Sterling Selling and Fulfillment Foundation

Installation Guide

Release 9.2.0.16

Sterling Selling and Fulfillment Foundation

Installation Guide

Release 9.2.0.16

Note
Before using this information and the product it supports, read the information in "Notices" on page 179.

Contents

Chapter 1. Getting Started 1
Before You Begin 1
High-Level Installation Summary 1

Chapter 2. Creating a Security Plan . . . 3
About Creating a Security Plan 3
Security-Planning Your Deployment Architecture . . 3
Security Planning-Current Security Infrastructure Analysis 4
Security Planning-Authentication and Authorization 4
Security Planning-Data Encryption 4
Security Planning-Network Topology 4
Security Planning-Java Protocol Security Measures. . 5
Disabling Java Protocols-EJB 6
Disabling Java Protocols-HTTP 6
Disabling Java Protocols-JMS 6
Securing Java Protocols 7
Web Security Planning 8
Web Security Planning-Post installation Recommendations 9
Web Security Planning-Session Security 9
Web Security Planning-Operating System Permissions. 9
Web Security Planning-Documentation 9
Web Security Planning-Routing 9
Web Security Planning-Web Server Executables . . 9
Database Security 10
Database Security-Credit Card Encryption . . . 10
Adding Sterling Selling and Fulfillment Foundation as a Trusted Web site 10

Chapter 3. Installing and Configuring Application Tier Software. 13
Installing Your Application Server 13
Installing and Configuring a Proxy Server 13
About Configuring a Proxy Server for SSL or HTTPS Configuration 13
Configuring a Proxy Server for SSL or HTTPS-Example 14
Setting Up an Image Server for RCP 15
Setting Up Apache Image Server-Example . . . 15

Chapter 4. Before Installing and Configuring a Database 17
Database Sizing 17
Database Capacity Planning 17
Disk Estimation for IBM Sterling Distributed Order Management. 17
Tracking and Estimating Future Disk Requirements. 19

Chapter 5. Installing and Configuring Database Tier Software on UNIX or Linux 21
Database Configuration on UNIX/Linux-Overview 21
Overview of Oracle Database Configuration . . 21
Configuring Oracle with Single or Multiple Byte Characters. 21
Oracle Database User Privileges 22
Manually Creating Views on Oracle 23
Setting Up Scripts When Using Locally Managed Tablespaces or Other Utility to Size Database . . 24
Setting Database Parameters in Oracle 24
Rolling Back or Undoing Changes in Oracle . . 24
Enabling Failover in a Multiple Node Oracle RAC Database Cluster (UNIX/Linux) 25
Configuring the NLS_LANG Parameter for an Oracle Client 25
Configuring DB2 on UNIX or Linux 26
DB2 Parameter Settings 26
DB2 JDBC Drivers 26
Tuning Considerations for DB2 27
DB2 User Privileges 28
Manually Creating Objects on DB2-Setting Up Scripts 28
Manually Creating Objects on DB2-Running Scripts 28
Manually Creating Views on DB2 29
Enabling the Text Search Feature on DB2 . . . 29

Chapter 6. Installing and Configuring Database Tier Software on Windows . . 31
Database Configuration on Windows-Overview . . 31
Configuring a Database Server on Windows . . . 31
Configuring a Database for a Multiple Schema Environment 31
Generating Data Backup and Restore Scripts . . . 31
Configuring an Oracle Database on Windows . . . 32
Setting Database Parameters in Oracle 33
Rolling Back or Undoing Changes in Oracle . . 33
Configuring Oracle for Single or Multiple Byte Characters. 33
Oracle Database User Privileges 34
Manually Creating Views on Oracle 35
Enabling Failover in a Multiple Node Oracle RAC Database Cluster in Windows 35

Chapter 7. Installing Sterling Selling and Fulfillment Foundation in a Windows Environment 37
High Level Overview of Installation Process . . . 37
Installation Worksheet for Windows 38
Memory Parameter Settings 39
Installation Directory Information for Windows 40
Installation Methods on Windows 40

Install the IBM Installation Manager 40
Add a Repository for Sterling Selling and Fulfillment Foundation in IBM Installation Manager . 41
About Recording Silent Installation Files for Windows 41
About Applying Database Definition Language (DDL) Statements After a Windows Installation . . 41
Running an Installation on Windows 42
Running a GUI-Based Installation on Windows 42
Running the Text-Based Interactive Installation Program on Windows 44
Running a Silent Installation on Windows . . . 47
Post Installation Tasks for Windows 53
Starting and Stopping the Information Center . . 53
Web Security Planning-Post installation Recommendations 54
Upgrade Information for Windows 54
Upgrade Installation Considerations for Windows 54
Running a Silent Upgrade on Windows 55
Post Upgrade Tasks for Windows 55

Chapter 8. Installing Sterling Selling and Fulfillment Foundation in UNIX and Linux Environments 57
High Level Overview of Installation Process on UNIX/Linux 57
Installation Worksheet for UNIX and Linux. 58
Memory Parameter Settings 59
Installation Directory Information for UNIX and Linux 59
Installation Methods on UNIX/Linux. 60
Setting Up X Windows for a GUI Installation . . 60
About the JDK for UNIX/Linux 60
Install the IBM Installation Manager 61
Add a Repository for Sterling Selling and Fulfillment Foundation in IBM Installation Manager . 61
About Recording Silent Installation Files in UNIX or Linux 62
Environment Specific Notes for UNIX and Linux . . 62
Before AIX Installations Only 63
Before Linux Installations Only 63
Before RedHat Enterprise Linux Installations Only. 63
Running an Installation in a UNIX or Linux Environment 64
Running a GUI-Based Installation on UNIX or Linux 64
Running the Text-Based Interactive Installation Program in UNIX or Linux 66
Running a Silent Installation in UNIX or Linux 68
Post Installation Tasks for UNIX/Linux 75
Starting and Stopping the Information Center . . 75
Web Security Planning-Post installation Recommendations 76
Upgrade Information for UNIX/Linux 76
Running a Silent Upgrade in UNIX or Linux . . 76
Post Upgrade Tasks for UNIX/Linux 77

Chapter 9. Installing the IBM Sterling Sensitive Data Capture Server 79
Installing the IBM Sterling Sensitive Data Capture Server 79

Chapter 10. Installing the Sterling Selling and Fulfillment Foundation Language Pack 81
Installing the Sterling Selling and Fulfillment Foundation Language Pack 81
Loading the Sterling Selling and Fulfillment Foundation Language Pack Factory Defaults-Overview 81
Loading the Sterling Selling and Fulfillment Foundation Language Pack Factory Defaults for Windows 81
Loading the Sterling Selling and Fulfillment Foundation Language Pack Factory Defaults for UNIX/Linux 82
Loading the Language Pack Translations in Windows 82
Loading the Language Pack Translations in UNIX/Linux 83
Switching the Sterling Selling and Fulfillment Foundation Base Language 84
Creating and Deploying the Enterprise Archive . . 84
Configuring Locales in UNIX/Linux 84
Configuring Locales in Windows 85
About Localized Content in the Information Center 85

Chapter 11. Using a Local Version of the Information Center 87
Updating a Local Version of the Information Center 88
Changing the Port Number Used for Context-Sensitive Help Topics 88

Chapter 12. Installing a Print Server . . 91
Installing a Print Server - Overview 91
Installing Loftware Components 91
Defining Printers on Loftware 92
Defining Printers for the Sterling Store Inventory Management Installation 92
Installing JasperReports 92

