Deployment Services Scripting is installed in */QOpenSys/usr/local/lib/python2.4/site-packages*.

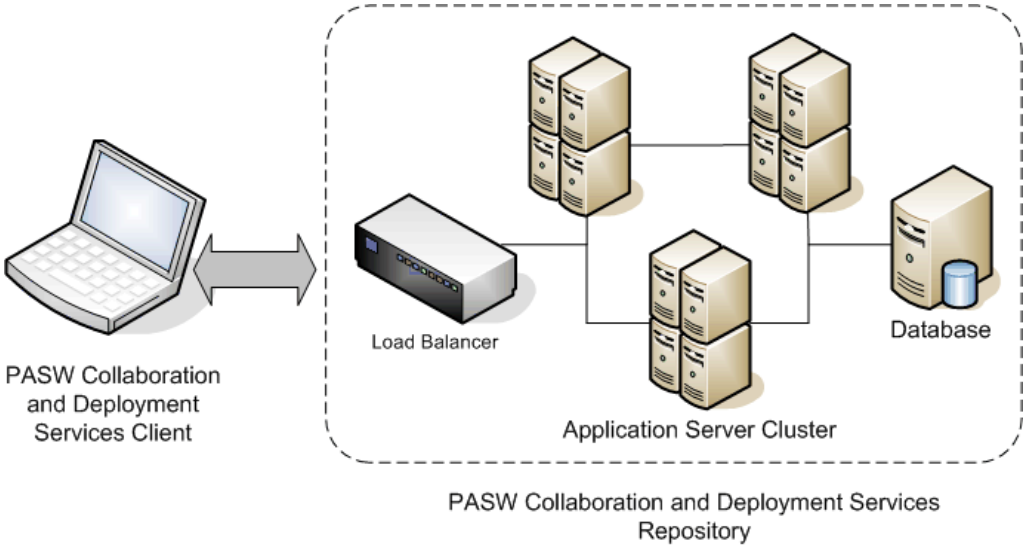
   For information about using scripting, see the customization documentation.

Chapter 6

Clustering

The repository can be deployed into an environment of clustered J2EE application servers. Each application server in the cluster should have the identical configuration for the hosted application components and the repository is accessed through a hardware- or software-based load balancer. This architecture allow processing to be distributed among multiple applications servers and it also provides redundancy in case of a single server failure.

Figure 6-1
PASW Collaboration and Deployment Services clustered deployment architecture



PASW Collaboration and Deployment Services currently supports clustering WebSphere and WebLogic application servers.

Installation

The process of installing the repository into the cluster includes the following steps:

- Initial installation and configuration of application components on an arbitrarily selected node in the cluster, which is performed by the repository installation wizard.
- Subsequent deployment of the application components into all of the nodes of the cluster performed through Jython-based script utilities or manually.

Initial installation of the repository components must follow these guidelines:

- The repository should be installed on a single node in the cluster.

- The cluster install location should be a shared directory available to all nodes in the cluster as a sheared directory or a mounted drive.
- In the setup wizard, clustered install option must be selected.
- Regardless of the application server type, the following application server information must be provided:

Property	Description
Cluster Install Location	Location of the files to be deployed into the cluster. This directory will contain the repository applications and configuration files and a set of scripts that can assist in cluster configuration.
Cluster Name	The name of the WebSphere/WebLogic cluster that you will deploy into. If a cluster has been preconfigured, the cluster name must be provided. Otherwise, you must remember the name you specify because it will be required to create the cluster at a later time.
Load Balancer URL	The URL the clients will use to connect to the cluster. Typically this will be the URL of the load balancer.
Secure Communication	The option specifies whether secure communication will be used for HTTP/SOAP messages within the cluster. If it is selected, SSL must be configured in the cluster.

- The setup must be completed. For more information about the wizard, see Installing the Repository on p. 18

After initial installation and configuration has been completed, the following directory structure is created in cluster install directory:

Subdirectory	Description
bin	OS-specific scripts for automating cluster deployment and configuration.
doc	Text files containing application server-specific instructions for deploying the repository into a cluster.
lib	Global libraries required for running the repository.
logging	Logging configuration files.
scripts	Jython scripts for automating cluster deployment and configuration.
toDeploy	Deployable application files.

Follow the application server-specific instructions in the following topics to complete a script-assisted or manual deployment into all nodes.

WebSphere

There are two methods for to deploying PASW Collaboration and Deployment Services into a WebSphere cluster:

Scripted Deployment is intended for less advanced users who want assistance with deploying the platform into an application server. Several scripts are provided for automating the deployment process.For more information, see the topic Scripted Deployment on p. 48.

Manual Deployment is intended for advanced users who want precise control over their application server environment. All deployment can be done through the administration console.For more information, see the topic Manual Deployment on p. 51.

Notes

- Deploying PASW Collaboration and Deployment Services into a WebSphere cluster requires the use of a WebSphere shared library scoped to the WebSphere cluster. In order to use this functionality, WebSphere must have be at Fix Pack 13 (6.1.0.11) or later. If you are using the deployment scripts (*wsadmin*), you must also recreate your WebSphere Deployment Manager profile after the fix pack is applied.
- If you use a Windows share as the shared file system for installing PASW Collaboration and Deployment Services, you must configure the Node Agent Windows service to run as a Windows user that has access to the share. You must also use the UNC path as opposed to a mapped drive when configuring the installation because mapped drives are not available to Windows services.
- Make sure you use the IBM JVM/JRE when running setup. This is required to ensure that the same JCE (encryption) provider is used at setup time as will be used at run time. Because IBM WebSphere uses its own JVM, you must run setup with that JVM, so the same JCE Provider is used to create the keystore as will be used to read the keystore when the server is started. For more information, see the topic Setup in Chapter 3 on p. 19.

Scripted Deployment

Scripted Deployment Files

There are several scripts that can be used to automate the cluster setup and the cluster deployment into WebSphere. These scripts are located in the *<cluster install location>/scripts* directory.

config.ini

The file contains the parameters used by the Jython scripts (described below) to automatically create a cluster. The file contains the following sections and properties:

cluster

- **name** name of the cluster (for example, *websphere_cluster*).
- **cell** name of the WebSphere cell for the cluster (for example, *WSC1Cell01*).
- **singletonServer** name of the server to deploy applications that can only run on a single server.
- **singletonNode** name of the node for the singleton server.

servers

- **name** name of the server (for example, *platServer1*).
- **node** name of the node for the server (for example, *WSC1Node01*).
- **javaInitHeapSize** initial Java heap size (for example, 256).
- **javaMaxHeapSize** maximum Java heap size (for example, 1024).

- **platformOS** operating system, valid values include aix, aix64, hpux64, linux, linux64, solaris64, windows, windows64.
- **platformSharedDir** shared platform installation directory (for example, *\\machine\shared\platform_install*).
- **platformLocalDir** directory of the local platform installation (for example, *C:\platformLocal*).
- **platformKeystoreLocation** location of the platform keystore created at install time.
- **platformKeystorePassword** password of the platform keystore created at install time.

jms

- **dataStoreSchema** name of the schema to use for the JMS datastore.

platform (prepopulated by setup)

- **database.name** name of database select in setup.
- **database.driver** platform database driver class name.
- **database.host** host for the database server.
- **database.library** database library (iSeries only).
- **database.user** platform database user.
- **database.password** platform database password (may be encrypted).
- **database.url** platform database URL.
- **deploy.directory** platform *toDeploy* directory.

These configuration parameters are used as follows:

- **platformDeploy.py** creates a new cluster containing all the servers defined in the servers section. The configuration in the platform section will be used to create the platform datasource and deploy the platform applications to the cluster.
- **platformClean.py** is used to remove the components installed by the *platformDeploy.py* script.

platformDeploy.py

The script is used to deploy platform components to a clustered WebSphere domain as configured in *config.ini*. Arguments include:

- **all** Deploy everything (default).
- **sharedLibrary** Deploy shared library only.
- **cluster** Deploy cluster, servers, and shared library.
- **servers** Deploy servers only.
- **virtualHosts** Update the virtual host aliases for the cluster.
- **components** Deploy datasource and JMS components.
- **datasource** Deploy datasource components only.
- **jms** Deploy JMS components only.
- **applications** Deploy platform applications.
- **patch** Deploy updated platform applications.

platformClean.py

The script is used to undeploy platform components from a clustered WebSphere domain as configured in *config.ini*.

- **all** Undeploy everything (default).
- **sharedLibrary** Undeploy shared library only.
- **cluster** Undeploy cluster, servers, and shared library.
- **servers** Undeploy servers only.
- **components** Undeploy datasource and JMS components.
- **datasource** Undeploy datasource components only.
- **jms** Undeploy JMS components only.
- **applications** Undeploy platform applications.
- **patch** Undeploy updated platform applications.

Note: If you clean your JMS components you will also need to remove the database tables before you can recreate them. Delete the tables that start with SIB in your platform database. They will be recreated on server startup once you recreate your JMS components.

The executable scripts for running Jython scripts are located in the *<cluster install location>/scripts* directory. They include:

- **setEnv** Sets up the environment.
- **wsadmin** Executes a specified Jython script.
- **installNode** Installs the necessary platform components on the local file system.

Scripted Deployment

1. Install the same version of WebSphere Network Deployment on each node in the cluster.
   - Setup a single WebSphere Deployment Manager. If you have patched WebSphere after you created the Deployment Manager profile, you may need to recreate the Deployment Manager profile in order for the *wsadmin* scripts to run correctly.
   - Federate all nodes in the cluster using the Deployment Manager.
2. Setup the platform install directory as a shared directory for each node in the cluster.
4. Update */bin/setEnv* to set the values for the following environment variables
   - **DM_PROFILE_HOME** Location of your WebSphere Deployment Manager profile
   - **WSADMIN_LANG** Language of your scripts (leave the default of Jython)
   - **WSADMIN_SECURITY** User name and password if administrative security is enabled
5. Update *config.ini* to set up your cluster configuration.
6. Run the script to deploy the platform components into a WebSphere cluster. The configuration is read from *config.ini*.
   - Open a command prompt to the *<cluster install location>/bin* directory.

- Execute wsadmin -f ../scripts/platformDeploy.py.
- The following components are deployed: Shared library targeted to the cluster; JDBC datasource targeted to the cluster; JDBC persistent stores targeted to a single server; JMS server targeted to a single server; a JMS connection factory targeted to the cluster; several JMS queues; all platform applications (EARs, WARs, and RARs).

7. Start all nodes in the cluster using the administration console.

Notes:

- If the ports are manually changed for any server in the cluster, the corresponding changes must be made to the default_host virtual host aliases in order to ensure cluster communication functions correctly.
- If *platformDeploy.py* is unable to create shared library objects for the cluster (error code WASX7129E), use instructions in "Shared Library" section of Manual Deployment to create a shared library manually and then rerun the script.

Installing New Packages and Patches

Any updates to the platform server will be available in the updates directory. Updates include patches and new packages installed via package manager. Each update will create a new timestamp directory. It will contain a *toDeploy* directory. To deploy the updates:

1. Modify theupdate.deploy.directory property in config.ini to point to the *toDeploy* directory inside the newly created timestamp directory.
2. Run the script to deploy the platform components to a WebSphere Cluster.
   - Open a command prompt to the *bin* directory.
   - Execute wsadmin -f ../scripts/platformDeploy.py patch.
   - All platform applications (EARs, WARs, and RARs) in the *updates* directory will be updated.

Manual Deployment

These instructions provide advanced J2EE users with the information necessary to deploy PASW Collaboration and Deployment Services into a WebSphere application server cluster. It is expected that the cluster has already been configured and is ready for deployment.

The instructions use the following path placeholders:

<platform_install_directory> The root of the shared installation directory for PASW Collaboration and Deployment Services on a single dedicated node. This is the directory containing folders such as setup, platform, and components.

<path_to_keystore_directory> The directory specified during install where the keystore was created.

<cluster_deploy_directory> The directory specified during install where the cluster deploy files were placed. The default location for this directory is *<platform_install_directory>/cluster_deploy*.

<node_local_directory> The root of the local PASW Collaboration and Deployment Services directory on the server nodes in the cluster. This can be any directory but it is suggested that the paths are the same on all servers.

<ws_cell> WebSphere server cell name.

<new_ear_name> The name of the EAR file that you will create by following these instructions.

Shared File System

The root of the *<platform_install_directory>* must be shared across all nodes in the cluster. It is necessary for each node to have read access to this directory and its entire contents. If *<path_to_keystore_directory>* is not *<platform_install_directory>* or one of its subfolders, *<path_to_keystore_directory>* must also be shared across all nodes in the cluster. On Windows, when referencing these directories from remote nodes, it is recommended that UNC paths be used rather than mapped drives.

Shared Library

A shared library must be configured and scoped to the PASW Collaboration and Deployment Services cluster. The classpath must contain the following entries:

- ${SPSSPLATFORM_DIR}/setup/resources/websphere
- ${SPSSPLATFORM_DIR}/platform/globalLibraries
- ${SPSSPLATFORM_DIR}/setup/lib/DataDirectAdapter.jar
- ${SPSSPLATFORM_DIR}/setup/lib/MFbase.jar
- ${SPSSPLATFORM_DIR}/setup/lib/MFsqlserver.jar
- ${SPSSPLATFORM_DIR}/setup/lib/MFdb2.jar
- ${SPSSPLATFORM_DIR}/setup/lib/MForacle.jar
- ${SPSSPLATFORM_DIR}/setup/lib/MFinformix.jar
- ${SPSSPLATFORM_DIR}/setup/lib/MFsybase.jar
- ${SPSSPLATFORM_DIR}/setup/lib/MFutil.jar
- ${SPSSPLATFORM_DIR}/setup/lib/jt400.jar
- ${SPSSPLATFORM_DIR}/setup/lib/log4j.jar
- ${SPSSPLATFORM_DIR}/setup/lib/commons-logging.jar
- ${SPSSPLATFORM_DIR}/setup/lib/icu4j.jar
- ${SPSSPLATFORM_DIR}/setup/lib/security-global.jar
- ${SPSSPLATFORM_DIR}/setup/lib/search-global.jar
- ${SPSSPLATFORM_DIR}/setup/lib/spsslic.jar
- ${SPSSPLATFORM_DIR}/setup/lib/spsslic7-global.jar
- ${SPSSPLATFORM_DIR}/setup/lib/userpref-global.jar
- ${SPSSPLATFORM_DIR}/components/process/workunit/process-native.jar

- ${SPSSPLATFORM_DIR}/components/process/workunit/JimiProClasses.jar
- ${SPSSPLATFORM_DIR}/components/process/workunit/nvizn.jar
- ${SPSSPLATFORM_DIR}/components/process/workunit/visual_parse.jar
- ${SPSSPLATFORM_DIR}/setup/lib/spsswebsphere.jar

WebSphere variable SPSSPLATFORM_DIR must be set up for each node in the cluster and point to *<platform_install_directory>*. For each server in the cluster, a classloader needs to exist with the following settings:

- Classes loaded with parent class loader first.
- Contains a shared library reference to the library defined above.

Datasource

A JAAS Authentication Data entry with an alias of PlatformAuth must be created with the appropriate database username and password. The JDBC datasource with the following parameters must be configured and targeted to the server cluster:

1. JNDI name for the datasource must be set to jdbc/spss/PlatformDS.
2. Authentication Data Alias must be set to PlatformAuth.
3. Minimum connections 20, maximum connections 100.
4. The Datasource Helper Classname must be set to com.spss.setup.websphere.SPSSDataStoreHelper.
5. The ClassPath must be set as follows:
   - ${SPSSPLATFORM_DIR}/setup/lib/DataDirectAdapter.jar
   - ${SPSSPLATFORM_DIR}/setup/lib/MFbase.jar
   - ${SPSSPLATFORM_DIR}/setup/lib/MFsqlserver.jar
   - ${SPSSPLATFORM_DIR}/setup/lib/MFdb2.jar
   - ${SPSPLSATFORM_DIR}/setup/lib/MForacle.jar
   - ${SPSSPLATFORM_DIR}/setup/lib/MFinformix.jar
   - ${SPSSPLATFORM_DIR}/setup/lib/MFsybase.jar
   - ${SPSSPLATFORM_DIR}/setup/lib/MFutil.jar
   - ${SPSSPLATFORM_DIR}/setup/lib/spsswebsphere.jar
6. Implementation Class Name must be set as *com.spss.datadirect.jdbc.SPSSDataSource*.
7. Test Connection must be set to true.
8. Test Connection Interval must be set to 10.
9. The following properties must be added to the datasource:
   - URL must be set to the Database URL.