Chapter 13. Configuring Utilities. . . . 93
Configuring Sterling Selling and Fulfillment Foundation Utilities - Overview 93
Installation Utilities. 93
Creating Database Schemas and Loading Factory Defaults After Installation 94
Development Utilities - Overview 102
About the Configuration Deployment Tool . . 102
Truncating Transaction Data 103
Run-Time Utilities. 103
Setting Up the Runtime Utilities 105

Chapter 14. Deploying Sterling Selling and Fulfillment Foundation 109

Sterling Selling and Fulfillment Foundation Deployment - Overview 109
Configuring an Oracle WebLogic Application Server - Overview 109
Setting Up the WebLogic Script File 110
Configuring Oracle WebLogic XML Registry . . 111
Disabling Instrumented Stack Traces in WebLogic 111
Setting Up WebLogic to Display Barcodes and Graphs 112
Setting Up WebLogic to Use HTTP In-Memory Session Replication 112
Building the Enterprise Archive (EAR) Package on WebLogic 113
Building Web Services on WebLogic 113
EJB Web Services Overview 114
JAX-WS Web Services Overview 115
Including Custom Classes on WebLogic . . . 121
Creating the EAR on WebLogic 122
Installing and Deploying Sterling Selling and Fulfillment Foundation on Different Servers . . 123
Precompiling the WAR File on WebLogic . . . 123
Deploying the Enterprise Archive (EAR) on WebLogic 123
Verifying Installation on WebLogic 124
Configuring a WebSphere Application Server - Overview 125
Configuring WebSphere to Avoid Direct Datasource Lookup Warning Messages at Runtime 125
Configuring WebSphere to Avoid FileNotFound Exceptions 125
Configuring the WebSphere Classloader . . . 125
Setting Up Application Clients to Invoke Sterling Selling and Fulfillment Foundation EJBs 126
Configuring WebSphere JVM Settings 126
Setting Up WebSphere to Display Barcodes and Graphs 127
Building the Enterprise Archive (EAR) Package on WebSphere 128
Building Web Services on WebSphere 128
EJB Web Services Overview 129
Defining an EJB Web Service with WebSphere 129
About EJB Web Services and Security 130
JAX-WS Web Services Overview 130
About JAX-WS Web Services Security 131
Defining a JAX-WS Web Service with WebSphere 131
Example of webservicesbeans.xml file - WebSphere 134
Exposing an SDF Service as a Web Service on WebSphere 134
Running the JAX-WS Client Generator on WebSphere 135
Sample Code for BeanService Classes - WebSphere 136
Including Custom Classes on WebSphere . . . 137
Creating the EAR File on WebSphere 137
Precompiling the WAR File on WebSphere. . . 138
Deploying the Enterprise Archive (EAR) Using the WebSphere Admin Console 139
Verifying the Installation on WebSphere . . . 140
Configuring a JBoss Application Server 141
Setting Up JBoss to Display Barcodes and Graphs 142
Building the Enterprise Archive (EAR) Package on JBoss 142
Building Web Services on JBoss 143
EJB Web Services Overview 143
JAX-WS Web Services Overview 145
Example of webservicesbeans.xml file - JBoss 149
Exposing an SDF Service as a Web Service on JBoss 149
Running the JAX-WS Client Generator on JBoss 150
Sample Code for BeanService Classes - JBoss 151
Including Custom Classes on JBoss 152
Creating the EAR on JBoss 152
Precompiling the WAR File on JBoss 153
Deploying the Enterprise Archive (EAR) on JBoss 153
Configuring DataSource Connection Pooling on WebLogic, WebSphere, and JBoss 155
Configuring Data Source Connection Pooling for Single Schema Installations 155
Configuring Data Source Connection Pooling for Multischema Installations 156
Configuring a Restrictive Cookie Path - Overview 157
Configuring a Restrictive Cookie Path on WebLogic. 157
Configuring a Restrictive Cookie Path on WebSphere 157
Configuring a Restrictive Cookie Path on JBoss 158
Setting the Client Character Display 158
Clearing Browser Caches 158
Clearing the Java Plugin Cache 158
Statistics Monitoring 159
Setting an Enterprise for Logging In to IBM Sterling Business Center 159

Chapter 15. Deploying and Updating the Rich Client Platform Applications . 161

Before You Begin RCP Deployment 161
Deploying RCP - Creating the RCP_EXTN_FOLDER Folder 162
Deploying RCP - Configuring Locations . . . 163
Deploying RCP - Localizing Bundle and Theme Files 165
Deploying RCP - Enabling HTTPS 165
Deploying RCP - Applying Updates 165
Deploying RCP - Running the Ant Script . . . 167
Deploying RCP - Deploying Clients through a Remote Terminal 168
Deploying RCP - Configuring Location Settings 168
Deploying RCP - Connection Settings 170
Deploying RCP - Connection Settings for Fetching Images from the Server 171
Deploying RCP - Configuring Connection Settings for HTTPS Connection 172
Deploying RCP - Security Certificates 173
Deploying RCP - Compression in the Rich Client Platform 173

Deploying RCP - Creating Server-Side Commands
Without Running the Application. 174

Index 175

Notices 179

Chapter 1. Getting Started

Before You Begin

Before you begin installing IBM® Sterling Selling and Fulfillment Foundation, read this guide thoroughly. Then define your processes for handling the following:

- Development and Test Environments
- Security Strategy
- Change Management Strategy
- Development and Test Procedures
- Rollback Strategy
- Upgrades and Maintenance Strategy

In addition, before starting the installation process, read the *Sterling Selling and Fulfillment Foundation: Performance Management Guide* which contains information that helps you optimize the performance of your Sterling Selling and Fulfillment Foundation.

High-Level Installation Summary

This summary gives a high-level description of the entire installation process for Sterling Selling and Fulfillment Foundation, which provides a world-class order management and fulfillment solution across extended enterprises.

Because Sterling Selling and Fulfillment Foundation includes several component solutions, installation is a tiered process. Follow the sequence of steps outlined here to install, configure, and deploy Sterling Selling and Fulfillment Foundation

During installation and configuration, you can also refer to the *Sterling Selling and Fulfillment Foundation: Performance Management Guide*, which offers configuration and tuning information.

Installation Summary

1. Ensure that you have the necessary system requirements to install and run Sterling Selling and Fulfillment Foundation. For information about system requirements, go to the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning.
2. Set up your security infrastructure.
3. Install and configure your application server.
4. Install and configure your WebServer or Proxy Server.
5. Size your database on Windows or UNIX.
6. Install and configure your database software on Windows or UNIX/Linux.
7. Install the Sterling Selling and Fulfillment Foundation application on Windows or UNIX/Linux.
8. Install and configure the Sterling Sensitive Data Capture Server (recommended).
9. Install the Sterling Selling and Fulfillment Foundation language pack (optional).
10. Optionally install the print server.

11. Configure the Sterling Selling and Fulfillment Foundation properties to use with the database, agent servers, LDAP servers, logging, and so forth. See the *Sterling Selling and Fulfillment Foundation: Properties Guide* on the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning for more information.
12. Configure the Sterling Selling and Fulfillment Foundation utilities for installation, runtime, migration, and production.
13. Set up the application server for use with Sterling Selling and Fulfillment Foundation on WebLogic, WebSphere®, or JBoss.
14. Build your Enterprise Archive (EAR) on WebLogic, WebSphere, or JBoss.
15. Deploy the EAR to your application server as appropriate on WebLogic, WebSphere, or JBoss.
16. Optionally deploy and update the Sterling Rich Client applications.
17. Optionally run the configuration deployment tool to migrate your configuration data. For information about this tool, see the *Sterling Selling and Fulfillment Foundation: Configuration Deployment Tool Guide* on the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning.
18. Optionally log into the IBM Sterling Business Center application, which is part of the Sterling Selling and Fulfillment Foundation installation. For documentation about IBM Sterling Business Center and Sterling Selling and Fulfillment Foundation, go to the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning.