- *enable2phase* must be set to false.
- *preTestSQLString* must be set to SELECT COUNT(*) FROM SPSSSETUP_PLUGINS.

JMS

The following JMS components must be configured.

1. Non-secured System Integration Bus.
   - It must be created as a *DataStore* with a data source reference of *jdbc/spss/PlatformDS*.
   - The Authentication Alias must be set to *PlatformAuth*.
   - The JMS Data Score Schema will also need to be set.
2. JMS connection factory with the JNDI name *ConnectionFactory*.
3. JMS topic connection factory with the JNDI name *TopicConnectionFactory*.
4. JMS topic *PASWMessageBusTopic* with the JNDI name *topic/PASWMessageBus* for use by the PASW Collaboration and Deployment Services components.
5. JMS queue *PASWScoringQueue* with the JNDI name *queue/PASWScoring* for use by the scoring message driven bean *ScoringMDB*.
6. JMS queue *PASWLogQueue* with the JNDI name *queue/PASWLog* for use by the scoring component.
7. JMS queue *SPSSAuditQueue* with the JNDI name *queue/SPSSAudit* for use by the auditing message driven bean *AuditMDB*.
8. JMS queue *SPSSNotificationQueue* with the JNDI name *queue/SPSSNotification* for use by the notification component.
9. JMS queue *SPSSProcessQueue* with the JNDI name *queue/SPSSProcess* for use by the process component.
10. JMS activation specification *SPSSAuditActivationSpec*.
    - JNDI name of *spss/AuditMDBAS*.
    - Destination JNDI name of *queue/SPSSAudit*.
    - Destination type of Queue
11. JMS Activation Specification *SPSSProcessEventActivationSpec*.
    - JNDI name of *spss/ProcessEventMDBAS*.
    - Destination JNDI name of *queue/SPSSProcess*.
    - Destination type of Queue.
12. JMS Activation Specification *PASWScoringActivationSpec*.
    - JNDI name of *pasw/ScoringMDBAS*.
    - Destination JNDI name of *queue/PASWScoring*.
    - Destination type of Queue.

13. JMS Activation Specification *PASWScoringNotificationsSpec*.
    - JNDI name of *pasw/ScoringNotificationsMDBAS*.
    - Destination JNDI name of *topic/PASWMessageBus*.
    - Destination type of *TopicSpace*.
14. JMS Activation Specification *PASWScoreLogSpec*.
    - JNDI name of *pasw/ScoreLogMDBAS*.
    - Destination JNDI name of *queue/PASWLog*.
    - Destination type of Queue.
15. JMS Activation Specification *PASWDMSResponseLogSpec*.
    - JNDI name of *pasw/DMSResponseLogMDBAS*.
    - Destination JNDI name of *queue/PASWLog*.
    - Destination type of Queue.
16. JMS Activation Specification *PASWDMSSimulationLogSpec*.
    - JNDI name of *pasw/DMSSimulationLogMDBAS*.
    - Destination JNDI name of *queue/PASWLog*.
    - Destination type of Queue.

    Note: All activation specifications must have a maximum concurrency of 5.

JCA Resource Adaptors

Each resource adapter must be deployed to the PASW Collaboration and Deployment Services cluster. In order to do this, they must first be deployed to a single node and then copied to the cluster scope using the *wsadmin* command *AdminTask.copyResourceAdapter*. A deep copy must be performed. The following settings must be used for the resource adapter:

1. Archive path *<platform_install_directory>/platform/resourceAdapters*.
2. Classpath:

   ${SPSSPLATFORM_DIR}/platform/resourceAdapters/<name>.rar

   ${SPSSPLATFORM_DIR}/platform/globalLibraries/<global_dependency>.jar (if applicable)
3. Native path ${SPSSPLATFORM_DIR}/platform/resourceAdapters/<name>.rar.

A *J2CConnectionFactory* must be created for use by the resource adapter. The JNDI name indicated by the resource adapter must be used for the connection factory. The specifics for configuring the various resource adapters can be found in the [RAR_CONNECTION_FACTORIES] section of the *<cluster_deploy_directory>/doc/environment_<timestamp>.properties* files. It may also be necessary to add classpath and/or native path entries if indicated in the *<cluster_deploy_directory>/doc/environment_<timestamp>.properties* files.

Application Deployment

All J2EE applications in the *<cluster_deploy_directory>/toDeploy* directory must be deployed to the PASW Collaboration and Deployment Services cluster.

The workflow application will not be initialized by default in a cluster environment. In order to use the workflow component, the following system property must be set on a single server in the cluster: -Dcom.spss.workflow.active.override=true.

The workflow component must not be initialized on multiple servers in the cluster. All PASW Collaboration and Deployment Services J2EE applications must be deployed using *PARENT LAST* classloading (not the default). *PARENT LAST* classloading must be used for both the deployment classloader as well as the web module class loader. If the applications are not deployed in this manner they will not run correctly.

MDB Deployment

When deploying EAR files, several EJB JNDI Bindings must be defined. The following are the bindings for the various ear files:

auditmdb.ear

- Binding name *SPSSauditMDB*.
  - Module name of *auditmdb*.
  - URI of *auditMDB.jar*, *META-INF/ejb-jar.xml*.
  - Activation specification of *spss/AuditMDBAS*.

process-ejb.ear

- Binding name *ProcessEventMDB*.
  - Module name of *process-ejb*.
  - URI of *process-ejb.jar*, *META-INF/ejb-jar.xml*.
  - Activation specification of *spss/ProcessEventMDBAS*.

scoring-ejb.ear

1. Binding name *ScoringMDB*.
   - Module name of *scoring-ejb*.
   - URI of *scoring-ejb.jar*, *META-INF/ejb-jar.xml*.
   - Activation specification of *pasw/ScoringMDBAS*.
2. Binding name *ScoringNotificationsMDB*.
   - Module name of *scoring-ejb*.
   - URI of *scoring-ejb.jar*, *META-INF/ejb-jar.xml*.
   - Activation specification of *pasw/ScoringNotificationsMDBAS*.
3. Binding name *ScoreLogMDB*.

- Module name of *scoring-ejb*.
- URI of *scoring-ejb.jar,META-INF/ejb-jar.xml*.
- Activation specification of *pasw/ScoreLogMDBAS*.

pasw_dms.ear

1. Binding name *DMSResponseLogMDB*.
   - Module name of *pasw_dms*.
   - URI of *PASWLoggingMdb.jar*, *META-INF/ejb-jar.xml*.
   - Activation specification of *pasw/DMSResponseLogMDBAS*.
2. Binding name *DMSSimulationLogMDB*.
   - Module name of *pasw_dms*.
   - URI of *PASWLoggingMdb.jar,META-INF/ejb-jar.xml*.
   - Activation specification of *pasw/DMSSimulationLogMDBAS*.

Java System Properties

The following Java system properties must be set appropriately for each server in the cluster:

1. JVM memory arguments must be set appropriately (recommended 1024m minimum maximum heap size).
2. -Dcom.spss.configsys.installBase.override=<platform_install_directory>.
4. -Dlog4j.configuration=<node_local_directory>/logging/log4j.xml.
5. -Dplatform.keystore.file=<path_to_keystore_directory>.
6. -Dplatform.keystore.password=<keystore_password>.
   - A non-cleartext version of the keystore password can be found in the *<platform_install_directory>/platform/setupinfo.xml* file.
7. The *Java.library.path* must contain the following:
   - *<platform_install_directory>/components/setup/jni/<OS>*
   - For each resource adapter deployed: *<path_to_domain>/<rar_name>/bin*.
   - Additional native libraries may be added to the *Java.library.path* if required.
8. Check [JAVA_PROPERTIES] section in all *<cluster_deploy_directory>/doc/environment_<timestamp>.properties* files for additional properties required by the resource adapters

Environment Variables

The path environment variable must be set for the Node Manager process on each machine in the cluster. Select the correct variable name for your environment (LD_LIBRARY_PATH, SHLIB_PATH, LIB_PATH, PATH).

1. *<platform_install_directory>/components/setup/jni/<OS>* must be appended to the variable.

2. Check [INCLUDE_PATHS] section in all *<platform_install_directory>/cluster_deploy/doc/environment_<timestamp>.properties* files for additional paths required by the resource adapters.

Virtual Hosts

There must be virtual host aliases created for the default hosts for each server in the cluster. There must be an alias setup for each *WC_defaulthost* endpoint and *WC_defaulthost_secure* endpoint for all servers in the cluster. If these are not configured the cluster communication will not function properly.

Installing New Packages and Patches

Any updates to the platform server will be available in the *<cluster_deploy_directory>/updates* directory. Updates include patches and new packages installed with Package Manager. Each update will create a new timestamp directory. It will contain *toDeploy* directory. The applications in the *toDeploy* directory must be deployed to the application server. Check for a new *<cluster_deploy_directory>/doc/environment_<timestamp>.properties* file as there may be amended or additional Java properties or system paths that need to be set or removed.

WebLogic

There are two methods for to deploying PASW Collaboration and Deployment Services into a WebLogic cluster:

Scripted Deployment is intended for less advanced users who want assistance with deploying the platform into an application server. Several scripts are provided for automating the deployment process.For more information, see the topic Scripted Deployment on p. 58.

Manual Deployment is intended for advanced users who want precise control over their application server environment. All deployment can be done through the administration console.For more information, see the topic Manual Deployment on p. 63.

Note: If you use a Windows share as the shared file system for installing PASW Collaboration and Deployment Services, you must configure the Node Agent Windows service to run as a Windows user that has access to the share. You must also use the UNC path as opposed to a mapped drive when configuring the installation because mapped drives are not available to Windows services.

Scripted Deployment

Scripted Deployment Files

There are several scripts that can be used to automate the cluster setup and the cluster deployment into WebLogic. These scripts are located in the *<cluster install location>/scripts* directory.

config.ini

The file contains the parameters used by the Jython scripts (described below) to automatically create a cluster. The file contains the following sections and properties:

weblogic

- **home** location of WebLogic home (for example, *C:\bea\weblogic92*).

domain

- **name** name of the domain (for example, *platformDomain*).
- **location** directory to create the domain in (for example, *C:\bea\weblogic92*).

cluster

- **name** name of the cluster (for example, *websphere_cluster*).
- **multiAddr** multicast address for the cluster (for example, 237.0.0.101).
- **multiPort** multicast port for the cluster (for example, 9200)
- **singletonServer** name of the server to deploy applications that can only run on a single server.

servers

- **name** name of the server (for example, *platServer1*).
- **address** host name of the server (for example, *YourHostName*).
- **port** port of the server (for example, 8080)
- **machine** name of the machine the server will run on.
- **domainDir** directory of the local domain
- **platformSharedDir** shared platform installation directory (for example, *\\machine\shared\platform_install*).
- **platformLocalDir** directory of the local platform installation (for example, *C:\platformLocal*).
- **platformOS** operating system, valid values include aix, aix64, hpux64, linux, linux64, solaris64, windows, windows64.
- **platformKeystoreLocation** location of the platform keystore created at install time.
- **platformKeystorePassword** password of the platform keystore created at install time.
- **javaHome** location of Java environment to be used with WebLogic (for example, *C:\bea\jdk150_06*).
- **javaVender** vendor of Java environment. Valid values include Sun and BEA.
- **javaMemoryArgs** memory arguments used to start the server (for example, -Xms128m -Xmx1024m -XX:MaxPermSize=512m).

machines

- **name** machine name (for example, *YourMachineName*).
- **nodeManagerAddr** node manager address (for example, *YourHostName*).
- **nodeManagerPort** node manager port (for example, 5556).

jms

- **target** name of the server to run a JMS server on.

admin

- **server** host name of admin server (for example, *YourHostName*).
- **port** port of admin server (for example, 7001).
- **user** user of admin server (for example, *weblogic*).
- **password** password of admin server (example, *weblogic*).
  platform (prepopulated by setup)
- **database.driver** platform database driver class name.
- **database.user** platform database user.
- **database.password** platform database password (may be encrypted).
- **database.url** platform database URL.
- **deploy.directory** platform *toDeploy* directory.

These configuration parameters are used as follows:

- **platformDeploy.py** Creates a new domain with exactly one cluster. The cluster includes all the servers defined in the "servers" section of *config.ini*. The machines defined in the "machines" section will also be created. An admin server will be created using the configuration in the "admin" section. A new JMS server will be created for each entry in the "jms" section. The configuration parameters in the "platform" section will be used to create the platform datasource and deploy the platform applications into the cluster.
- **platformClean.py** is used to remove the components installed by the *platformDeploy.py* script.

platformDeploy.py

The script is used to deploy platform components to a clustered WebLogic domain as configured in *config.ini*. Script parameters include:

- **all** Deploy everything (default).
- **cluster** Deploy cluster, servers, and shared library.
- **machines** Deploy machines only.
- **servers** Deploy servers only.
- **components** Deploy datasource and JMS components.
- **datasource** Deploy datasource components only.
- **jms** Deploy JMS components only.
- **applications** Deploy platform applications.
- **patch** Deploy updated platform applications.

platformClean.py

The script is used to undeploy platform components from a clustered WebLogic domain as configured in *config.ini*. Script parameters include:

- **all** Undeploy everything (default).
- **cluster** Undeploy cluster, servers, and shared library.
- **machines** Undeploy machines only.
- **servers** Undeploy servers only.
- **components** Undeploy datasource and JMS components.
- **datasource** Undeploy datasource components only.
- **jms** Undeploy JMS components only.
- **applications** Undeploy platform applications.

The executable scripts for running Jython scripts are located in the *<cluster install location>/scripts* directory. They include:

- **setEnv** Sets up the environment.
- **wsadmin** Executes a specified Jython script.
- **installNode** Installs the necessary platform components on the local file system.
- **createTemplate** Creates the managed server template for use in configuring individual nodes.
- **configureTemplate** Helps set the system environment for the individual nodes.
- **deployTemplate** Deploys the managed server template for use in configuring individual nodes.
- **startAdminServer** Starts the admin server for the domain.
- **startNodeManager** Starts the node manager service.

Scripted Deployment

Note: To simplify deployment, it is recommended that the paths to WebLogic home, user domain home, and Java environment be the same on all systems used for nodes in the cluster.

1. Install the same version of WebLogic on each node in the cluster.
2. Set up the platform install directory as a shared directory for each node in the cluster.
3. Run the script to install the local platform components.
   - Open a command prompt to the bin directory.
   - Run installNode <local_path> to install logging configuration files.
   - The logging configuration can be updated by editing the *<local_path>/logging/log4j.xml* file.
4. Update *bin/setEnv* to set the following environment variables.
   - **WL_HOME** Location of your WebLogic install (for example, *C:\bea\weblogic92*).
   - **JAVA_HOME** Location of your JDK install (for example, *C:\bea\jdk150_04*).
   - **DOMAIN_HOME** Location of your WebLogic domain (for example, *C:\bea\user_projects\domains\platformDomain*).

5. If you are using a Java environment other than the one provided with WebLogic, modify the appropriate variable (BEA_JAVA_HOME or SUN_JAVA_HOME) in the *<domain_home>/bin/setDomainEnv script.*

6. Update *scripts/config.ini* to set up your cluster configuration.
   - It is recommended to use the Sun JDK for all servers in the cluster.

7. Run the script to deploy the platform components to a WebLogic cluster. The configuration is read from the *scripts/config.ini* file.
   - Open a command prompt to the bin directory.
   - Run *wlst ../scripts/platformDeploy.py*.
   - The following components are deployed: A JDBC datasource targeted to the cluster; a JDBC persistent stores targeted to a single server; JMS servers targeted to a single server; JMS module targeted to the cluster; a JMS connection factory targeted to the cluster; two JMS Uniform Distributed Queues targeted to the cluster; platform applications (EARs, WARs, and RARs).

8. Copy the jars in the lib directory to the *<domain_home>/lib* directory.

9. Run the *configureTemplate* script to initialize the domain's system environment variables.
   - The process is interactive and requires you to enter the server name as specified in *config.ini*.

10. Modify the *<domain_home>/bin/startWebLogic* script to include a call to the *<domain_home>/bin/setStartUpEnv* script.
    - The inserted call must come at the beginning of the file right before the existing call to the *<domain_home>/bin/setDomainEnv* script.

11. Run the *createTemplate* script to create a template jar for the cluster.

12. Run the *deployTemplate* script on each remote node in the cluster.
    - Ignore the warning about an invalid "Servers, Cluster and Machine" configuration.
    - The process is interactive and requires you to enter the server name as specified in *config.ini*.

13. If any of the paths to WebLogic home, user domain home, or Java environment differ on any of the remote nodes, those nodes will need to have the scripts *<domain_home>/bin/startWebLogic* and *<domain_home>/bin/setDomainEnv* modified to reflect the correct paths for the nodes.

14. Modify *<wl_home>/common/nodemanager/nodemanager.properties* to have the value StartScriptEnabled=true on all cluster servers (default is false).

15. Start the admin server by running the *startAdminServer* script located in the bin directory.
    - Node manager must be running on each node in the cluster.

16. Start all nodes in the cluster using the administration console

Installing New Packages and Patches

Any updates to the platform server will be available in the *updates* directory. Updates include patches and new packages installed via package manager. Each update will create a new timestamp directory. It will contain a *toDeploy* directory. To deploy the updates:

1. Modify the *update.deploy.directory* property in the *config.ini* to point to the *toDeploy* directory inside the newly created timestamp directory.
2. Copy the jars in the lib directory (if any) to *<domain_home>/lib* for each node in the cluster.
3. Copy the scripts to *<domain_home>/bin* for each node in the cluster.
4. Use the WebLogic admin console to restart each server in the cluster.
5. Run the script to update the platform components in the WebLogic cluster.
   - Open a command prompt to the *bin* directory.
   - Run wlst ../scripts/platformDeploy.py patch.
   - All platform applications (EARs, WARs, and RARs) in the *updates* directory will be updated.

Manual Deployment

These instructions provide advanced J2EE users with the information necessary to deploy PASW Collaboration and Deployment Services into a WebSphere application server cluster. It is expected that the cluster has already been configured and is ready for deployment.

The instructions use the following path placeholders:

<platform_install_directory> The root of the shared installation directory for PASW Collaboration and Deployment Services on a single dedicated node. This is the directory containing folders such as setup, platform, and components.

<path_to_keystore_directory> The directory specified during install where the keystore was created.

<cluster_deploy_directory> The directory specified during install where the cluster deploy files were placed. The default location for this directory is *<platform_install_directory>/cluster_deploy*.

<node_local_directory> The root of the local PASW Collaboration and Deployment Services directory on the server nodes in the cluster. This can be any directory but it is suggested that the paths are the same on all servers.

<path_to_domain> The path to the WebLogic application server domain for the PASW Collaboration and Deployment Services installation on each server node. It is suggested that the same domain name is used on all servers.

Shared File System

The root of the *<platform_install_directory>* must be shared across all nodes in the cluster. It is necessary for each node to have read access to this directory and its entire contents. If *<path_to_keystore_directory>* is not *<platform_install_directory>* or one of its subfolders, *<path_to_keystore_directory>* must also be shared across all nodes in the cluster. On Windows, when referencing these directories from remote nodes, it is recommended that UNC paths be used rather than mapped drives.

Logging

Logging must be configured on each node in the cluster. Steps for configuring logging on each node in the cluster:

Create *<node_local_directory>/logging* directory on the local file system.

Copy *<cluster_deploy_directory>/logging/log4j.xml* to *<node_local_directory>/logging*.

Update *<node_local_directory>/logging/log4j.xml* to specify the log file location.

JDK

It is recommended that all servers in the cluster run with the Sun JDK.

Global Libraries

There are several libraries that must be in the server classpath to be globally available to all deployed applications. In order to configure the global libraries, copy the JAR files from *<cluster_deploy_directory>/lib* directory to *<path_to_domain>/lib* directory for each node in the server cluster.

Datasource

A JDBC datasource with the following parameters must be configured and targeted to the server cluster. Do not use any of the preconfigured WebLogic JDBC drivers.

1. JNDI name for the datasource must be set to jdb/spss/PlatformDS.
2. Database driver class name must be set to com.spss.datadirect.jdbc.SPSSDriver.
3. Database JDBC URL, username, and password must be set as appropriate.
4. *Use XA Datasource Interface* property must be set to false.
5. Use initial capacity of 20, increment by 5, maximum of 100.
6. *Test Connections On Reserve* property must be set to true.
7. *Test Table Name* property must be set to SPSSSETUP_PLUGINS.
8. *Connection Creation Retry Frequency* must be set to 20 seconds.

JMS

The following JMS components must be configured.

1. A JDBC persistent store for use with the JMS Server.
2. At least one JMS server must be available within the cluster.
3. JMS connection factory *PlatformJMSConnectionFactory* with the JNDI name *ConnectionFactory*.
   - *Default Targeting Enabled* property must be set to true.
   - *Server Affinity Enabled* property in the load balancing parameters must be set to false.
4. Uniform distributed topic *PASWMessageBus* with the JNDI name *topic/PASWMessageBus* for use by the PASW Collaboration and Deployment Services components.
5. Uniform distributed queue *PASWScoringQueue* with the JNDI name *queue/PASWScoring* for use by the scoring message driven bean *ScoringMDB*.
6. Uniform distributed queue *PASWLogQueue* with the JNDI name *queue/PASWLog* for use by the scoring component.
7. Uniform distributed queue *SPSSAuditQueue* with the JNDI name *queue/SPSSAudit* for use by the auditing message driven bean *AuditMDB*.
8. Uniform distributed queue *SPSSNotificationQueue* with the JNDI name *queue/SPSSNotification* for use by the notification component.
9. Uniform distributed queue *SPSSProcessQueue* with the JNDI name *queue/SPSSProcess* for use by the process component.

   Note: All topics and queues must have the *Default Targeting Enabled* property set to true.

Application Deployment

1. All applications in the *<cluster_deploy_directory>/toDeploy* directory must be deployed to the cluster.
2. All applications in the *<cluster_deploy_directory>/toDeploy/explode* directory must be deployed to the cluster in exploded format.

   Information for resource adapter deployment including JNDI name, Connection Factory Implementation class, and Java Classpath can be found in the [RAR_CONNECTION_FACTORIES] section of the *<cluster_deploy_directory>/doc/environment_<timestamp>.properties* files.

   The workflow application will not be initialized by default in a cluster environment. In order to use the workflow component, the following system property must be set on only one server in the cluster: -Dcom.spss.workflow.active.override=true.

   The workflow component must not be initialized on multiple servers in the cluster.

Java System Properties

The following Java system properties must be set appropriately for each server in the cluster:

1. JVM memory arguments must be set appropriately (recommended 1024m minimum max heap size).

2. -Dcom.spss.configsys.installBase.override=<platform_install_directory>.

4. -Dlog4j.configuration=<node_local_directory>/logging/log4j.xml.

5. -Dplatform.keystore.file=<path_to_keystore_directory>.

6. -Dplatform.keystore.password=<keystore_password>.
   - A non-cleartext version of the keystore password can be found in the *<platform_install_directory>/platform/setupinfo.xml* file.

7. The *java.library.path* must contain the following:
   - *<platform_install_directory>/components/setup/jni/<OS>*.
   - For each resource adapter deployed: *<path_to_domain>/<rar_name>/bin*.
   - Additional native libraries may be added to the *java.library.path* if required.

8. Check the [JAVA_PROPERTIES] section in all *<cluster_deploy_directory>/doc/environment_<timestamp>.properties* files for additional properties required by the resource adapters.

Environment Variables

The path environment variable must be set for the Node Manager process on each machine in the cluster. Select the correct variable name for your environment (LD_LIBRARY_PATH, SHLIB_PATH, LIB_PATH, PATH).

1. *<platform_install_directory>/components/setup/jni/<OS>* must be appended to the variable.

2. Check [INCLUDE_PATHS] section in all *<cluster_deploy_directory>/doc/environment_<timestamp>.properties* files for additional paths required by the resource adapters.

Installing New Packages and Patches

Any updates to the platform server will be available in the *<cluster_deploy_directory>/updates* directory. Updates include patches and new packages installed with Package Manager. Each update will create a new timestamp directory. It will contain *toDeploy* directory. The applications in the *toDeploy* directory must be deployed to the application server. Check for a new *<cluster_deploy_directory>/doc/environment_<timestamp>.properties* file as there may be amended or additional Java properties or system paths that need to be set or removed.

Load Balancer Configuration

A software- or hardware-based load balancer must be configured for accessing the repository in a clustered environment. Both WebLogic and WebSphere application servers provide built-in software-based load-balancer utilities.

WebLogic Apache Plugin

WebLogic ships with a plugin that can be used with the Apache Web Server to act as a load balancer.

The plugin setup includes the following steps:

1. Install Apache Web server. For more information, see Apache documentation at *http://httpd.apache.org/docs/2.0/install.html*
2. Configure the WebLogic plugin. For more information, see WebLogic documentation. It can be accessed online at *http://e-docs.bea.com/wls/docs92/plugins/apache.html*.

   Plugin configuration requires editing the corresponding section of the configuration *httpd.conf* file to specify the nodes in the cluster, for example:

```
# Sample from httpd.conf

LoadModule weblogic_module modules/mod_wl_20.so

<IfModule mod_weblogic.c>
  Debug                ON
  DebugConfigInfo      ON
  KeepAliveEnabled     ON
  KeepAliveSecs        30
  MatchExpression      WebLogicCluster=WLC1:8080,WLC2:8080,WLC3:8080|Debug=ON
  WLForwardUriUnparsed ON
</IfModule>
```

   Note: Unparsed URI forwarding (*WLForwardUriUnparsed* parameter) must be enabled to prevent errors when accessing repository resources with names that contain double-byte characters and spaces.

IBM HTTP Server for WebSphere Application Server

IBM HTTP server can be configured to act as a load balancer.

The configuration includes the following steps:

1. Install IBM HTTP Server. For more information, see WebSphere documentation. It can be accessed online at *http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp*
2. Use the administration console to create a Web server object.
3. Use the administration console to generate a plugin descriptor and propagate it to IBM HTTP Server.
4. Start IBM HTTP Server.

Job Step Failover

PASW Collaboration and Deployment Services 4.1 supports rerunning of failed job steps only in clustered WebSphere environments. To ensure that in cases of a cluster node failure a job step will be processed by other nodes, the cluster must be configured for JMS failover. There are several ways to enable JMS failover.

Setting JMS message reliability level to "Assured Persistent"

1. In WebSphere Administration Console, open
   Resources
   JMS
   Queue Connection factory
   PlatformJMSConnection
2. From the ConnectionFactory, change the Quality of Service to "Assured Persistent."
3. Save the configuration. You may need to restart each node in the cluster. With this change all JMS queue messages in PASW Collaboration and Deployment Services will be guaranteed delivery. The drawback of this method is that it is very resource-intensive.

Setting ConnectionFactory as bus destination

1. In WebSphere Administration Console, open
   Resources
   JMS
   Queue Connection factory
   PlatformJMSConnection
2. From the ConnectionFactory, change the Quality of Service to "As Bus Destination."
3. Save the configuration.
4. In WebSphere Administration Console, open
   Service Integration
   Buses
   Platform Bus
   Destinations
   SPSSProcessQueue_Bus
5. Uncheck the "Enable producers to override default reliability." The bus destination will ensure each message is "Assured Persistent" and guaranteed delivery and processing.

Workload Balancing

By default, WebSphere does not do workload balancing of the JMS messages. PASW Collaboration and Deployment Services configuration creates a single shared JMS engine for the cluster, and the entire messaging engine will failover to another node in the cluster. It is possible to configure the cluster running PASW Collaboration and Deployment Services for workload balancing by using additional JMS engines. For more information, see WebSphere documentation.

Chapter
7

Single EAR File Deployment

PASW Collaboration and Deployment Services consist of many individual J2EE applications deployed as WAR files. To simplify the management in clustered application server environments running many other J2EE applications, PASW Collaboration and Deployment Services applications can be combined in single EAR (Enterprise Archive) file. This appendix provides the instructions for creating such a file for deployment into WebSphere and WebLogic application servers. In order to successfully complete these instructions, you must understand the JAR specification, how it is applied to enterprise archives, as well as understand how to manually deploy archives into J2EE servers.

WebSphere

These instructions are for creating a single EAR file of all PASW Collaboration and Deployment Services WAR files for deployment into a WebSphere server cluster. Note that due to issues with WebSphere class loading, the resulting EAR file will not include such components as RAR files and EJB modules

The instructions use the following path placeholders:

<platform_install_directory> The root of the shared installation directory for PASW Collaboration and Deployment Services on a single dedicated node. This is the directory containing folders such as setup, platform, and components.

<path_to_keystore_directory> The directory specified during install where the keystore was created.

<cluster_deploy_directory> The directory specified during install where the cluster deploy files were placed. The default location for this directory is *<platform_install_directory>/cluster_deploy*.

<node_local_directory> The root of the local PASW Collaboration and Deployment Services directory on the server nodes in the cluster. This can be any directory but it is suggested that the paths are the same on all servers.

<ws_cell> WebSphere server cell name.

<new_ear_name> The name of the EAR file that you will create by following these instructions.

The instructions assume that PASW Collaboration and Deployment Services 4.1 has already been installed into a WebSphere cluster. For more information, see the topic Clustering in Chapter 6 on p. 46. If you are installing into a “clean” WebSphere cluster (i.e., a cluster with no existing applications), then you may want to consider using the scripted install which will install all of the

necessary components automatically. For more information, see the topic Scripted Deployment in Chapter 6 on p. 48. Since a scripted install deploys all applications by default, you will either need to remove the WAR files that it installs after the script completes or remove the WAR files from the *<cluster_deploy_directory>\toDeploy* directory prior to running the scripted install.

If you have a WebSphere cluster that is not "clean" (i.e., you have existing applications installed) or you prefer to do the installation yourself, you should install using the manual deployment instructions. For more information, see the topic Manual Deployment in Chapter 6 on p. 51.

The deploy PASW Collaboration and Deployment Services into a WebSphere cluster as a single EAR file, follow these general step steps:

1. Create the archive directory structure.
2. Create *application.xml*.
3. Create the EAR file.
4. Deploy the EAR.
5. Deploy other modules (optional).

Detailed information for these steps is provided below.

EAR Directory Structure

An EAR file is a compressed archive that follows the conventions of the JAR specification. In order to create a single EAR, you must create a directory structure containing PASW Collaboration and Deployment Services components, and compress these directories into a single archive. Several items are located at the root of the directory structure, such as the META-INF folder and WAR files.

- The META-INF folder will contain a manifest and deployment descriptors (more on the details of this later).
- The WAR files can simply be placed into the root of the directory structure.

Note that you will find the required WAR files in the *<cluster_deploy_directory>/toDeploy* directory.

The directory structure of the EAR file must be as follows:

```
\__ META-INF
    \__ application.xml
    \__ MANIFEST.MF
\__ admin.war
\__ birt-viewer.war
\__ clientinstall.war
\__ config.war
\__ cr-ws.war
\__ cr_web.war
\__ er-extension.war
\__ groupman.war
```

```
\__ jmxhttp.war
\__ langman.war
\__ notification.war
\__ paswTagLib.war
\__ peb-job.war
\__ peb-mmd.war
\__ peb-scoring.war
\__ peb-svws.war
\__ peb.war
\__ pem.war
\__ pev.war
\__ process.war
\__ processui.war
\__ reporting-ws.war
\__ root.war
\__ scoring.war
\__ search-ws.war
\__ security-ws.war
\__ security.war
\__ spsscop-ws.war
\__ userpref-ws.war
```

application.xml

Next, *application.xml* file must be created. The file lets the application server know what modules are available and provides configuration information about each module. Module entries will instruct the application server where to find the module within our EAR, so you will have to create an entry in the *application.xml* file for each of these modules. Just keep in mind that the <context-root> element is the name of the WAR file, but without the extension. So for example, the <context-root> element for *admin.war* becomes admin.