Chapter 2. Creating a Security Plan

About Creating a Security Plan

This chapter provides security recommendations and guidelines for running Sterling Selling and Fulfillment Foundation. It is intended to help you create a reasonably secure implementation of the application.

Because we recognize that you may have unique business or operational requirements, Sterling Selling and Fulfillment Foundation does not provide a specific set of instructions that you can follow to completion for creating a security plan. Typically, it is not possible to configure a system solely for security at the detriment of other engineering or business realities.

Given the complicated nature of security, it is recommended that you refer to the following documents:

- http://www.nsa.gov/snac for tips on how to harden your operating system, database, and network
- http://download.oracle.com/docs/cd/E12840_01/wls/docs103/security/index.html for tips on how to secure Oracle WebLogic 10.3
- The Rhino9 Team, *The Modern Hackers Desk Reference*; available at http://www.f4.ca/text/mhdr.html.
- Tom Bialaski and Michael Haines, *Solaris and LDAP Naming Services, Deploying LDAP in the Enterprise*; Prentice Hall PTR, 2001.

Security-Planning Your Deployment Architecture

Prior to procuring and implementing the hardware and software that make up Sterling Selling and Fulfillment Foundation, you need to plan your deployment architecture by completing the following tasks:

- Conduct an analysis of the current security infrastructure in your organization. For more information on identifying the correct security infrastructure in Sterling Selling and Fulfillment Foundation, see “Security Planning-Current Security Infrastructure Analysis” on page 4.
- Conduct an analysis of authentication and authorization mechanisms in your organization to identify the steps needed to incorporate them into Sterling Selling and Fulfillment Foundation. For more information on the mechanism used for authentication in Sterling Selling and Fulfillment Foundation, see “Security Planning-Authentication and Authorization” on page 4.
- Conduct an analysis of your data encryption mechanisms for deploying Sterling Selling and Fulfillment Foundation over the internet. For more information on the different variations of the data encryption mechanisms, refer to “Security Planning-Data Encryption” on page 4.
- Conduct an analysis of your organization's network topology required to deploy Sterling Selling and Fulfillment Foundation. For more information on the various methods to deploy the application, refer to “Security Planning-Network Topology” on page 4.

Completing these tasks enables you to:

- Estimate your server requirements.

- List the major security software and hardware needed to implement Sterling Selling and Fulfillment Foundation.

Security Planning-Current Security Infrastructure Analysis

In order to ensure that your Sterling Selling and Fulfillment Foundation is a secure Web application, there are many factors involved. Be sure to answer the following questions before you start your Sterling Selling and Fulfillment Foundation implementation.

- Does your organization have security personnel? If not, you may wish to seek input from an Internet security company in your area.
- Do you own a network scanner such as Internet Security Systems System Scanner or Internet Scanner? Products like these help you identify common problems with servers that are exposed to the Internet.
- Do you own an intrusion detection system such as Symantec Intruder Alert? This type of product works with your firewall to stop an intrusion before mission-critical data or systems are tampered with.

Security Planning-Authentication and Authorization

Authentication and authorization are vital to security. Due to the constantly changing authentication methodologies, including biometrics, public key infrastructure (PKI), and ever-increasing encryption algorithms, Sterling Selling and Fulfillment Foundation provides documentation on implementing a lightweight directory access protocol (LDAP) or any Java™ Authentication and Authorization Service (JAAS) compliant security module for authentication. With LDAP user and password management can be centralized. For information on deploying Sterling Selling and Fulfillment Foundation and integrating with LDAP, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*. The default authentication mechanism is implemented against the Sterling Selling and Fulfillment Foundation database.

Security Planning-Data Encryption

Due to the differences in the nature of businesses, you may implement different security measures when implementing a web application. How you plan to deploy the application and what security measures are taken are unique to each business. Most security measures come at a cost of performance. The Internet is a public network. Sensitive data should be encrypted while traveling across it. Encrypting information that travels across the Internet has an associated cost. If Sterling Selling and Fulfillment Foundation is not to be deployed on the Internet, encryption may not be necessary and the cost is thereby negated.

The data encryption mechanisms recommended for Sterling Selling and Fulfillment Foundation are:

- SSL - 128-bit encryption is the recommended encryption level.
- VPN - 3DES or AES is the recommended encryption algorithm.

Security Planning-Network Topology

Where is Sterling Selling and Fulfillment Foundation being accessed from?

- Public Internet?
- Virtual private network (VPN)?
- Internal Local area network (LAN)?

Sterling Selling and Fulfillment Foundation is typically implemented as an internal application that is accessible from an Internal Network or across from VPN.

Regardless of which network, we strongly recommend that you use SSL to encrypt all the Sterling Selling and Fulfillment Foundation screen requests. SSL processing can be expensive and can add an additional 30% or more processing overhead to each application server transaction. Depending on your transaction volumes, you may want to offload your SSL processing to specialized devices such as an F5 load-balancer with built-in hardware SSL engines.

Accessing Over the Public Internet

If you are accessing Sterling Selling and Fulfillment Foundation over the Public Internet you have to also consider additional security concerns such as denial of service attacks.

Deploying Over a Virtual Private Network

If you are deploying Sterling Selling and Fulfillment Foundation over a virtual private network (VPN), the major factor in security and performance is the VPN encryption. Many firewall providers offer encryption and decryption accelerators that can be added directly to their firewalls. Checkpoint's FireWall-1, VPN-1 Accelerator Card II, is an example of this. However, one consideration for purchasing accelerator cards is how many VPN tunnels are needed. You also need to determine if the VPN is being set up for site-to-site implementation or if each individual user opens their own tunnel. If you decide on a site-to-site VPN, typically memory in the firewall is the greatest concern. If each user opens their own tunnel, processor speed is the largest concern.

In many cases the deciding factor is the speed at which your VPN is connected. If you have a T1 line, a single processor machine may suit your needs. If you plan to deploy over a T3 line, you may wish to consider a multiple-processor machine. Most firewall and VPN vendors can help you size the machine you purchase from them for optimal security and performance.

Deploying Over a Local Area Network

If you are deploying Sterling Selling and Fulfillment Foundation over a local area network (LAN), performance should not be an issue. We strongly recommend you SSL all Sterling Selling and Fulfillment Foundation screens even on an Internal Network.

Security Planning-Java Protocol Security Measures

As with the usage of any protocol technology there are certain associated risks. The Sterling Selling and Fulfillment Foundation APIs are exposed over various protocols. Therefore, IBM strongly recommends that you disable protocols that you do not use.

Disabling Java Protocols-EJB

About this task

To disable Enterprise JavaBeans (EJB) from Sterling Selling and Fulfillment Foundation, comment out the "session" element in the XML descriptor file, <INSTALL_DIR>/repository/eardata/platform/descriptors/<App_Server>/EJB/META-INF/ejb-jar.xml.