Below is an example *application.xml* file that you can use as a starting point. Please bear in mind that you should verify that all entries are in place, because this example file only covers the basic modules that ship with PASW Collaboration and Deployment Services. Note that it covers only the modules that ship with PASW Collaboration and Deployment Services 4.1 and does not include the modules that are not included in the distribution, for example, PASW Modeler scoring provider.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE application PUBLIC "-//Sun Microsystems, Inc.//DTD J2EE Application 1.3//EN"
  "http://java.sun.com/dtd/application_1_3.dtd">
<application>
  <display-name>CDS EAR</display-name>
  <description>CDS Application</description>

  <!–core –>
  <module>
    <web>
      <web-uri>admin.war</web-uri>
      <context-root>admin</context-root>
    </web>
```

```
</module>

<module>
  <web>
    <web-uri>birt-viewer.war</web-uri>
    <context-root>birt-viewer</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>clientinstall.war</web-uri>
    <context-root>clientinstall</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>config.war</web-uri>
    <context-root>config</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>cr-ws.war</web-uri>
    <context-root>cr-ws</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>cr_web.war</web-uri>
    <context-root>cr_web</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>er-extension.war</web-uri>
    <context-root>er-extension</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>groupman.war</web-uri>
    <context-root>groupman</context-root>
  </web>
</module>

<module>
```

```
  <web>
    <web-uri>jmxhttp.war</web-uri>
    <context-root>jmxhttp</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>langman.war</web-uri>
    <context-root>langman</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>notification.war</web-uri>
    <context-root>notification</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>paswTagLib.war</web-uri>
    <context-root>paswTagLib</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>peb.war</web-uri>
    <context-root>peb</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>peb-job.war</web-uri>
    <context-root>peb-job</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>peb-svws.war</web-uri>
    <context-root>peb-svws</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>peb-mmd.war</web-uri>
    <context-root>peb-mmd</context-root>
```

```
    </web>
  </module>

  <module>
    <web>
      <web-uri>pem.war</web-uri>
      <context-root>pem</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>pev.war</web-uri>
      <context-root>pev</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>processui.war</web-uri>
      <context-root>processui</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>reporting-ws.war</web-uri>
      <context-root>reporting-ws</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>root.war</web-uri>
      <context-root>root</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>search-ws.war</web-uri>
      <context-root>search-ws</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>security.war</web-uri>
      <context-root>security</context-root>
    </web>
  </module>
```

```
  <module>
    <web>
      <web-uri>security-ws.war</web-uri>
      <context-root>security-ws</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>spsscop-ws.war</web-uri>
      <context-root>spsscop-ws</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>userpref-ws.war</web-uri>
      <context-root>userpref-ws</context-root>
    </web>
  </module>

  <!–scoring –>
  <module>
    <web>
      <web-uri>scoring.war</web-uri>
      <context-root>scoring</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>peb-scoring.war</web-uri>
      <context-root>peb-scoring</context-root>
    </web>
  </module>


  <!–process –>
  <module>
    <web>
      <web-uri>process.war</web-uri>
      <context-root>process</context-root>
    </web>
  </module>

</application>
```

Deploying the EAR File

Once the single EAR file is ready, you can use the WebSphere console to deploy it. You must also complete additional configuration steps as described below. Note that these instructions cover only the modules that ship with PASW Collaboration and Deployment Services 4.1 and additional configuration may be required for any other modules.

- The shared library must be added to the new single EAR application, as well as any other modules (such as RAR and EJB modules). For more information, see the topic Deploying Other Modules (Optional) on p. 76.
- All PASW Collaboration and Deployment Services J2EE applications should be deployed using *Parent Last* classloading (this is not the default). *Parent Last* classloading should be used for both the deployment classloader as well as the web module class loader. If the applications are not deployed in this manner they will fail to run correctly. In WebSphere Web console, click Manage Modules for the new application and set all WAR files to use *Application First* class loading.
- You should restart your application server after everything has been configured to ensure the native libraries are loaded correctly.

Deployment Problems

- You may encounter out of memory errors when deploying the EAR file. The problem may be corrected by increasing the values of max heap size (-Xmx) and the max perm size (MaxPermSize) parameters in *<WebSphere_root>/deploytool/itp/ejbdeploy.sh*.
- You may also encounter out of memory errors in WebSphere Deployment Manager after deploying the EAR file.
  - In the Administrative Console, open

System Administration
  Deployment Manager
    Process Definition
      Java Virtual Machine

  - Specify 256 for Initial Heap Size and 1500 for Maximum Heap Size.
  - Save your changes and restart the Deployment Manager.

Deploying Other Modules (Optional)

This section provides additional information for deploying EJB and RAR modules not found in the manual install instructions. If you successfully completed a scripted installation, these modules will be already installed.

EJB Deployment

When manually deploying the preexisting EAR files, several EJB JNDI Bindings will need to be defined. The bindings are as follows:

Binding Name	Module	Activation Specification
ScoringMDB	scoring-ejb	pasw/ScoringMDBAS

Binding Name	Module	Activation Specification
ScoringNotificationsMDB	scoring-ejb	pasw/ScoringNotificationsMDBAS
ScoreLogMDB	scoring-ejb	pasw/ScoreLogMDBAS
SPSSauditMDB	auditmdb	spss/AuditMDBAS
ProcessEventMDB	process-ejb	spss/ProcessEventMDBAS

Enter the JNDI name for non-MDB bean. These values should be set to the name found in the EJB column. The same value must be used for mapping EJB references to beans. For example:

- ScoringTimerSessionBean
- CalendarMonitorTimedObject
- MessageMonitorTimedObject

JCA Resource Adapters

One or more resource adapters will be deployed to PASW Collaboration and Deployment Services application cluster. In order to do this, they must first be deployed to a single node and then copied to the cluster scope using the wsadmin command *AdminTask.copyResourceAdapter*. A deep copy should be performed. The following settings should be used for the resource adapter:

- Archive Path: *<platform_install_directory>/platform/resourceAdapters*
- Classpath:
  - ${SPSSPLATFORM_DIR}/platform/resourceAdapters/<name>.rar
  - ${SPSSPLATFORM_DIR}/platform/globalLibraries/<global_dependency>.jar (if applicable)
- Native Path:
  - ${SPSSPLATFORM_DIR}/platform/resourceAdapters/<name>.rar

Example JCA Resource Adapter Settings for SPSS Smart Score:

- Archive Path

C:/installs/CADS41/platform/resourceAdapters/smartscore.rar

- Class Path

${SPSSPLATFORM_DIR}/platform/resourceAdapters/smartscore.rar
${SPSSPLATFORM_DIR}/platform/globalLibraries/smartscore-client.jar
${SPSSPLATFORM_DIR}/platform/globalLibraries/smartscorej-client.jar

- Native path (this has no value initially)

${SPSSPLATFORM_DIR}/platform/resourceAdapters/smartscore.rar

J2C Connection Factory

A J2CConnectionFactory should be created for use by the resource adapter. The JNDI name indicated by the resource adapter should be used for the connection factory. The specifics for configuring the resource adapters can be found in the [RAR_CONNECTION_FACTORIES] section of the *<cluster_deploy_directory>/doc/environment_<timestamp>.properties* files. It

may also be necessary to add Class Path and/or Native Path entries if they are specified in the *<cluster_deploy_directory>/doc/environment_<timestamp>.properties* files.

Example J2C Connection Factory Settings

When you deploy the JCA Resource Adapters, the J2C connection factory is automatically created, but it contains incorrect information. Using the Smart Score JCA Resource Adapter as an example, you should change:

- name = com.spss.smartscore.ra.SmartScoreConnectionFactory to name = SmartScoreConnectionFactory
- JNDI name = eis/com.spss.smartscore.ra.SmartScoreConnectionFactory to JNDI name = SmartScoreConnectionFactory

SPSSSharedLibrary Settings

- classpath

```
${SPSSPLATFORM_DIR}/setup/resources/websphere
${SPSSPLATFORM_DIR}/setup/lib/DataDirectAdapter.jar
${SPSSPLATFORM_DIR}/setup/lib/MFsqlserver.jar
${SPSSPLATFORM_DIR}/setup/lib/MFdb2.jar
${SPSSPLATFORM_DIR}/setup/lib/MForacle.jar
${SPSSPLATFORM_DIR}/setup/lib/MFmysql.jar
${SPSSPLATFORM_DIR}/setup/lib/MFinformix.jar
${SPSSPLATFORM_DIR}/setup/lib/MFsybase.jar
${SPSSPLATFORM_DIR}/setup/lib/jt400.jar
${SPSSPLATFORM_DIR}/setup/lib/log4j.jar
${SPSSPLATFORM_DIR}/setup/lib/commons-logging.jar
${SPSSPLATFORM_DIR}/setup/lib/icu4j.jar
${SPSSPLATFORM_DIR}/setup/lib/security-global.jar
${SPSSPLATFORM_DIR}/setup/lib/search-global.jar
${SPSSPLATFORM_DIR}/setup/lib/spsslic.jar
${SPSSPLATFORM_DIR}/setup/lib/spsslic7-global.jar
${SPSSPLATFORM_DIR}/setup/lib/userpref-global.jar
${SPSSPLATFORM_DIR}/components/process/workunit/process-native.jar
${SPSSPLATFORM_DIR}/setup/lib/spsswebsphere.jar
${SPSSPLATFORM_DIR}/platform/globalLibraries/XFjc.jar
${SPSSPLATFORM_DIR}/platform/globalLibraries/XFssl14.jar
${SPSSPLATFORM_DIR}/platform/globalLibraries/smartscore-client.jar
${SPSSPLATFORM_DIR}/platform/globalLibraries/smartscorej-client.jar
```

A WebSphere variable (SPSSPLATFORM_DIR) should be set up for each node in the cluster and point to *<platform_install_directory>*.

- Native Path

```
${SPSSPLATFORM_DIR}/components/setup/jni/windows
$(APP_INSTALL_ROOT)/<ws_cell>/<new_ear_name>/smartscore.rar
${SPSSPLATFORM_DIR}/components/smartscore/win32
```

Installing New Packages and Patches

Any updates to the platform server will be available in the *<cluster_deploy_directory>/updates* directory. Updates include patches and installing new packages via package manager. Each update will create a new timestamp directory. It will contain a *toDeploy* directory. The applications in the *toDeploy* directory should be deployed to the application server. Check to see if there is a new *<cluster_deploy_directory>/doc/environment_<timestamp>.properties* file as there may be amended or additional Java properties or system paths that need to be set or removed.

WebLogic

These instructions are for creating a single EAR file of all PASW Collaboration and Deployment Services WAR, JAR and RAR files for deployment into a WebLogic server cluster.

The instructions use the following path placeholders:

<platform_install_directory> The root of the shared installation directory for on a single dedicated node. This is the directory containing folders such as setup, platform, and components.

<path_to_keystore_directory> The directory specified during install where the keystore was created.

<cluster_deploy_directory> The directory specified during install where the cluster deploy files were placed. The default location for this directory is *<platform_install_directory>/cluster_deploy*.

<node_local_directory> The root of the local PASW Collaboration and Deployment Services directory on the server nodes in the cluster. This can be any directory but it is suggested that the paths are the same on all servers.

<path_to_domain> The path to the WebLogic application server domain for the PASW Collaboration and Deployment Services installation on each server node. It is suggested that the same domain name is used on all servers.

The instructions assume that PASW Collaboration and Deployment Services 4.1 has already been installed into a WebLogic cluster.

The procedure is as follows:

1. Create the EAR directory structure.
2. Update *application.xml*.
3. Update ejb-link references.
4. Add *weblogic-application.xml*.
5. Update JSTL references.
6. Create the EAR file.
7. Deploy the EAR.

Detailed information for these steps is provided below.

EAR Directory Structure

An EAR file is a compressed archive that follows the conventions of the JAR specification. In order to create a single EAR, you must create a directory structure containing PASW Collaboration and Deployment Services components and compress these directories into a single archive. Several items are located at the root of the directory structure, such as the META-INF folder and WAR files.

- The META-INF folder will contain a manifest and deployment descriptors (more on the details of this later).
- Each of the EJB folders reflects the contents of the existing EAR files that ship with PASW Collaboration and Deployment Services. You should expand these existing EAR files at the root and ensure that the folder names match the name of the existing EAR (not including the .ear extension). For example, *auditmdb.ear* becomes *auditmdb*. You must remove the resulting META-INF folder since its contents will be represented in our single ear deployment descriptors (more on the details of this later).
- The WAR files can simply be placed into the root of the directory structure.

Note that you will find the required EAR, WAR, and RAR files in the *<cluster_deploy_directory>/toDeploy* and *<cluster_deploy_directory>/toDeploy/explode* directories. In normal circumstances the folders contained in the *<cluster_deploy_directory>/toDeploy/explode* directory are deployed in exploded format, however, in this case we will convert the contents of these directories back into archives. So for example, the contents of the folder called *<cluster_deploy_directory>/toDeploy/explode/smartscore.rar* should be added to a compressed archive called *smartscore.rar* and included at the root directory along with the WAR files. Take care when doing this because you want the contents of the folder to be located at the root of this new archive. A common mistake is to archive the folder itself, rather than the contents of the folder.

The directory structure of the EAR file must be as follows:

```
\__ META-INF
    \__ application.xml
    \__ MANIFEST.MF
    \__ weblogic-application.xml
\__ auditmdb
    \__ audit-component.jar
    \__ auditMDB.jar – note that the MDB is located in here, and its manifest points to jars in the same dir
    \__ cacheservice.jar
    \__ castor.jar
    \__ commons-codec.jar
    \__ config.jar
    \__ jakarta-oro.jar
    \__ language.jar
    \__ rdmcon.jar
    \__ security-access.jar
    \__ security-action.jar
    \__ security-authentication.jar
    \__ security-capabilities.jar
```

```
    \__ security-client.jar
    \__ util.jar
\__ process-ejb
    \__ antlr.jar
    \__ audit-component.jar
    \__ axis.jar
    \__ cacheservice.jar
    \__ castor.jar
    \__ cmor.jar
    \__ commons-codec-1.2.jar
    \__ commons-collections-3.1.jar
    \__ commons-discovery-0.2.jar
    \__ commons-io-1.0.jar
    \__ communication.jar
    \__ config.jar
    \__ cop-client.jar
    \__ groupman.jar
    \__ jakarta-oro.jar
    \__ jmxhttp.jar
    \__ json-lib.jar
    \__ language.jar
    \__ notification.jar
    \__ process-ejb.jar – note that the MDB is located in here, and its manifest points to jars in the same dir
    \__ process-ejb.war
    \__ process.jar
    \__ rdmcon.jar
    \__ repository-client.jar
    \__ security-access.jar
    \__ security-action.jar
    \__ security-capabilities.jar
    \__ security-client.jar
    \__ setup-component.jar
    \__ transformations.jar
    \__ util.jar
    \__ velocity.jar
    \__ wsdl4j-1.5.1.jar
\__ scoring-ejb
    \__ castor.jar
    \__ config.jar
    \__ jakarta-oro.jar
    \__ language.jar
    \__ logging.jar
    \__ rdmcon.jar
    \__ repository-client.jar
    \__ scoring-ejb.jar – note that the ejb and MDB is located in here, and its manifest points to jars in the same dir
    \__ scoring-timer.war – Note that the war for this item is located in the scoring-ejb dir
    \__ security-capabilities.jar
    \__ security-client.jar
    \__ setup-component.jar
    \__ util.jar
\__ admin.war
\__ birt-viewer.war
```

```
\__ clientinstall.war
\__ config.war
\__ cr-ws.war
\__ cr_web.war
\__ er-extension.war
\__ groupman.war
\__ jmxhttp.war
\__ langman.war
\__ notification.war
\__ paswTagLib.war
\__ peb-job.war
\__ peb-mmd.war
\__ peb-scoring.war
\__ peb-svws.war
\__ peb.war
\__ pem.war
\__ pev.war
\__ process.war
\__ processui.war
\__ reporting-ws.war
\__ root.war
\__ scoring.war
\__ search-ws.war
\__ security-ws.war
\__ security.war
\__ smartscore.rar
\__ spsscop-ws.war
\__ userpref-ws.war
```

application.xml

Next, *application.xml* file must be created. The file lets the application server know what modules are available and provides configuration information about each module. Some of the information is already provided in the *application.xml* file the preexisting EAR files, but they cannot be used as-is because the directory structure of the new EAR is not the same. The <web-uri> element needs to be updated to reflect the fact that the new EAR we are creating contains a folder. So for example, the <web-uri> element for the *scoring-ejb.ear* specifies *scoring-timer.war*, but in the new EAR we would need to specify this as *scoring-ejb/scoring-timer.war* instead. Note the additional folder directory followed by a slash. Module entries will instruct the application server where to find the module within our EAR, so you will have to create an entry in the *application.xml* file for each of these modules. Note that the <context-root> element is the name of the WAR file, but without the extension. So for example, the <context-root> element for *admin.war* becomes admin.

Below is an example *application.xml* file that you can use as a starting point. Note that you should verify that all entries are in place, because this example file only covers the basic modules that ship with PASW Collaboration and Deployment Services. Note that it covers only the modules that ship with PASW Collaboration and Deployment Services 4.1 and does not include the modules that are not included in the distribution, for example, the PASW Modeler scoring provider.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE application PUBLIC "-//Sun Microsystems, Inc.//DTD J2EE Application 1.3//EN"
  "http://java.sun.com/dtd/application_1_3.dtd">
<application>
  <display-name>CDS EAR</display-name>
  <description>CDS Application</description>

  <!–core –>
  <module>
    <web>
      <web-uri>admin.war</web-uri>
      <context-root>admin</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>birt-viewer.war</web-uri>
      <context-root>birt-viewer</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>clientinstall.war</web-uri>
      <context-root>clientinstall</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>config.war</web-uri>
      <context-root>config</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>cr-ws.war</web-uri>
      <context-root>cr-ws</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>cr_web.war</web-uri>
      <context-root>cr_web</context-root>
    </web>
  </module>

  <module>
    <web>
```

```
      <web-uri>er-extension.war</web-uri>
      <context-root>er-extension</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>groupman.war</web-uri>
      <context-root>groupman</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>jmxhttp.war</web-uri>
      <context-root>jmxhttp</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>langman.war</web-uri>
      <context-root>langman</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>notification.war</web-uri>
      <context-root>notification</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>paswTagLib.war</web-uri>
      <context-root>paswTagLib</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>peb.war</web-uri>
      <context-root>peb</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>peb-job.war</web-uri>
      <context-root>peb-job</context-root>
    </web>
```

```
</module>

<module>
  <web>
    <web-uri>peb-svws.war</web-uri>
    <context-root>peb-svws</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>peb-mmd.war</web-uri>
    <context-root>peb-mmd</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>pem.war</web-uri>
    <context-root>pem</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>pev.war</web-uri>
    <context-root>pev</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>processui.war</web-uri>
    <context-root>processui</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>reporting-ws.war</web-uri>
    <context-root>reporting-ws</context-root>
  </web>
</module>

<module>
  <web>
    <web-uri>root.war</web-uri>
    <context-root>root</context-root>
  </web>
</module>

<module>
```

```
    <web>
      <web-uri>search-ws.war</web-uri>
      <context-root>search-ws</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>security.war</web-uri>
      <context-root>security</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>security-ws.war</web-uri>
      <context-root>security-ws</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>spsscop-ws.war</web-uri>
      <context-root>spsscop-ws</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>userpref-ws.war</web-uri>
      <context-root>userpref-ws</context-root>
    </web>
  </module>

  <!–scoring –>
  <module>
    <ejb>scoring-ejb/scoring-ejb.jar</ejb>
  </module>

  <module>
    <web>
      <web-uri>scoring.war</web-uri>
      <context-root>scoring</context-root>
    </web>
  </module>

  <module>
    <web>
      <web-uri>peb-scoring.war</web-uri>
      <context-root>peb-scoring</context-root>
    </web>
  </module>
```

```
<module>
  <web>
    <web-uri>scoring-ejb/scoring-timer.war</web-uri>
    <context-root>scoring-timer</context-root>
  </web>
</module>

<module>
  <connector>smartscore.rar</connector>
</module>

<!–process –>
<module>
  <web>
    <web-uri>process.war</web-uri>
    <context-root>process</context-root>
  </web>
</module>

<module>
  <ejb>process-ejb/process-ejb.jar</ejb>
</module>

<module>
  <web>
    <web-uri>process-ejb/process-ejb.war</web-uri>
    <context-root>process-ejb</context-root>
  </web>
</module>

<!–audit –>
<module>
  <ejb>auditmdb/auditMDB.jar</ejb>
</module>

</application>
```

Updating EJB-Link References

Because the EAR you are building has directories (for example, *scoring-ejb*) that contain the contents for each preexisting EAR (e.g. scoring-ejb.ear), you will need to update some configuration files in order to include these directories, so that WebLogic can find the code it needs to run PASW Collaboration and Deployment Services. This is necessary because WebLogic will refuse to start an application if it can not find the JAR that contains an EJB. You will need to modify the <ejb-link> references defined in any WAR file that is contained in the preexisting EAR files. The <ejb-link> reference is a URI that points from the root of the EAR to the EJB jar, followed by a # symbol and the name of the bean. To make this change, you will need to modify the *web.xml* file contained within the WAR. Find the <ejb-link> element, and modify the value so that it includes the path to the jar file that contains the EJB as described above.

For example, the *scoring-timer.war* will need to be modified so that its *web.xml* file uses the correct URI for the *scoring-ejb.jar*. Note that the EJB-link had to be changed from ScoringTimerSessionBean to /scoring-ejb/scoring-ejb.jar#ScoringTimerSessionBean. So the *web.xml* file in this example looks like the following after being modified:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
Application 2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <servlet>
    <servlet-name>ScoringTimerStartup</servlet-name>
    <servlet-class>com.spss.scoring.internal.web.servlets.ScoringTimerStartup</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <ejb-ref>
    <description>This is a reference for the J2EE timer we use
    in the scoring service.</description>
    <ejb-ref-name>ScoringTimerSessionBean</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.spss.scoring.internal.ejb.ScoringTimerHome</home>
    <remote>com.spss.scoring.internal.ejb.ScoringTimer</remote>
    <ejb-link>/scoring-ejb/scoring-ejb.jar#ScoringTimerSessionBean</ejb-link>
  </ejb-ref>
</web-app>
```

The *process-ejb.war* also requires a similar modification. The <ejb-link> elements are changed as follows:

Old Value	New Value
CalendarMonitorTimedObject	/process-ejb/process-ejb.jar#CalendarMonitorTimedObject
MessageMonitorTimedObject	/process-ejb/process-ejb.jar#MessageMonitorTimedObject

At the time this readme was written, only scoring-timer.war and process-ejb.war are the only WAR files that need to change. You should verify that there are no other WAR files contained within an preexisting EAR. IF there are others, you need to make a similar modification as described in this section.

Updating JSTL Library References

Some PASW Collaboration and Deployment Services WAR components depend on JSTL 1.2 library, an optional shared library in WebLogic located in *<path_to_weblogic_home>/common/deployable-libraries/jstl-1.2.war*. There are two ways to resolve this dependence:

1. Install JSTL library as a shared library on the node. Shared libraries can be used by any J2EE application running on WebLogic. Use this method if installing this library will not interfere with any other application in the cluster. For instructions on installing JSTL library, consult WebLogic documentation.

2. Integrate JSTL library into the following PASW Collaboration and Deployment Services WAR components:
   - *er-extension.war*
   - *paswTagLib.war*
   - *peb.war*
   - *peb-job.war*
   - *peb-mmd.war*
   - *peb-scoring.war*

   To integrate the JSTL library into a WAR component:
   - Remove the following JSTL library reference element from the component's *WEB-INF/weblogic.xml* descriptor.

```
<library-ref>
  <library-name>jstl</library-name>
  <specification-version>1.2</specification-version>
  <implementation-version>1.2</implementation-version>
  <exact-match>false</exact-match>
</library-ref>
```

   - Add *glassfish.jstl_1.2.0.1.jar* to *WEB-INF/lib* folder of the WAR.

weblogic-application.xml

Next, you will need to add a weblogic-application.xml file to the META-INF directory of your new EAR. This file instructs WebLogic to create a hierarchy of class loaders, and limits the "scope" of each module. It is also used to inform WebLogic about what classes should always be loaded from the application (rather than from the system class loader).

Using separate class loaders keeps the modules from loading classes elsewhere in the application server. The structure found in the following weblogic-application.xml file ensures that the classes within the module are loaded first and if not found searched for higher up in the class loader hierarchy. Note that the scoring specific code shares a class loader between the scoring-timer.war and scoring-ejb.jar. The same is true for the process specific code, which shares a class loader between process-ejb.jar and process-ejb.war. All other modules are kept separate from each other.

We use "prefer-application-packages" in the following weblogic-application.xml to avoid issues where weblogic includes classes for its own use (for example, org.mozilla.*), which conflict with the versions we include in *birt-viewer.war*.

At the time this readme was written, the following example includes an entry for each module that ships with PASW Collaboration and Deployment Services. Optional installs (e.g. Modeler score provider) and additional modules not known at the time this readme was written are not included, so you should verify the file contents prior to deployment.

```
<?xml version="1.0" encoding="UTF-8"?>
<weblogic-application xmlns="http://www.bea.com/ns/weblogic/90"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bea.com/ns/weblogic/90 http://www.bea.com/ns/weblogic/90/weblogic-application.xsd">
<classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>scoring-ejb/scoring-timer.war</module-uri>
    </module-ref>
    <module-ref>
      <module-uri>scoring-ejb/scoring-ejb.jar</module-uri>
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>auditmdb/auditMDB.jar</module-uri>
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>process-ejb/process-ejb.jar</module-uri>
    </module-ref>
    <module-ref>
      <module-uri>process-ejb/process-ejb.war</module-uri>
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>admin.war</module-uri>
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>birt-viewer.war</module-uri>
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>clientinstall.war</module-uri>
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>config.war</module-uri>
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>cr-ws.war</module-uri>
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>cr_web.war</module-uri>
```

```
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>er-extension.war</module-uri>
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>groupman.war</module-uri>
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>jmxhttp.war</module-uri>
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>langman.war</module-uri>
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>notification.war</module-uri>
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>paswTagLib.war</module-uri>
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>peb.war</module-uri>
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>peb-job.war</module-uri>
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>peb-mmd.war</module-uri>
    </module-ref>
  </classloader-structure>
  <classloader-structure>
    <module-ref>
      <module-uri>peb-scoring.war</module-uri>
    </module-ref>
  </classloader-structure>
```

```
<classloader-structure>
  <module-ref>
    <module-uri>peb-svws.war</module-uri>
  </module-ref>
</classloader-structure>
<classloader-structure>
  <module-ref>
    <module-uri>pem.war</module-uri>
  </module-ref>
</classloader-structure>
<classloader-structure>
  <module-ref>
    <module-uri>pev.war</module-uri>
  </module-ref>
</classloader-structure>
<classloader-structure>
  <module-ref>
    <module-uri>process.war</module-uri>
  </module-ref>
</classloader-structure>
<classloader-structure>
  <module-ref>
    <module-uri>processui.war</module-uri>
  </module-ref>
</classloader-structure>
<classloader-structure>
  <module-ref>
    <module-uri>reporting-ws.war</module-uri>
  </module-ref>
</classloader-structure>
<classloader-structure>
  <module-ref>
    <module-uri>root.war</module-uri>
  </module-ref>
</classloader-structure>
<classloader-structure>
  <module-ref>
    <module-uri>scoring.war</module-uri>
  </module-ref>
</classloader-structure>
<classloader-structure>
  <module-ref>
    <module-uri>search-ws.war</module-uri>
  </module-ref>
</classloader-structure>
<classloader-structure>
  <module-ref>
    <module-uri>security-ws.war</module-uri>
  </module-ref>
</classloader-structure>
<classloader-structure>
  <module-ref>
```