Important: To avoid an error when deploying the ejb-jar.xml for WebLogic, you must comment out the following session bean of the XML file:

```
<session>
   <display-name> The Sterling Selling and
Fulfillment Foundation DOM API Session bean
   </display-name>
   <ejb-name> interop.services.ejb.InteropEJBApi </ejb-name>
   <home> com.yantra.interop.services.ejb.InteropEJBHome </home>
   <remote> com.yantra.interop.services.ejb.InteropEJBApi </remote>
   <ejb-class> com.yantra.interop.services.ejb.InteropEJBImpl </ejb-class>
   <session-type> Stateless </session-type>
   <transaction-type> Bean </transaction-type>
</session>
```

This session bean is deprecated as of Release 7.7.

Disabling Java Protocols-HTTP

About this task

To disable Hypertext Transfer Protocol (HTTP) as the means to enter API information in Sterling Selling and Fulfillment Foundation, the deployment descriptor needs to be modified. The deployment descriptor, web.xml, is defined by the servlet specification from Sun Microsystems. This deployment descriptor can be used to deploy a web application on any J2EE-compliant application server. The deployment descriptors for Sterling Selling and Fulfillment Foundation are stored in the <INSTALL_DIR>/repository/eardata/smcfs<application_name>/descriptors/<App_Server>/WAR/WEB-INF directory. The deployment descriptor for the InteropHttpServlet needs to be removed from the web.xml file to disable the servlet. Remember to remove both the servlet-name and the servlet-mapping entries from this file.

Disabling Java Protocols-JMS

About this task

In order to use the Java Messaging Service (JMS) features of Sterling Selling and Fulfillment Foundation, there must be a JMS server. There must be queues set up both on the JMS Server and within Sterling Selling and Fulfillment Foundation.

To ensure that JMS is not used without authorization there should be appropriate permissions on the JMS server and in Sterling Selling and Fulfillment Foundation. You can limit the ability of users to enable JMS by disabling permissions using Process Modeling in the Applications Manager. For more information about enabling and disabling permissions, see the *Sterling Selling and Fulfillment Foundation: Configuration Guide*.

Securing Java Protocols

About this task

Protocols are specified in the `yifclient.properties` file as LOCAL. To specify a different protocol, use the `<INSTALL_DIR>/properties/customer_overrides.properties` file to override the `yif.apifactory.protocol=<protocol_type>` property. Other valid values for <protocol_type> are HTTP, HTTPS, and EJB. For additional information about overriding properties using the `customer_overrides.properties` file, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.

Note: If you use an EJB protocol, you must also add the following property entries to the `<INSTALL_DIR>/properties/customer_overrides.properties` file based on your application server:

For WebLogic:

```
yif.java.naming.factory.initial=weblogic.jndi.WLInitialContextFactory
yif.java.naming.provider.url=t3://<ipaddress>:<port>
```

For WebSphere:

```
yif.java.naming.factory.initial=com.ibm.websphere.naming.WsnInitialContextFactory
yif.java.naming.provider.url=iiop://<ipAddress>:<port>
```

For additional information about overriding properties using the `customer_overrides.properties` file, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.

Securing Java Protocols-EJB

When the Sterling Selling and Fulfillment Foundation APIs are deployed through EJB, they use a Java Naming and Directory Interface (JNDI) lookup for a context to call the EJB Objects. JNDI looks up a context that is a handle to the EJB Object or API. The APIs do not have authentication or authorization. However, security principal and credentials can be supplied by specifying them in the `yifclient.properties` configuration file. The server can be set up to validate the passed security credentials.

The Sterling Selling and Fulfillment Foundation HTTP/HTTPS Interface uses JavaServer Pages (JSPs) installed on the application server and does not need access to JNDI. There are two ways to protect the Sterling Selling and Fulfillment Foundation APIs over EJB:

- WebLogic allows JNDI and remote method invocation (RMI) to be tunneled over HTTP. In your architecture there should be a proxy to inspect all the requests for Sterling Selling and Fulfillment Foundation. This ensures that all the requests are for HTML, and not tunneled RMI or JNDI over HTTP.
- If Sterling Selling and Fulfillment Foundation is deployed on WebLogic, a security realm should be set up to protect JNDI resources. This does not affect any screens that are packaged with Sterling Selling and Fulfillment Foundation or any screens that extend Sterling Selling and Fulfillment Foundation.

If the application is deployed on WebSphere or JBoss, you must set up permissions for EJB method. This does not affect any standard screens that are packaged with Sterling Selling and Fulfillment Foundation or the custom screens you create.

Important: If you attempt to run Sterling Selling and Fulfillment Foundation using HTTPS, the Applications Manager does not open.

If a custom user interface is being built using the Sterling Selling and Fulfillment Foundation APIs through EJB and not by extending the Sterling Selling and Fulfillment Foundation Presentation Framework, you cannot use the client wrapper supplied with Sterling Selling and Fulfillment Foundation because it currently is incapable of passing credentials. This also applies to any use of the `YIFAPIFactory` class.

Securing Java Protocols-HTTP API Tester

The HTTP API tester is provided *only* to test APIs in development mode. If you plan to provide access to this page in production, you should secure access to it.

You can use the HTTP API tester to test the upload and download of binary large objects (BLOBs). To upload a BLOB, user information (user ID and password) should already be present in the session. If a session is not already open, you can make a dummy API call so that user information gets stored in the session. You do not need to make a dummy API call to download a BLOB.

To secure access to the Sterling Selling and Fulfillment Foundation `httpapitester`, the deployment descriptor needs to be modified. The deployment descriptor's `web.xml` is defined by the servlet specification from Sun Microsystems. This deployment descriptor can be used to deploy a web application on any J2EE-compliant application server. The deployment descriptor for Sterling Selling and Fulfillment Foundation are stored in the `<INSTALL_DIR>/repository/eardata/smcfs<application_name>/descriptors/<App_Server>/WAR/WEB-INF` directory. By using the security-constraint element with the web-resource-collection element, you can set up authorization to protect this page from unauthorized access. For more information about the `web.xml` deployment descriptor, see the documentation for your application server.

Note: After buildear.sh is run, a `web.xml.sample` file is generated in the `<INSTALL_DIR>/repository/eardata/smcfs/extn` folder. To perform any changes to the `web.xml` file, copy the `web.xml.sample` file to the same folder (`<INSTALL_DIR>/repository/eardata/smcfs/extn`) and rename it to `web.xml`. Now perform changes to the `web.xml` file in the `<INSTALL_DIR>/repository/smcfs/extn` folder.

Alternatively, you can simply remove the `yfshttpapi` directory under `<INSTALL_DIR>/repository/eardata/platform/war` and secure the `/interop/InteropHttpServlet` servlet using the security features provided by your application server.

Specify the following URL to access the HTTP API tester: `http://<ipaddress>:<port>/smcfs/yfshttpapi/yantrahttpapitester.jsp`

Securing Java Protocols-COM+

The extended Component Object Model (COM+) specification covers security in great detail. Any COM+ object deployed on a server complies with this standard. For information on setting up security for COM+ objects, see *The Microsoft Developers Network* article available at: http://msdn.microsoft.com/en-us/library/ms681314.aspx

Web Security Planning

IBM highly recommends that a security audit is made prior to deployment.

IBM also recommends that you write log files to several servers. There are several applications that do this with no specific need for Sterling Selling and Fulfillment Foundation to duplicate their efforts. Additionally, products like Symantec's Intruder Alert monitor log files for authentication failures and alert an administrator if a threshold is exceeded.

Web Security Planning-Post installation Recommendations

About this task

After the installation of Sterling Selling and Fulfillment Foundation, be sure to complete the following for ensured security:

Procedure

1. Change the password of the default user (admin).
2. Change permissions on *INSTALL_DIR*/bin/migrator.* files to non-executable.
3. API security is enabled during installation. After installation, you may want to reset the api.security.mode property and carefully consider your API security configuration. For more information about API security modes, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.
4. Ensure that the required permissions to access the API resources are defined for an application. For more information about configuring API security, see the *Sterling Selling and Fulfillment Foundation: Configuration Guide*.

Web Security Planning-Session Security

Session security is handled by the application server, and is stored in a non-persistent cookie on the client. You should ensure that all transactions with the application server are protected with SSL to prevent session hijacking attacks.

Web Security Planning-Operating System Permissions

The following files contain confidential information, such as user name and password combinations stored in clear text. These files should be secured through operating system permissions:

- `sandbox.cfg`
- `<appserver>.log`
- `jdbc.properties.in`
- `yfs.properties.in`
- `yifclient.properties.in`

Web Security Planning-Documentation

All the documentation files for Sterling Selling and Fulfillment Foundation and third-party software should be removed from any production servers.

Web Security Planning-Routing

Routing should not be enabled on a production web server.

Web Security Planning-Web Server Executables

Web servers should *not* be run as root. This ensures that if someone compromises any software associated with the deployment through a bug, they don't have root privileges to damage the server. Web servers allow you to access files on their host machines and as root any of those files can be modified for a deeper attack or deleted to make your web servers unavailable.

It is acceptable, although not recommended, to start the web server as `root`. A proxy server can be used to accept HTTP traffic and redirect it to a port above 1024 on a UNIX system. If a proxy is not available and the web server must be started on port 80 it is necessary to start the web server as `root`. The web server then calls `setuid` to transfer root privileges to a generic unprivileged account. The web server's configuration file should allow you to specify what user it runs as. Any user may own the binary. The `setuid` bit should not be set on the web server binary.

Database Security

Set up separate accounts on the database server for installing the Sterling Selling and Fulfillment Foundation schema and for accessing the application database.

If using an Oracle database on the production database server, the Oracle parameter `DBLINK_ENCRYPT_LOGIN` in your `init.ora` file should be set to `TRUE`. This ensures that all connections to the database are not sent as clear text.

Database Security-Credit Card Encryption

IBM provides an application, the IBM Sterling Sensitive Data Capture Server, that captures and tokenizes credit card numbers and store value card numbers. IBM recommends that you review the *Sterling Selling and Fulfillment Foundation: Secure Deployment Guide* for the IBM approach to meeting PCI DSS and PA-DSS requirements.

If you want to ensure that credit card numbers are encrypted at the database level, you configure that functionality when setting Hub attributes in the Applications Manager. When setting Hub attributes, make sure that the credit card number encrypting option is checked. For more information and specific instructions for setting up security, see the *Sterling Selling and Fulfillment Foundation: Configuration Guide*.

Sterling Selling and Fulfillment Foundation also supplies APIs and user exits to encrypt credit card and other secure information. For more information about these APIs, user exits, and other data encryption, see the *Sterling Selling and Fulfillment Foundation: Extending Transactions* and the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Adding Sterling Selling and Fulfillment Foundation as a Trusted Web site

About this task

You should set Sterling Selling and Fulfillment Foundation to be recognized as a trusted Web site. Not doing so could cause certain pop-up windows such as date and time selection to display a status bar, thereby hiding certain action buttons.

To add Sterling Selling and Fulfillment Foundation to the list of trusted Web sites:

Procedure

1. In the Internet Explorer menu bar, select Tools > Internet Options. The Internet Options pop-up window is displayed.
2. In the Internet Options pop-up window, select the Security tab.
3. Click the Trusted Sites icon.

4. Click the Sites action button. The Trusted Sites pop-up window is displayed.
5. In the ‘Add this Web site to the zone' text box, enter the server address where the Application Console is installed. The port number does not need to be specified.
6. Clear the ‘Require server verification (HTTPS:) for all sites in this zone' checkbox.
7. Click OK. This takes you back to the Internet Options pop-up window.
8. Click OK.

Chapter 3. Installing and Configuring Application Tier Software

Installing Your Application Server

Before installing your application server, check the requirements at the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning to make sure you have the applicable hardware and software versions installed.

Install your application server according to the vendor instructions on the product CD-ROM disk.

You should install the Java Development Kit (JDK) that is shipped with your application server (unless otherwise specified at the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning. When upgrading the JDK, be sure to set the correct JAVA_HOME environment variable and update the PATH.

Installing and Configuring a Proxy Server

Installing a proxy web server on dedicated hardware provides:

- Additional network security layers.
- Additional processing power for data encryption protocols.
- Additional options for high availability for your application.

You can install a proxy or Web server to avoid any bottlenecks that may occur when systems try to access Sterling Selling and Fulfillment Foundation installed on your application server. Install and configure the Web server version specified by your application server provider.

About Configuring a Proxy Server for SSL or HTTPS Configuration

You can configure a Web server as an SSL proxy and a load balancer. For an example of an Apache and WebLogic, see “Configuring a Proxy Server for SSL or HTTPS-Example” on page 14. For specific instructions about configuring the proxy server for SSL or HTTPS using the Apache HTTP Server and IBM WebSphere, refer to the appropriate IBM documentation. Information about configuring the Apache HTTP server as a proxy server and a load balancer using SSL or HTTPS on JBoss can be found at: http://community.jboss.org/wiki/UsingModproxyWithJBoss

The SSL proxy allows the web server to manage the SSL encryption load and pass clear text back to application servers. It also divides the workload among the available application servers using the "round-robin load balancing" algorithm. This reduces the network traffic between the web server and application server. The web server enables you to use one secure URL to access any number of application servers that run IBM Sterling Selling and Fulfillment Suite applications.

Configuring a Proxy Server for SSL or HTTPS-Example

About this task

The following procedure explains how to configure a proxy server for WebLogic. Refer to the WebLogic example provided here as a general guideline for configuring a Web server. For information about how to configure a proxy server for WebSphere and JBoss, refer to the documentation pertaining to those products.

Before you begin: The Oracle WebLogic 10.3 installation does not include the Apache HTTP server plug-ins. You must download these plug-ins in a separate compressed file from the Oracle download and support sites.

To configure a proxy server for SSL or HTTPS using the Apache HTTP Server and Oracle WebLogic:

Procedure

1. Install and run Sterling Selling and Fulfillment Foundation on the application servers.
2. Copy the appropriate plug-in to the `/etc/apache2/modules` directory.

   For WebLogic 10.3.2, this is:

   `<WL_HOME>/wlserver_10.3/server/plugin/<OS>/<processor type>`

   Here, `<WL_HOME>` refers to the WebLogic installation directory.
   Example:
   `<WL_HOME>/wls103Linux/wlserver_10.3/server/plugin/linux/x86_64`
   - For i686, copy the WLS plug-in.
   - For x86_64, copy the 64-bit plug-in. The 64-bit plug-in must be requested from Oracle Customer Service.
   - Ensure the plugin is executable.
3. To enable the WebLogic plug-in for load-balancing using HTTP or HTTPS, modify the `httpd.conf` file and add the following. To enable an SSL, add `include conf.d/ssl.conf` as instructed by Apache. By default, RHAS3 has `Include conf.d/*`, which includes `ssl.conf`.