```
        <module-uri>security.war</module-uri>
      </module-ref>
    </classloader-structure>
    <classloader-structure>
      <module-ref>
        <module-uri>spsscop-ws.war</module-uri>
      </module-ref>
    </classloader-structure>
    <classloader-structure>
      <module-ref>
        <module-uri>userpref-ws.war</module-uri>
      </module-ref>
    </classloader-structure>
    <classloader-structure>
      <module-ref>
        <module-uri>smartscore.rar</module-uri>
      </module-ref>
    </classloader-structure>
  </classloader-structure>
  <prefer-application-packages>
    <package-name>org.mozilla.*</package-name>
  </prefer-application-packages>
</weblogic-application>
```

Deploying the EAR

Once the EAR is ready, you can use the WebLogic web interface to deploy the EAR. You should choose to deploy the EAR as an application, target all servers in the cluster and leave all of the other values at their default values.

Installing New Packages and Patches

Any updates to the platform server will be available in the *<cluster_deploy_directory>/updates* directory. Updates include patches and installing new packages via package manager. Each update will create a new timestamp directory. It will contain *toDeploy* and *lib* directories. The applications in the *toDeploy* directory should be deployed to the application server. The JARs in the *lib* directory should be copied to *<path_to_domain>/lib*for each node in the cluster. Check to see if there is a new *<cluster_deploy_directory>/doc/environment_<timestamp>.properties* file as there may be amended or additional Java properties or system paths that need to be set or removed.

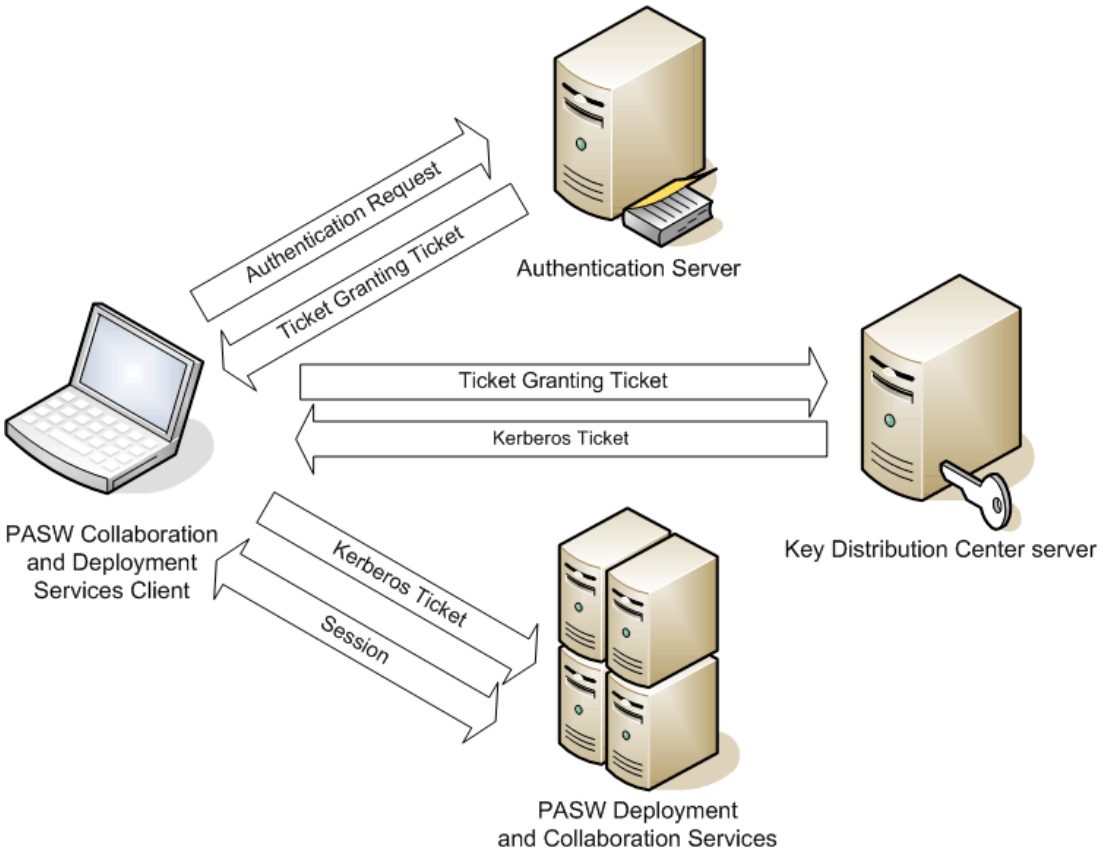
Chapter
8

Single Sign-On

Single sign-on (SSO) is a method of access control that enables a user to log in once and gain access to the resources of multiple software systems without being prompted to log in again. PASW Collaboration and Deployment Services provides the single sign-on capability by initially authenticating users through an external directory service based on **Kerberos** security protocol, and subsequently using the credentials in all PASW Collaboration and Deployment Services applications, for example, Deployment Manager, Deployment Portal, or a portal server without additional authentication.

Note: Single sign-on is not allowed for browser-based Deployment Manager.

Figure 8-1
PASW Collaboration and Deployment Services SSO architecture



For example, if PASW Collaboration and Deployment Services is used in conjunction with Windows Active directory, to enable single sign-on it is necessary to configure **Kerberos Key Distribution Center (KDC)** service. The service will supply session tickets and temporary session keys to users and computers within an Active Directory domain. The KDC must run on each domain controller as part of Active Directory Domain Services (AD DS). When single sign-on is enabled, PASW Collaboration and Deployment Services applications log into to a Kerberos domain and use Kerberos tokens for Web services authentication. It is strongly recommended that SSL communication be configured for the repository if single sign-on is enabled.

PASW Collaboration and Deployment Services single sign-on configuration is performed on the Server Administration tab of Deployment Manager. For more information, see PASW Collaboration and Deployment Services administrator documentation.

The following prerequisites must be in place before single sign-on is set up for PASW Collaboration and Deployment Services:

- Directory authentication server must be configured. Authentication can be based on Microsoft Active Directory, OpenLDAP directory, or IBM i profile directory.
- Kerberos Key Distribution Center server must be configured.
- Credential delegation must be enabled for the Kerberos Service Principal on the Kerberos Key Distribution Center server. The procedure for enabling credential delegation will be different depending on your directory server and Kerberos environment.
- Kerberos credential delegation must also be enabled on all client systems.
- For Windows client systems, the registry must be updated for Kerberos LSA access.
- Depending on the database used with the repository, the database may need to be configured for single sign-on.
- Depending on the application server used with repository, it may be necessary to update the application server configuration.
- Windows client systems must have HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\Kerberos\ registry value updated. For more information, Updating Windows Systems Registry for Single Sign-On
- For thin-client access to PASW Collaboration and Deployment Services (for example, with Deployment Portal), the Web browser must have Simple and Protected GSS-API Negotiation (SPNEGO) enabled.

Updating Windows Systems Registry for Single Sign-On

PASW Collaboration and Deployment Services installation Disk 1 includes registry update files for configuring Windows XP SP2, Windows Vista, and Windows 2003 systems for Kerberos-based single sign-on. The files are as follows:

- */PASW/Kerberos/Win2003_Kerberos.reg*
- */PASW/Kerberos/WinXPSP2_Kerberos.reg*

For Windows Vista systems, use the *Win2003_Kerberos.reg* file.

The registry files allow the system administrator to push registry changes to all systems on the network that must have single sign-on access to PASW Collaboration and Deployment Services.

Chapter 9

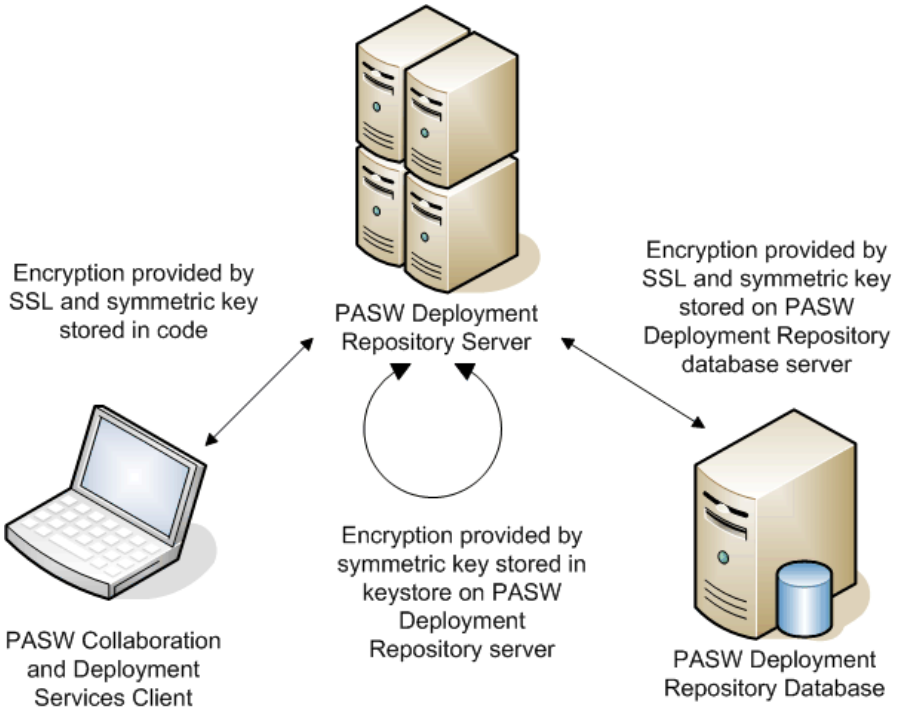
FIPS 140–2 Compliance

The Federal Information Processing Standard (FIPS) Publication 140-2, FIPS PUB 140-2, is a U.S. government computer security standard used to accredit cryptographic modules. The document specifies the requirements for cryptography modules which include both hardware and software components, corresponding to four different levels of security that are mandated for organization that do business with the U.S. government. PASW Collaboration and Deployment Services can be configured to provide Security Level 1 as specified by FIPS 140-2.

Security configuration for FIPS 140-2-compliance must follow these guidelines:

- Communications between the repository and client applications must use SSL for transport layer security of general data transfers. Additional AES encryption is provided for credential passwords using a shared key stored in the application code. For more information, see the topic Using SSL to Secure Data Transfer in Chapter 10 on p. 100.
- The repository server uses AES algorithm with the key stored in a keystore on the server file system to encrypt passwords in PASW Collaboration and Deployment Services configuration files, application server configuration files, security provider configuration files, etc.
- Communications between the repository server and the database server can optionally use SSL for transport layer security for general data transfer. AES encryption is provided for credential passwords, configuration passwords, user preference passwords, etc. using a shared key stored in a keystore on the database server file system.

Figure 9-1
PASW Collaboration and Deployment Services FIPS 140-2-compliant security setup



Repository Configuration

PASW Collaboration and Deployment Services repository configuration for FIPS 140-2-compliance must follow these guidelines:

- The database must be set up to accept SSL communications; the JCE encryption module must also be configured.
- If the repository is installed on UNIX, the default JRE must be set up with a JCE module.
- The application server JRE must also be set up with a JCE module.
- The application server must be configured to accept SSL communications; a JCE module must also be configured.
- If the repository is installed on Windows, you must exit the installation at setup screen, configure a JCE module, then restart the setup and select to run in FIPS 140-2-compliant mode on the appropriate screen. For more information about the installation wizard, see Installing the Repository on p. 18
- If PASW Collaboration and Deployment Services is deployed into a clustered environment, keystore must be replicated to all nodes in the cluster.
- The JREs that are being used by SPSS Inc. server applications interacting with PASW Collaboration and Deployment Services, such as PASW Statistics Server and PASW Modeler Server, must have SSL certificates installed.

Desktop Client Configuration

For PASW Collaboration and Deployment Services desktop client applications, such as Deployment Manager, JCE encryption module must be enabled for the JRE used to run the applications. The JRE must have SSL certificates installed.

Browser Configuration

- Mozilla Firefox can be configured to run in FIPS 140-2 compliant mode by modifying the application options. For more information, see *http://support.mozilla.com/en-US/kb/Configuring+Firefox+for+FIPS+140-2*.
- Internet Explorer configuration requires enabling Windows cryptography and modifying the browser settings. For more information, see*http://support.microsoft.com/kb/811833*.
- Apple Safari cannot be used in FIPS 140-2 compliant mode.

Chapter
10

Using SSL to Secure Data Transfer

Security Sockets Layer (SSL) is a protocol for encrypting data transferred between two computers. SSL ensures that communication between the computers is secure. SSL can encrypt the authentication of a username/password and the contents of an exchange between a server and client.

How SSL Works

SSL relies on the server's public and private keys, in addition to a public key certificate that binds the server's identity to its public key.

- When a client connects to a server, the client authenticates the server with the public key certificate.
- The client then generates a random number, encrypts the number with the server's public key, and sends the encrypted message back to the server.
- The server decrypts the random number with its private key.
- From the random number, both the server and client create the session keys used for encrypting and decrypting subsequent information.

The public key certificate is typically signed by a certificate authority. Certificate authorities, such as VeriSign and Thawte, are organizations that issue, authenticate, and manage security credentials contained in the public key certificates. Essentially, the certificate authority confirms the identity of the server. The certificate authority usually charges a monetary fee for a certificate, but self-signed certificates can also be generated.

Securing Client-Server and Server-Server Communications with SSL

The main steps in securing client-server and server-server communications with SSL are:

- Obtain and install the SSL certificate and keys.
- If desired, install unlimited strength encryption on the client computers.
- If using a self-signed certificate, copy the certificate on the client computer.
- Instruct end users to enable SSL when connecting to the server.

Note: Occasionally a server product acts as a client. An example is PASW Statistics Server connecting to the repository. In this case, PASW Statistics Server is the *client*.

Obtain and Install SSL Certificate and Keys

- Obtain an SSL certificate and key file. There are two ways you can do this:
  - Purchase them from a public certificate authority (such as VeriSign or Thawte). The public certificate authority signs the certificate to verify the server that uses it.
  - Generate the key and certificate files with an internal self-signed certificate authority. OpenSSL provides a certificate management tool for this purpose.
- Install the SSL certificate and keys on the application server. For additional information on how the keys and certificate interoperate with a specific application server, see the original vendor's documentation. Note that you may be required to add the certificate and keys to the Java keystore.

Install Unlimited Strength Encryption

The Java Runtime Environment shipped with the product has US export-strength encryption enabled. For enhanced security of your data, we recommend that this is upgraded to unlimited-strength encryption.

- Download the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 5.0 from *http://java.sun.com/javase/downloads/index.jsp*.
- Unzip the downloaded file.
- Copy the two *.jar* files *local_policy.jar* and *US_export_policy.jar* into *<installation folder>/jre/lib/security*, where *<installation folder>* is the folder in which you installed the product.

Copy the Certificate File to Client Computers

Note: Skip this step if you are using a certificate that is signed by a certificate authority.

If you are using a self-signed certificate, you need to copy the certificate to the *client* computers. Be aware that a server computer may also act as a client. An example is PASW Statistics Server connecting to the repository. In this case, PASW Statistics Server is the *client*, and therefore you need to copy the certificate for the repository server to the PASW Statistics Server.

- Copy *root.pem* to the following location on the client computers. By default, all SPSS Inc. client products look in this location for trusted self-signed certificate files. If you would like to use another location, create an `SSL_CERT_DIR` environment variable and set the value of the variable to the location.

  Windows. *C:\Documents and Settings\All Users\Application Data\SPSSInc\certificates*

  If you already copied a *root.pem* file to the client for another SPSS Inc. product, append the certificate information from the new server to the existing *root.pem* file. This file is a text file so you can copy and paste the certificate.

Add the Certificate to Client Keystore (For Connections to PASW Collaboration and Deployment Services)

Note: Skip this step if you are using a certificate that is signed by a certificate authority.

If you using SSL to connect to PASW Collaboration and Deployment Services and you are using self-signed certificates, you need to add the certificate to the client's Java keystore. The following steps are completed on the client computer.

- Open a command prompt and change directories to the following location, where <*product install dir*> is the directory in which you installed the product:

```
<product install dir>/jre/bin
```

- Enter the following command:

```
keytool -import -alias <alias name> -file <path to cert> -keystore <path to key store>
```

Where <*alias name*> is an arbitrary alias for the certificate, <*path to cert*> is the full path to the certificate, and <*path to key store*> is the full path to the Java keystore, which may be <*product install dir*>*/lib/security/jssecacerts* or <*product install dir*>*/lib/security/cacerts*.

- When prompted, enter the keystore password, which is `changeit` by default.
- When prompted about trusting the certificate, enter `yes`.

Instruct End Users to Enable SSL

When end users connect to the server through a client product, they need to enable SSL in the dialog box for connecting to the server. Be sure to tell your users to select the appropriate check box.

URL Prefix Configuration

If PASW Collaboration and Deployment Services is set up for SSL access, the value of the URL Prefix configuration setting must be modified as follows:

1. Log into PASW Collaboration and Deployment Services using browser-based Deployment Manager.
2. Open *URL Prefix* configuration option.

   Configuration
   Setup
   URL Prefix

3. Set the value of the prefix to https instead of http and set the port value to the SSL port number. For example:

   [default]
   http://<hostname>:<port>
   [SSL-enabled]
   https://<hostname>:<SSLport>

Securing LDAP with SSL

Lightweight Directory Access Protocol (LDAP) is an Internet Engineering Task Force (IETF) standard for exchanging information between network directories and databases containing any level of information. For systems requiring additional security, LDAP providers, such as Microsoft's Active Directory, can operate over Secure Socket Layer (SSL), provided that the Web or application server supports LDAP over SSL. Using SSL in conjunction with LDAP can ensure that login passwords, application information, and other sensitive data are not hijacked, compromised, or stolen.

The following example illustrates how to enable LDAPS using Microsoft's Active Directory as a security provider. For more specific information on any of the steps or to find details that address a particular release of the security provider, see the original vendor documentation.

1. Verify that Active Directory and the Enterprise Certificate Authority are installed and functioning.
2. Use the certificate authority to generate a certificate, and import the certificate into the certificate store of the Deployment Manager installation. This allows the LDAPS connection to be established between the repository and an Active Directory server.

   To configure Deployment Manager for secure Active Directory connections, verify that a connection exists to the repository.
3. Launch the Deployment Manager.
4. From the Tools menu, choose Server Administration.
5. Log in to a previously defined administered repository server.
6. Double-click the Configuration icon for the server to expand the hierarchy.
7. Double-click the Security Providers icon to expand the hierarchy.
8. Double-click the Active Directory security provider.
9. Enter configuration values for the instance of Active Directory with security certificates installed.
10. Select the Use SSL check box.
11. Note the name in the Domain User field. Subsequent logins using Active Directory are authenticated using SSL.

    For additional information about installing, configuring, and implementing LDAPS on a particular application server, see the original vendor's documentation.

Chapter
11

Updating the Repository

Occasionally it may be necessary to install updates for the repository as such updates are made available. It may also be necessary to install optional components that extend repository functionality to support additional content types, security providers, etc., or install Deployment Manager updates which will be pushed to clients when they access the server.

Updates are deployed on the repository server as compressed files with **.package* extension in the *<PASW Collaboration and Deployment Services Installation Directory>/staging/* directory with the Package Manager utility. A number of optional packages, including Clementine 12.x adaptors, Coherence cache provide, SiteMinder security provider, etc., are available in the */PASW/optional* directory of installation Disk 1.

Installing Packages

Package Manager utility can be used as a GUI application or as a command line application. It can also be called in batch mode by other applications to install their package files into PASW Collaboration and Deployment Services. The repository must be stopped prior to installing packages.

Note: If WebSphere application server is used with PASW Collaboration and Deployment Services, it must be running while packages are being installed. WebSphere must be stopped and restarted prior to running Package Manager.

The user must have administrator-level privileges to be able to install packages. The deployed packages are located in the *<PASW Collaboration and Deployment Services Installation Directory>/staging/*.

To prevent the newer version of a package from being overwritten by an older version, Package Manager performs a version check. Package manager also checks for prerequisite components to ensure that they are installed and their versions are equal to or newer than the required version. If any of these checks fail, the package is marked as missing prerequisites in the dialog pane nut can still be installed. However, it is not recommended to install packages that failed dependencies checks.

Note: Dependency checks failures cannot be overridden if Package Manager is called in batch mode.

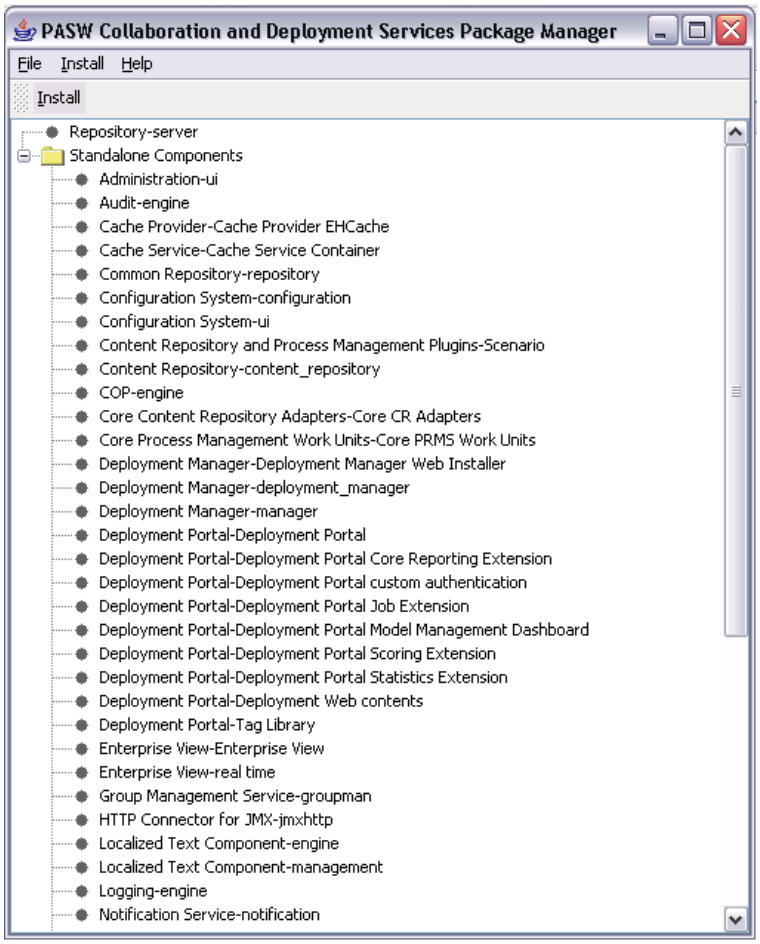
To install a package using the GUI interface:

1. Navigate to *<PASW Collaboration and Deployment Services Installation Directory>/setup/*.
2. Depending on the operating system, execute *packagemanager.bat* on Windows or *packagemanager.sh* on UNIX.
3. When prompted, enter the user name and password.

Figure 11-1
Admin Login

4. Click OK to login. The Package Manager dialog box appears.

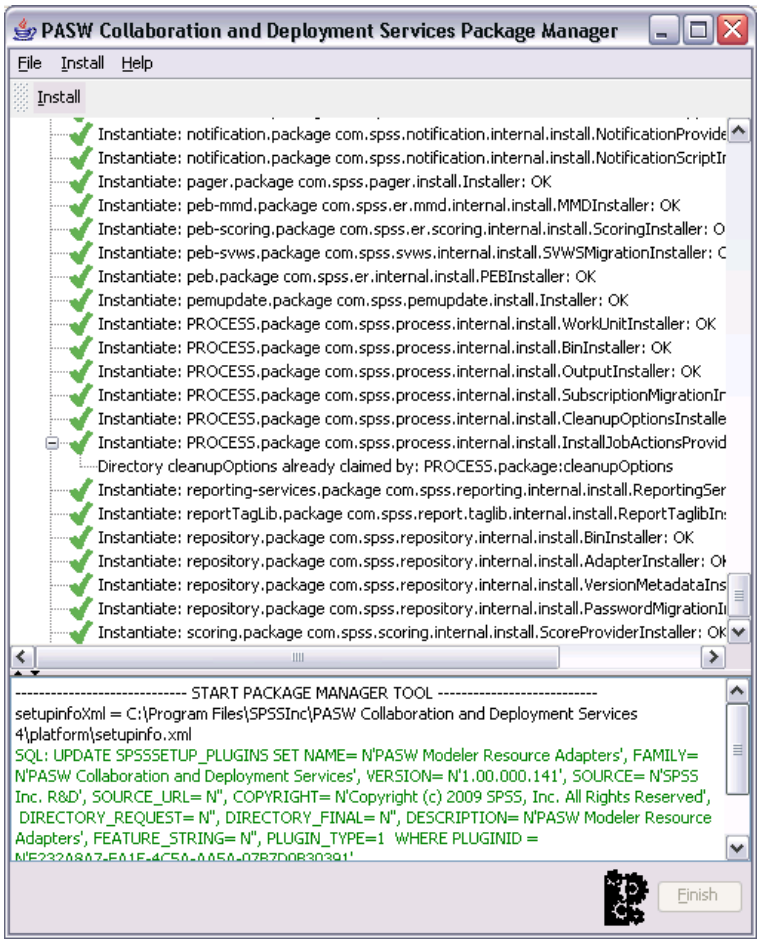
Figure 11-2
Package Manager



5. From the Install menu, select Install.
6. From the installation path, navigate to the location of the package file.
7. Select the package and click OK. The installation status panel appears.

If failed dependencies are detected, the panel displays the Install packages with failed dependencies check box. Select the check box and click OK to continue the installation, or click Cancel to abort.

Figure 11-3
Package Manager installation



Installation log can be found in *<PASW Collaboration and Deployment Services Installation Directory>/setup/logs/setup.log*.

8. Click Finish when the installation is complete. If errors occur during installation, they are displayed in red in the bottom pane. To close Package Manager dialogue box, click Abort.

To install a package from the command line:

1. Navigate to *<PASW Collaboration and Deployment Services Installation Directory>/setup/*.
2. Depending on the operating system, execute *clipackagemanager.bat* on Windows or *clipackagemanager.sh* on UNIX.
3. When prompted, enter the user name and password.

Note: The password is not masked when it is entered in the command prompt.

4. Type the install command and press Enter. The command must include the `install` option and the path of the package in quotes, as in the following example:

   install 'C:\dir one\package1.package'

   If failed dependencies are detected, you will be presented with a choice to ignore the failures and continue the installation or abort.

5. When the installation is completed, use exit command to exit Package Manager.

   Note: To display more command line install options, type `help` and press Enter key. The option include:

   - info "<package path>" Display information for a specified package file
   - install "<package path>" Install the specified package files into PASW Collaboration and Deployment Services
   - tree Display installed package tree information

Chapter
12

Logging Services

Logging tools are essential when troubleshooting existing problems as well as when planning preventive maintenance activities. As system and application events are generated, administrative personnel can be alerted when warning thresholds are reached or critical system events occur. Additionally, verbose information output can be stored in a text file or Syslog record for analysis at a later time.

The repository uses the **log4j** package for handling log information. Log4j is Apache Software Foundation's logging solution for **J2EE** applications. The log4j approach permits logging control using an XML-based configuration file; the application binary does not have to be modified. For a comprehensive discussion of log4j, see the log4j Web site (*http://logging.apache.org/log4j/docs/*).

The location of the *log4j.xml* configuration file varies, depending on the host application server:

- JBoss—<*JBoss installation directory*>*server**default**conf*.
- WebLogic—<*Repository installation directory*>*SPSSDomain**lib*\. Note that log4j components used by PASW Collaboration and Deployment Services for logging on to WebLogic are part of the repository installation.
- WebSphere—<*Repository installation directory*>*setup**resources**websphere*.

This file controls both the destination and the amount of log output. Configuration of log4j is handled by modifying this file to define **appenders** for log destinations and to route **logger** output to those appenders.

Appenders

Log output can be sent to a variety of destinations. In log4j, the destination is referred to as an **appender**. Table 12-1 describes the appenders available in log4j.

Table 12-1
Log4j appenders

Appender class	Description
org.apache.log4j.ConsoleAppender	*System.out* or *System.err* streams
org.apache.log4j.FileAppender	Log file
org.apache.log4j.DailyRollingFileAppender	Log file that is automatically backed up at a specified frequency
org.apache.log4j.RollingFileAppender	Log file that is automatically backed up at a specified size
org.apache.log4j.net.SMTPAppender	E-mail notification of log events
org.apache.log4j.jdbc.JDBCAppender	Database for log events
org.apache.log4j.net.JMSAppender	Notification of log events using Java Messaging Service

Appender class	Description
org.apache.log4j.lf5.LF5Appender	Swing-based logging console
org.apache.log4j.nt.NTEventLogAppender	Appends log events to NT event logs
org.apache.log4j.net.SocketAppender	Remote log server
org.apache.log4j.net.SocketHubAppender	Set of remote log servers
org.apache.log4j.net.SyslogAppender	Syslog daemon
org.apache.log4j.net.TelnetAppender	Read-only socket that can be monitored using TCP/IP
org.apache.log4j.ext.SNMPTrapAppender	Log4j extension that sends SNMP traps

The configuration file defines appenders using the appender element. This definition includes a name and class specification, plus any appender-specific parameters. The following example illustrates a *ConsoleAppender*. For more information about the child elements of appender, see the log4j documentation.

```
<appender name="CONSOLE" class="org.apache.log4j.ConsoleAppender">
   <errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
   <param name="Target" value="System.out"/>
   <param name="Threshold" value="INFO"/>
   <layout class="org.apache.log4j.PatternLayout">
      <!–The default pattern: Date Priority [Category] Message\n –>
      <param name="ConversionPattern" value="%d{ABSOLUTE} %-5p [%c{1}] %m%n"/>
   </layout>
</appender>
```

By default, repository uses two appenders:

- *FILE*, a *DailyRollingFileAppender* that sends the log to a file named *server.log* in the JBoss log folder. At midnight, the year, month, and day are appended as a suffix to the filename, and a new *server.log* file begins recording log events for the next day.
- *CONSOLE*, a *ConsoleAppender* that sends the log to the *System.out* stream for display in a console window.

In addition, the configuration file includes a definition for a *DailyRollingFileAppender* named *FILE-MM*. This appender corresponds to a file named *mm.log* in the JBoss log folder that is similar to the *server.log* file. However, *FILE-MM* can be used for repository loggers to separate log information for the application from log information for the application server. The *FILE-MM* appender appears below:

```
<appender name="FILE-MM" class="org.jboss.logging.appender.DailyRollingFileAppender">
   <errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
   <param name="File" value="${jboss.server.home.dir}/log/mm.log"/>
   <param name="Append" value="false"/>
   <!–Rollover at midnight each day –>
   <param name="DatePattern" value="'.'yyyy-MM-dd"/>
   <layout class="org.apache.log4j.PatternLayout">
      <!–The default pattern: Date Priority [Category] Message\n –>
      <param name="ConversionPattern" value="%-5p [%c] %m%n"/>
   </layout>
</appender>
```

Defining Appenders

To define an appender:

1. Open the *log4j.xml* configuration file in a text editor.
2. Locate the appender element that corresponds to the logging destination you want to employ. If the appender element is commented out of the file, remove the comment symbols (<!−and −>) that enclose the appender.
3. If the configuration file does not contain the desired appender, create a new appender element. Assign a name and specify the class for the desired log destination. See Table 12-1 on p. 108.
4. Modify the content of the appender element as needed to reflect your system and network settings.
5. Save the file.

   The repository automatically updates to reflect the changes. A restart of the server is not needed.

Loggers

Loggers represent application systems that generate log output. For each logger, the *log4j* configuration file specifies both the amount of information logged and the destination for that information.

Logger names typically consist of a series of text strings separated by periods corresponding to the names of software components, such as *com.spss.process*. This naming convention defines a hierarchy of parent/child relationships for loggers. For example, the *com.spss.cmor* logger is a child of the *com.spss* logger, which itself is a child of the *com* logger. The exception to this rule is the *root* logger, which is an ancestor of all loggers in the system. The table below lists the loggers available in the repository.

Table 12-2
Loggers

Logger	Description
root	Root logger
com.spss.cmor	Repository events
com.spss.security	Security events
com.spss.process	Job scheduling events

In the configuration file, the root and category elements define logger properties. The root element defines log destinations for all loggers in the system. The category element allows specification of behavior for particular loggers. The category specification for the repository follows:

```
<category name="com.spss.cmor">
   <priority value="WARN"/>
</category>
<category name="com.spss.security">
   <priority value="WARN"/>
</category>
<category name="com.spss.process">
   <priority value="WARN"/>
```

```
</category>
```

The priority element defines a logging level for the corresponding logger. The level controls the amount of information logged.

Logging Levels

The amount of information contained in the log output is controlled by the logging level. Valid levels include:

- **FATAL**. Severe errors that cause the application to fail.
- **ERROR**. *FATAL*-level errors plus errors resulting from specific requests that allow the application to continue functioning.
- **WARN**. *ERROR*-level errors plus suboptimal or unexpected events.
- **INFO**. *WARN*-level errors plus status messages reflecting general application processes.
- **DEBUG**. *INFO*-level errors plus detailed status messages used for application debugging purposes.

Levels are hierarchical; each level includes all of the output for levels above it. For example, setting the logging level to *WARN* results in all *WARN*, *ERROR*, and *FATAL* output being logged.

Configure the logging level for a particular logger using the priority element in the configuration file. This element uses the value attribute to specify the logging level. The following example sets the level for the *com.spss.cmor* logger to *WARN*:

```
<category name="com.spss.cmor">
   <priority value="WARN"/>
</category>
```

By default, the repository logs all information at the *WARN* level.

In the absence of a priority element for a logger, that logger inherits the level of the nearest ancestor. As a result, the logging level for all repository loggers could be set to the same level using the *com.spss* parent logger:

```
<category name="com.spss">
   <priority value="WARN"/>
</category>
```

Modifying Logging Levels

To modify a logging level:

1. Open the *log4j.xml* configuration file in a text editor.
2. Locate the category element for the logger to be modified.
3. Change the value for the child priority element to the desired logging level. For more information, see Logging Levels on p. 111
4. Save the file.

The repository automatically updates to reflect the changes. A restart of the server is not needed.

Routing Logs

Routing log information involves associating appenders with loggers. **Loggers** define the amount of information being logged; **appenders** define the destination for the information. In the *log4j* configuration file, use the appender-ref element to assign appenders to loggers.

In log4j, all log output is sent to any appenders associated with the *root* logger. The repository uses the *CONSOLE* and *FILE* appenders for the *root* logger, defined by using two appender-ref elements as children of the root element.

```
<root>
   <appender-ref ref="CONSOLE"/>
   <appender-ref ref="FILE"/>
</root>
```

To send the output for a specific logger to an alternative destination, add an appender-ref element as a child of the category element for the logger. For example, suppose we wanted to isolate all job scheduling log output in a single file. Using the appender-ref element, we add a reference to the *FILE-MM* appender for the *com.spss.process* logger.

```
<category name="com.spss.process">
   <priority value="WARN"/>
   <appender-ref ref="FILE-MM"/>
</category>
```

In this case, the job scheduling log is sent to the *FILE-MM* appender plus any appenders defined for the *root* category. To prevent the scheduling log from going to the *root* appenders, set the additivity attribute for the appender-ref element to *false*.