   ```
   LoadModule weblogic_module /etc/apache2/modules/<appserver_plugin_file>
   ```

   For an HTTP proxy, outside any VirtualHost, add the following section. The `context_root` value is the context_root for the web application being proxied:

   ```
   <IfModule mod_weblogic.c>
           WebLogicCluster
   <managed_server1_hostname/IP_address>,<managed_server2_hostname/IP_address>
           DynamicServerList OFF
           Debug ON
           IdemPotent OFF
   </IfModule>
   <Location /context_root>
           SetHandler weblogic-handler
   </Location>
   ```
4. Modify the `ssl.conf` file and add the following lines to the <VirtualHost _default_:443> section. The `context_root` value is the context_root for the web application being proxied.

   ```
   <IfModule mod_weblogic.c>
             WebLogicCluster
   <managed_server1_hostname/IP_address>,<managed_server2_hostname/IP_address>
             DynamicServerList OFF
             Debug ON
             IdemPotent OFF
   ```

```
</IfModule>
<Location /context_root>
            SetHandler weblogic-handler
</Location>
```

5. Create security or SSL certificate, if necessary. If you do not have a CA-signed certificate, you can get one from the Certificate Authority companies such as VeriSign. For more information about security or SSL certificates, see "Deploying RCP - Security Certificates" on page 173.
6. Restart Apache, and verify access with any browser.

   Continue and complete steps 7-10 if using an RCP application only.
7. Copy the security certificate to the `<RCP_EXTENSIONS_FOLDER>/truststore` directory.
8. Build the RCP client.
9. Edit the `locations.ycfg` file and modify the protocol, server, and port attributes of the Config element. Ensure that these attributes point to the proxy.
10. Start the client.

Setting Up an Image Server for RCP

About this task

If fetching images for RCP-based PCAs, you must set up an image server. You can set up any server (such as Apache) as your image server.

To set up the image server:

Procedure

1. Install a web server on any system on which you intend to host the images. For example, you can install an Apache web server on a Windows system.
2. Use the default port # 80 (or any available port #) while installing the Image Server & exclude this port from the OS firewall, if required.
3. Store the images in any convenient location under the <IMAGE_SERVER_HOME> directory. For example, you can store the images under the following directory:

   `<IMAGE_SERVER_HOME>/icons/rcp`

   Here, `<IMAGE_SERVER_HOME>` refers to the name of the directory to which the web server that you have installed points.

   For more information about configuring connection settings to fetch images from the server, see "Deploying RCP - Connection Settings for Fetching Images from the Server" on page 171.

Setting Up Apache Image Server-Example

About this task

This example illustrates the procedure given in "Setting Up an Image Server for RCP." If you install Apache as the web server, then to configure it as the image server, do the following:

Procedure

1. Edit the httpd.conf file to define an alias directive. You can find this file under the following directory structure:

<APACHE_HOME>/conf/httpd.conf

Here, <APACHE_HOME> refers to the name of the directory where you have installed Apache.

The following is a sample entry from the httpd.conf file:

```
Alias /icons "<INSTALL_DIR>/COM/images/icons/"
<Directory "<INSTALL_DIR>/COM/images/icons>"
AllowOverride None
Order allow, deny
Allow from all
</Directory>
```

where /icons is the <virtual dir path> that points to the <INSTALL_DIR>/COM/images/icons/ directory.

For more information about how to define alias directives, go to http://httpd.apache.org/docs/2.2/mod/mod_alias.html#alias. This link provides information about alias directives for Apache version 2.2.

2. Add a new entry or edit the existing entry for configuring the port. For example, add a new entry: Listen 80 in the httpd.conf file. This sets up the server to listen to port number 80 (default setting).
3. Restart the web server.

   When you apply the above configuration, the URL http://<IMAGE_SERVER_HOST_NAME>:<port>/icons points to the local directory <INSTALL_DIR>/COM/images/icons/ and the contents in the local directory are served by the web server.

Results

Finally, test to ensure that the images are accessible through the browser; for example, http://<IMAGE_SERVER_HOST_NAME>:<port>/<virtual dir path>/rcp/<IMAGE_FILE_NAME> from any system. If the images are not displayed, the image server is not configured properly.

Chapter 4. Before Installing and Configuring a Database

Database Sizing

Database sizing helps you estimate the requirements for your database, anticipate capacity for growth, and plan disk requirements. Capacity planning and the steps to estimate the disk size your business requires are described in "Database Capacity Planning" and "Disk Estimation for IBM Sterling Distributed Order Management." "Tracking and Estimating Future Disk Requirements" on page 19 explains the value of tracking your actual database usage.

Database Capacity Planning

"Disk Estimation for IBM Sterling Distributed Order Management" provides a methodology to estimate your initial disk requirements. By using this worksheet and then tracking your actual database usage over time, disk growth will be easier to anticipate.

In recent years, the cost of disks has dramatically decreased and the capacity and speed of disks has increased. Information system managers who order disk capacity have shifted from purchasing disk arrays that are dedicated to a particular database server and project to the use of Storage Area Networks (SANs).

Consider the confidence that you have in your data estimates when making the final purchase decision and adjust accordingly. After your initial purchase and production deployment, you can track disk growth for future purchase forecasts.

Disk Estimation for IBM Sterling Distributed Order Management

About this task

Estimate the disk space you will need for the Sterling Distributed Order Management module of Sterling Selling and Fulfillment Foundation.

The estimation methodology consists of three parts:

1. Estimate the number of orders and order lines you expect to keep in the database.
2. Multiply the number obtained in the previous step by a storage usage factor.
3. Add a minimum base amount.

Keep the following information in mind before calculating the estimated disk space:

- Gather information about the amount of time required to maintain the database, such as:
  - How long do you plan to keep data in the main transactional database before orders are purged to the history database?
  - How long are orders kept in the history database before they are purged?
  - Are you purchasing the storage for the first few years into the implementation?

  Use the following examples to help answer these questions:

– **Case 1** - You need to purchase storage for the first 3 years of the implementation, and your company's data retention policy says that you have to keep data online in the main transactional database for 1 year and in the history database for another 5 years. Orders that are older than 6 years are purged from the system.

  The following solution lets you achieve this goal:

  If you need to purchase storage to cover the first 3 years of implementation, that storage has to be sufficient for 3 years worth of data. At the end of year 3, your database has the data for the third year in the main transactional database while the data for the first and second years is in the history. In this example, you should enter the number 3 as the number of years worth of orders that you expect to keep in the database.

– **Case 2** - Sterling Selling and Fulfillment Foundation has been in production for 10 years and your company's data retention policy says that you have to keep data online in the main transactional database for 1 year and in the history database for another 5 years. Orders that are older than 6 years are purged from the system. Given the same data retention policy as above, how much storage is required?

  At the end of the tenth year, the database has the data for the tenth year in the main transactional database and the data for the fifth, sixth, seventh, eighth and ninth years in the history. Therefore, the database has six years (as dictated by the data retention policy) in the database. In this example, you should enter the number 6 as the number of years worth of orders that you expect to be kept in the database.

- The order in Table 1 includes sales, transfer, return, and work orders.
- This storage estimate is for work-in-progress tables that are used as part of order processing. When the orders are processed, the records in these tables can be purged from the system. These tables include the `YFS_IMPORT`, `YFS_EXPORT`, and so forth. You are strongly urged to aggressively purge data from these tables.
- When procuring your storage, ensure that the storage device has at least the amount of usable space specified in Step 8 of Table 1. This table provides an idea of the usable space for the storage device in your company. However, the actual amount you might require is a factor of Redundant Array of Inexpensive Disks (RAID) set up. This disk subsystem is composed of more than one disk drive to provide improved reliability, response time, and storage capacity.

Use the following worksheet to help you estimate required disk space.

Table 1. Steps for Disk Space Estimation for the Sterling Distributed Order Management Module

Step	Description	Disk Space
1.	Enter the number of years' worth of information to be kept in the system (retention time).	_____
2.	Enter the number of orders you expect to be in the system during the time period specified in Step 1.	_____
3.	Enter the number of order lines present in a typical order.	_____
4.	Enter the number of order lines that are to be stored in the database (multiply the values provided in Step 2 and Step 3).	_____

Table 1. Steps for Disk Space Estimation for the Sterling Distributed Order Management Module (continued)

Step	Description	Disk Space
5.	Enter the order line multiplier: Choose one of the following storage factors that most closely approximates a description of your Sterling Selling and Fulfillment Foundation system: (a) 30 KB - This is primarily used for order management with very little customization. (b) 35 KB - This is primarily used for order management with moderate amount of customization.	_____
6.	Multiply the expected number of order lines from Step 4 and the storage factor from Step 5.	_____
7.	The minimum base storage requirement.	150 MB
8.	The minimum operational storage requirements for Sterling Selling and Fulfillment Foundation.	500 MB
9.	Enter the total estimated storage obtained by adding the values from Step 6, Step 7, and Step 8.	_____

Tracking and Estimating Future Disk Requirements

You should track your actual database storage usage and the number of database records regularly. Correlating these two metrics enables you to plan your future disk requirements. Moreover, determining the average amount of space used for each order or shipment line enables you to accurately predict your future growth requirements.

Chapter 5. Installing and Configuring Database Tier Software on UNIX or Linux

Database Configuration on UNIX/Linux-Overview

This chapter describes how to install and configure the database tier software to run Sterling Selling and Fulfillment Foundation in a UNIX or Linux environment.

Before installing your database server, verify that you have the applicable software versions. For more information, go to the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning.

If you are planning a multischema installation, you must deploy the same database vendor and version across all deployments.

Overview of Oracle Database Configuration

About this task

You need to configure your Oracle database for running in a production environment with Sterling Selling and Fulfillment Foundation. To configure an Oracle database for a production environment, you must:

- Size the database by estimating the required disk space.
- Set the database connection properties.

To create the Oracle database to handle multiple byte characters

Procedure

1. Choose the correct data encoding format for your language. See "Configuring Oracle with Single or Multiple Byte Characters" for more information.
2. Choose the character set suitable for your language. See "Configuring Oracle with Single or Multiple Byte Characters" for specific settings to ensure the database field sizes.

Results

You can use an Oracle database for maintaining information on Sterling Selling and Fulfillment Foundation.

Configuring Oracle with Single or Multiple Byte Characters

About this task

To configure Oracle with single or multiple byte characters:

Procedure

1. After installing Oracle, run the create instance procedure. Use a character set appropriate for your desired language. For example:
   ```
   CHARACTER SET "UTF8"
   ```
2. Configure the INIT<INSTANCE_NAME>.ORA file for Oracle as follows:
   ```
   open_cursors= <set to appropriate value>
   ```

For example, the minimum value for WebLogic equals number of threads (across all application servers) + (connection pool size X prepared statement pool size)

```
cursor_sharing=FORCE
compatible=<set to appropriate value, or remove to default to current release>
timed_statistics=true
db_block_size=8192
optimizer_mode=ALL_ROWS
```

If you are using a multi-byte character set, set the following and restart Oracle:

```
nls_length_semantics=CHAR
```

Alternatively you can run the following prior to running any create table scripts:

```
alter session set nls_length_semantics = CHAR
```

Setting this attribute ensures that the field sizes are not impacted by the number of bytes a data type can store. For example, Varchar(40) would now be able to store 40 Japanese characters instead of 40/3 bytes in the UTF-8 character set.

For the Japanese locale, the AL32UTF-8 character set or the UTF-16 character set must be used.

When you change the multi-byte character set to CHAR by setting nls_length_semantics = CHAR, Oracle reserves space equivalent to 'n' chars, which is more than 'n' bytes. Therefore, when you run the `dbverify.sh` command, the reduced entries in table columns are printed in the `EFrame_Drops.lst` file.

3. Download the Oracle JDBC driver for the version of Oracle you are using from the Oracle Web site and copy it to a well known location for reference during installation.

   The Oracle JDBC driver can be found in the JDBC Driver Downloads section at:

   http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html

   Go to the IBM Support Portal at http://www.ibm.com/support/entry/portal/ Planning for Oracle database supported version information.

Results

If configuring for multibyte, do not modify the Sterling Selling and Fulfillment Foundation DDL.

Oracle Database User Privileges

Unless specifically stated for a given task, the Sterling Selling and Fulfillment Foundation user does not require database administrator (DBA) privileges.

Oracle Administrator Privileges

The following list includes some of the basic privileges that should be granted to the Sterling Selling and Fulfillment Foundation administrative user who creates or modifies the Oracle database:

- Alter any sequence
- Alter session
- Create any sequence
- Create procedure
- Create sequence
- Create session

- Create synonym
- Create table
- Create trigger
- Create type
- Create view
- Delete any table
- Execute any procedure
- Execute any type
- Connect
- Insert any table
- Select any dictionary
- Select any sequence
- Select any table
- Select catalog role
- Update any table

Assign the user responsible for creating and modifying the Oracle database a specified quota (extent) in the tablespace, even if that user was assigned an unlimited tablespace when created. Otherwise, the installer will generate a "ORA-01950: no privileges on tablespace name" error.

Oracle Privileges for Application Users

The following list contains basic privileges that should be granted to users who will only run the application.

The Privileges/Grant is only for the tables owned by the user. The following is an example script to create table update grants:

```
select 'grant update on '||table_name||' to [user];'
from user_tables
```

You can write additional scripts to create similar permissions for the following privileges:

- Alter session
- Create session
- Delete table
- Execute procedure
- Insert table
- Select sequence
- Select table
- Update table
- If you are using text indexes, you must also have privileges for CTXAPP or CTXCAT, depending on the type of text indexes you are using.

Manually Creating Views on Oracle

To configure your Oracle database for your production environment, you must set up and run two scripts, Interop_Views.sql and ImportExport_View.sql, to create the views for your schema.

These script files reside in the install_dir/database/oracle/scripts/CustomDBViews/transaction directory.

If you used the silent install method and set the -nodbverify option to true, you will also need to create the tables, indexes, sequences, and so forth for your schema.

Table, index, and sequence create DDLs are created during installation. These reside in install_dir/repository/scripts directory.

Setting Up Scripts When Using Locally Managed Tablespaces or Other Utility to Size Database

About this task

To configure your Oracle database for your production environment, you must set up and run a series of scripts to create the tables, indexes, sequences, and so forth for your schema.

These script files reside in the `<INSTALL_DIR>/database/oracle/scripts/` directory. The `yfs_master_db_script.sql` script is the master script that calls all view scripts required for creating views. Table, index, and sequence creation DDLs are created during installation. These reside in the `<INSTALL_DIR>/repository/scripts` directory.

You must run these scripts only if you are manually creating the views after installation (`REINIT_DB=no`). In the normal installation mode (`REINIT_DB=yes`), the views will be applied automatically.