```
<category name="com.spss.process">
   <priority value="WARN"/>
   <appender-ref ref="FILE-MM" additivity="false"/>
</category>
```

Assigning Appenders

To assign an appender to a logger:

1. Open the *log4j.xml* configuration file in a text editor.
2. Locate the category element for the logger to be modified.
3. Add a child appender-ref element. Supply an appender name as the value for the ref attribute. Use the additivity attribute to control whether the logger should continue to send information to the root appenders.
4. Save the file.

   The repository automatically updates to reflect the changes. A restart of the server is not needed.

Chapter
13

Import Tool

The Import Tool allows you to populate the repository with any file type, such as PASW Modeler streams. The PASW Modeler Stream Library is a set of streams that can help you learn to browse, view, and retrieve stored items. It also provides a methodology for organizing your own data mining work product. The streams provide a set of reusable data mining techniques that can help you to formulate solutions to business problems quickly.

The PASW Modeler Stream Library includes a set of sample streams organized into the following categories:

- Data Preparation—After cataloging data resources, data preparation includes any cleaning, selecting, constructing, integrating, and formatting of data.
- Data Understanding—An exploratory stage in which data are examined using plots, histograms, and basic summary statistics.
- Modeling—Information is extracted from the data using sophisticated analytical methods to select modeling techniques, generate test designs, and build and assess models.

Once the repository is installed and functioning, the streams included in the PASW Modeler Stream Library are imported into the database using the Import Tool Windows batch file or UNIX shell script. These import tools process streams, models, and standard output files included in the PASW Modeler Stream Library, but they can also be used to handle any data object stored in a file system.

Directory Structure

When the repository is installed, the Import Tool is included with the application. The tools are located in the */applications/ImportTool* directory within the repository installation directory and are described in the table below.

Table 13-1
File location and directory structure

Name	Description
ModelerStreamLibrary	Directory that contains subdirectories for: *Data Preparation* *Data Understanding* *Modeling* Each of the subdirectories contains the streams (*.str* files) that are imported into the database.
lib	Directory containing *.jar* files used by the application. These should not be changed or deleted.

Name	Description
importTool.bat	Windows batch file for importing data objects. When using the Import Tool on a supported Windows system, execute this file to populate the database.
importTool.sh	UNIX shell script for importing data objects. When using the Import Tool on a supported UNIX platform, execute this file to populate the database.
repository.properties	Configuration file containing system-specific attributes. Some attributes are required and must be changed before using the Import Tool.

Before You Begin

Before working with the Import Tool, the repository must be installed. The batch file and shell script both attempt to use the repository-installed JRE if *JAVA_HOME* is not set. You will need to change the value of the *MM_INSTALL_HOME* variable in the Windows batch file (*importTool.bat*) or the UNIX shell script (*importTool.sh*).

Before running the batch file or shell script, set the installation path of the repository. To set the installation path:

1. Open *importTool.bat* or *importTool.sh* in a text editor.
2. Change the value of *MM_INSTALL_HOME* to match the installation path of Deployment Manager.
3. Save and close the file.

Customizing Properties

Edit the *repository.properties* file using a text editor to customize the application properties. This file must specify the repository server name and login information. You may specify all of the properties for the connection, but the defaults are adequate in most cases.

Table 13-2
Description of repository.properties file

Name	Description
repository.host	The name of the server. *Required*.
repository.username	The name of the user being authenticated. *Required*.
repository.password	The associated password for the user being authenticated. *Required*.
streams.directory	The directory location of the files to load.
author.names	Assigned list of comma-separated names to apply to the imported files. Note that these are randomly assigned.
version.labels	Assigned version names for imported files. These are assigned in the order in which they are listed. The first time a file is imported, the first label is applied. The second time a file is imported, the second label is applied, and so on.
repository.port	The port number the server is using. By default, this value is 80. This must be changed if other applications are using the default or if the application server is assigned to another port.
repository.protocol	The protocol used. By default, this value is http.
repository.context	The URL context string.

Populating the Repository

To populate the repository, start the PASW Collaboration and Deployment Services server and execute the Windows batch file or run the shell script under a supported UNIX platform.

Note: Solaris users need to enter chmod +x importTool.sh before executing the shell script.

Verbose (and lengthy) INFO messages appear as the utility populates the PASW Collaboration and Deployment Services repository. Specific output varies for each installation but is similar to the following output:

```
Using JAVA_HOME installation at C:\SPSS\ModelManager\jre\
INFO  [main] - Creating URL with http://localhost:8080/cr-ws/services/ContentRepository
INFO  [main] - Starting directory: ClementineStreamLibrary
INFO  [main] - Validating repository connection
INFO  [main] - Connecting as admin
INFO  [main] - Service connection established.
INFO  [main] - Looking for topic: '/'
INFO  [main] - Found topic: /
INFO  [main] - Looking for topic: '//CRISP-DM'
INFO  [main] - Didn't find it.
INFO  [main] - Creating new topic: CRISP-DM in /
INFO  [main] - Created new topic with ID: 0a0b989f00b1b4c3000001028d5651008007
```

Note: The output should contain only INFO messages; output prefaced with ERROR indicates a configuration or system failure. Verify the settings in *repository.properties* and run the batch file or shell script again.

Assigning Topics

During stream import, the name of the file is used to assign a CRISP-DM topic to the stream. Topics provide searchable metadata to facilitate finding streams in the repository.

The first letter of the filename determines the topic assigned to the file. The table below describes the relationship between the first letter in the name and the assigned topics.

Table 13-3
Naming convention for topics

First letter	Assigned topics
p	CRISP-DM > Data Preparation
e	CRISP-DM > Data Understanding
m	CRISP-DM > Modeling
	CRISP-DM > Evaluation
d	CRISP-DM > Deployment

Files with names beginning with any other character are not automatically assigned a topic.

Verifying File Import

After the batch file or shell script has finished processing, verify that the files have been successfully imported using PASW Modeler or Deployment Manager.

PASW Modeler User Interface

To verify that files were imported correctly:

1. From the PASW Modeler user interface, establish a connection to the repository. For specific instructions, see the PASW Modeler documentation.
2. After a connection has been established, verify that the correct directory structure appears.

Deployment Manager User Interface

To verify that files were imported correctly:

1. From the Deployment Manager user interface, establish a connection to the repository.
2. In the Content Explorer, expand *Content Repository* by clicking the + icon.
3. Verify that the correct directory structure appears.

Appendix

A

SAP NetWeaver Configuration Notes

Additional configuration steps are required after the repository is installed on a system running SAP NetWeaver application server. They include modifying the system *PATH* variable and adding custom parameters to SAP NetWeaver configuration. For information about installing the repository, see Installing the Repository on p. 18

System Path Variable

- *PATH* must also be modified to include all the paths enumerated in the [INCLUDE_PATHS] section of the generated properties file *<PASW Collaboration and Deployment Services installation directory>/setup/resources/netweaver/config/environment.properties*.

```
...
[INCLUDE_PATHS]
E:/PASW_CADS/components\modeler\bin
E:/PASW_CADS/setup/resources/netweaver/bin/psapi.rar
E:/PASW_CADS/setup/resources/netweaver/bin/csp.rar
...
```

Custom Parameters

- The following custom parameters must be defined with Application Server Configuration Tool.

Configtool
  VM Parameter
    System

  - **com.spss.psapi.session.serverInstallationDirectory**
  - **com.spss.psapi.extensions.autoloadDirectory**
  - **csp.installationDirectory**

  The parameters and the corresponding values are listed in the [JAVA_PROPERTIES] section of the generated properties file *<PASW Collaboration and Deployment Services installation directory>/setup/resources/netweaver/config/environment.properties*.

```
...
[JAVA_PROPERTIES]
com.spss.psapi.session.serverInstallationDirectory=E:/PASW_CADS/components/modeler
com.spss.psapi.extensions.autoloadDirectory=E:/PASW_CADS/components/modeler/ext/lib
csp.installationDirectory=E:/PASW_CADS/components/modeler
...
```

- Keystore path and password used during the repository installation must also be added as custom parameters.

- **platform.keystore.file** The path of PASW Collaboration and Deployment Services keystore file.
- **platform.keystore.password** Keystore password.

▶ Java Preferences Factory custom parameter must also be added.

java.util.prefs.PreferencesFactory The parameter value must be set to java.util.prefs.WindowsPreferencesFactory.

▶ The SAP NetWeaver instance must be restarted for the parameters to take effect.

Appendix

B

Troubleshooting

Certain error messages and symptoms are common when installing and working with PASW Collaboration and Deployment Services. Methods for clearing these errors and establishing a functional system exist for:

- **PASW Collaboration and Deployment Services**. Common problems when installing and starting the application on supported server platforms.
- **DB2 for IBM i**. Symptoms and error messages that surface while transacting with a DB2 database running on IBM i.
- **Oracle 10g and 11g**. Symptoms and error messages that surface while transacting with an Oracle 10g and 11g databases.
- **JBoss**. JBoss application server running PASW Collaboration and Deployment Services.
- **Oracle WebLogic**. WebLogic application server running PASW Collaboration and Deployment Services.
- **WebSphere**. WebSphere application server running PASW Collaboration and Deployment Services.

It is always a good practice to refer to PASW Collaboration and Deployment Services log files to establish the cause of the problem. For more information, see the topic Logging Services in Chapter 12 on p. 108.

PASW Collaboration and Deployment Services

How do I prevent performance bottlenecks and CPU usage issues when starting and deploying PASW Collaboration and Deployment Services?

Depending on the specific system configuration, previously installed antivirus or spyware software may be configured for “deep scanning” of application components. These third party applications can be reconfigured to scan during certain times, or they can be turned off during installation and manually restarted.

Additionally, some of the more strict server-side firewall settings may negatively impact startup performance and not allow access.

If you are experiencing significant system degradation when starting the service, disable any nonessential processes and restart the PASW Collaboration and Deployment Services.

Once I log in to the administrative interface, how do I determine which database I am accessing?

Database connection information can be downloaded and accessed from the Web interface.

1. After authenticating, click About from the navigation list options. The About page appears.
2. Click the Download version and system details link at the bottom of the page. When prompted, save the file to disk.
3. Open the file in a text editor and search for *Database Details*. This section contains detailed information on the database being used, including name, version, and a table listing.

The application throws java.lang.OutOfMemoryError: PermGen space exception.

This error occurs when the JVM runs out of space in the permanent generation heap due to a large number of used classes. The solution is to increase the value specified with PermSize JVM parameter. For example, for JBoss installations, the size of permanent generation heap available to the wrapper service can be increased by modifying the following line in *<JBoss Installation Directory>/wrapper/conf/wrapper.conf*:

```
wrapper.java.additional.1=-Dprogram.name=run.bat -XX:PermSize=128m.
```

For information about increasing the permanent generation heap size for other application servers, see the application server vendor documentation.

Out of memory errors can also be prevented by adding JVM parameters to tune memory allocation and garbage collection, for example:

```
-XX:+CMSPermGenSweepingEnabled -XX:+CMSClassUnloadingEnabled
```

When a BIRT report is run in Deployment Portal, the application is not able to authenticate my credential for accessing the data source of the report and is repeatedly displaying the login screen.

- Verify that the data source for the report and the credentials are defined correctly. For more information, see the corresponding section of the *Deployment Manager User's Guide*.
- If the data source for the report is JDBC-based, verify that the proper driver is installed with repository. For driver path information specific to the operating platform, see the installation instructions.

How do I restore PASW Collaboration and Deployment Services if my keystore file has been lost?

The keystore file contains the keys used to encrypt passwords used by PASW Collaboration and Deployment Services, such as the master password for database access. If the keystore file is lost, the system becomes unusable. If backup of the keystore is available, it can be restored to the original location. If you are unsure what the original path of the keystore was, you can look up the *keystorePath* property of *keystoreSecurity* element in *<PASW Collaboration and Deployment Services Installation Directory>/platform/setupinfo.xml*.

If the keystore file is lost and backup is not available, the system must be reinstalled by re-running the setup utility in *<PASW Collaboration and Deployment Services Installation Directory>/setup* and pointing it to the existing repository database. All passwords that existed in the system, such as the passwords for external directory services, defined credentials, etc. must be manually reentered.

Oracle Database

How do I create a user and tablespace?

To clear and reestablish the *spssplat* user and tablespace from an Oracle database, issue the following set of commands:

```
drop user spssplat cascade; CREATE USER spssplat IDENTIFIED BY spssplat
DEFAULT TABLESPACE SPSSPLAT TEMPORARY TABLESPACE TEMP
QUOTA UNLIMITED ON SPSSPLAT;
@$ORACLE_HOME/sqlplus/admin/pupbld;
GRANT CONNECT, RESOURCE, UNLIMITED TABLESPACE TO spssplat;
```

JBoss

How is the session timeout value configured to adjust the amount of time a user can remain idle?

Once a user is logged in to PASW Collaboration and Deployment Services, a period of inactivity is allowed before the session is terminated and the user must reauthenticate. To increase or decrease this value:

1. From the installation directory, navigate to *\JBoss\server\default\deploy\jbossweb-tomcat50.sar*.
2. Open *web.xml* in a text editor.
3. Locate the section for *Default Session Configuration*, and edit the value for <session-timeout>.
4. Stop and restart the application.

   Note: This file is processed when the application is deployed; configuration changes do not take effect until the server is restarted.

How do I determine the port on which my version of JBoss is running?

The JBoss application server's HTTP port is defined in the file:

jboss-3.2.7\server\default\deploy\jbossweb-tomcat50.sar\server.xml

with the attribute:

/Server/Service/Connector@port

Note: Depending on the release of JBoss, the version numbers in the path may vary.

What additional settings are required for PASW Collaboration and Deployment Services FIPS 140-2 compliance on JBoss?

For PASW Collaboration and Deployment Services to function properly when running on JBoss in FIPS 140-2-compliant mode, {URIEncoding="UTF-8"} attribute must be specified for the HTTPS connector.

Alternatively, from the command line, the netstat command can be used to view applications and the ports that are in use.

WebLogic

"IOException: Resource has been deleted" is thrown in Deployment Portal when trying to access file attachments that contain reporting output.

The exception can occur if the PASW Collaboration and Deployment Services installation is running on WebLogic application server using JRockit rather than Sun JRE. If the exception occurs, reconfigure WebLogic to use Sun JRE. For more information, see WebLogic documentation.

Cascading parameters are not displayed correctly in reports when PASW Collaboration and Deployment Services is run with WebLogic 9.2 and 10 on Solaris 10.

-Djava.awt.headless=true startup argument must be added to the application server Java environment.

WebSphere

Miscellaneous errors occur during package installation (with Package Manager) into the repository using a WebSphere application server.

Make sure the latest vendor patches have been applied to the application server.

Server log is reporting encryption errors, such as exception com.ibm.crypto.provider.AESCipher.engineGetKeySize(Unknown Source)

The error occurs with WebSphere 6.1 Service Pack 19 and is caused by the incorrect password value. To correct the error, copy the value of platform.keystore.password from

<PASW Collaboration and Deployment Services installation directory>/platform/setupinfo.xml

to

<WEBSPHERE_HOME>/profiles/AppSrv01/config/cells/xi-wyueNode01Cell/nodes/xi-wyueNode01/servers/<server name>/server.xml

Upgrading to WebSphere 6.1 Service Pack 23 may also resolve encryption problems.

"CWSIS1535E: The messaging engine's unique id does not match that found in the data store" error

The error can be corrected by stopping PASW Collaboration and Deployment Services and deleting the repository database tables with names beginning with the *SIB* prefix. The tables will be recreated when PASW Collaboration and Deployment Services is restarted. Note that this solution applies only if you do not need to keep any of the currently stored

persistent messages. For more information about WebSphere JMS troubleshooting, see *http://www.redbooks.ibm.com/redpapers/pdfs/redp4076.pdf*.

Remote exception when running a BIRT report against a PASW Statistics data source (with PASW Statistics data file JDBC driver) on a WebSphere cluster

The problem may be resolved by adding Dcom.ibm.ws.classloader.encodeResourceURLs=true to generic JVM arguments using WebSphere administration console for every node of the cluster.

Index