Refer to “Enabling the Oracle Database Text Search Feature - Overview” on page 94 for information about text search features on the Oracle database.

Setting Database Parameters in Oracle

For information about required parameter settings in your Oracle database, see the *Sterling Selling and Fulfillment Foundation: Performance Management Guide*.

Rolling Back or Undoing Changes in Oracle

About this task

You can roll back or undo changes in Oracle by using the AUTO UNDO management option. It is recommended that you use this option. This avoids manual monitoring of UNDO segments.

If a server is upgraded from a previous version of Oracle, set the UNDO_MANAGEMENT=AUTO parameter in init<SID>.ora. Your database administrator needs to determine the UNDO_RETENTION setting. Ensure that the file system which has the UNDOTBS1 tablespace has enough space to use the AUTOGROW setting.

Enabling Failover in a Multiple Node Oracle RAC Database Cluster (UNIX/Linux)

About this task

To enable failover in a multiple node Oracle RAC database cluster in UNIX/Linux, do the following:

Procedure

1. Navigate to the <INSTALL_DIR>/properties directory, where you will modify the sandbox.cfg and customer_overrides.properties files.

   Note: You might need to create the customer_overrides.properties file, which is just for customizations and is not automatically created during an installation. For additional information about the customer_overrides.properties file, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.

2. In the sandbox.cfg file, add a new property for ORACLE_JDBC_URL which contains the Oracle RAC connection URL.

   The following example shows the suggested URL form. This example shows how the information is organized, but the property value must be one string of text, starting with ORACLE_JDBC_URL=. Your database administrator (DBA) can modify this URL as needed.

   ```
   jdbc:oracle:thin:@
      (DESCRIPTION=
         (ADDRESS_LIST=
            (FAILOVER=ON)
            (LOAD_BALANCE=ON)
            (ADDRESS=(PROTOCOL=TCP)(HOST=myhost1)(PORT=1521))
            (ADDRESS=(PROTOCOL=TCP)(HOST=myhost2)(PORT=1521))
          )
          (CONNECT_DATA = (SERVER = DEDICATED)(SERVICE_NAME = myservicename))
   )
   ```

3. In the customer_overrides.properties file, add the readTimeout property to all Oracle database pools. These values override the corresponding values in the jdbc.properties file.
   - jdbcService.oraclePool.prop_jdbc.readTimeout=90000
   - jdbcService.oraclePool_local.prop_jdbc.readTimeout=90000
   - jdbcService.oraclePool_NoTrans.prop_jdbc.readTimeout=90000
   - jdbcService.oraclePool_NoTrans.prop_jdbc.readTimeout=90000

     The readTimeout value will require tuning. If the value is too low, long-running queries in the system will be interrupted. If the value is too high, recovery when a RAC node fails will be delayed.

4. Run the setupfiles.sh command from the <INSTALL_DIR>/bin directory.
5. Set the propagation delay on the RAC server to 0.

Configuring the NLS_LANG Parameter for an Oracle Client

About this task

To ensure the compatibility of character sets between the Oracle client and the server, the value of the NLS_LANG parameter that is set in the client must match the value in the server.

The entire set of NLS settings pertaining to the database is provided in the NLS_DATABASE_PARAMETERS table.

Procedure

1. Run the following queries to get the corresponding values:
   - SELECT VALUE as Language FROM NLS_DATABASE_PARAMETERS WHERE PARAMETER='NLS_LANGUAGE';
   - SELECT VALUE as Territory FROM NLS_DATABASE_PARAMETERS WHERE PARAMETER='NLS_TERRITORY';
   - SELECT VALUE as Characterset FROM NLS_DATABASE_PARAMETERS WHERE PARAMETER='NLS_CHARACTERSET';

     The NLS_LANG parameter is set as: <Language>_<Territory>.<Characterset> (for example, set NLS_LANG = AMERICAN_AMERICA.UTF8)
2. To set the value of the NLS_LANG parameter in Windows, verify the HKEY_LOCAL_MACHINE/SOFTWARE/ORACLE/NLS_LANG entry in the registry.
3. To set the value of the NLS_LANG parameter in UNIX, NLS_LANG is set as a local environment variable.

Configuring DB2 on UNIX or Linux

Procedure

1. Create the database. Refer to the DB2® documentation for information about creating the database, including creating a schema repository, login, and tablespace. Be sure to install the correct version and fix pack. Be sure to install the client components and compilers before you install the fix pack.
2. Size the database by estimating the required disk space.
3. Set the database connection properties.

   Note: The installation script creates tables and indexes. Certain tables require a page size of 32K. You should have a tablespace to accommodate such tables. DB2 automatically places tables and indexes in the available tablespaces using its internal logic. You can move the tables to a different tablespace after the installation is complete.
4. Install Client Components, Compilers, and Fix Pack.
5. Set Parameters for DB2.
6. Grant Permissions for DB2.
7. Install JDBC Drivers for DB2.

DB2 Parameter Settings

For information about required parameter settings in your DB2 database, see the *Sterling Selling and Fulfillment Foundation: Performance Management Guide*.

DB2 JDBC Drivers

About this task

For DB2, install the appropriate DB2 JDBC Type 4 driver and any correlating patches. Go to the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning for supported version information.

You can obtain these files from the IBM Web site. After you obtain this JDBC driver, record the absolute path to its location on your system. You must supply this absolute path during installation.

If the JDBC driver provided by your database vendor is distributed among multiple files, you must place all the files that comprise the JDBC driver into one .jar file. Follow these steps to create one `.jar` file:

Procedure

1. Identify all the vendor database jar files for the JDBC driver.
2. Create a temporary working directory (`mkdir wd; cd wd`).
3. Extract the contents of each file used for the JDBC driver using the jar utility into the temporary working directory (`jar xvf <jdbc.jar>` for each supplied jar file).

   Note: Various IBM Sterling Selling and Fulfillment Foundation scripts, such as the one used for loading the factory defaults, specify a `DB_DRIVER`. The `DB_DRIVER` specified must include all of these `JAR` files. The `DB_DRIVER` setting is located in `sandbox.cfg`. To make changes to the `DB_DRIVER` setting, edit and save the file, then run `setupfiles.sh`.
4. Bundle the files in the temporary working directory into one file using the jar utility (jar cvf new.jar *).
5. Record the absolute path to this .jar file.

Results

The type-4 driver does not require a separate Java listener running on the database server. Instead, connect directly to the DB2 port.

Tuning Considerations for DB2

When setting up DB2, the Self-Tuning Memory Manager (STMM) should initially be enabled. As the database grows, these settings may be adjusted to tune the system loads. Refer to the *Sterling Selling and Fulfillment Foundation: Performance Management Guide* and the IBM documentation regarding these settings.

Registry Settings for DB2 on UNIX or Linux

For Sterling Selling and Fulfillment Foundation, set the registry variables as follows (the variables with nothing following the equal sign ensure the default setting):

```
db2set DB2_MMAP_WRITE=OFF
db2set DB2_MMAP_READ=OFF
db2set DB2_PINNED_BP=
db2set DB2MEMMAXFREE=

db2set DB2_ENABLE_BUFPD=
db2set DB2_USE_ALTERNATE_PAGE_CLEANING=ON
db2set DB2_EVALUNCOMMITTED=ON
db2set DB2_SKIPDELETED=ON
db2set DB2_SKIPINSERTED=ON
db2set DB2_SELECTIVITY=YES
db2set DB2LOCK_TO_RB=STATEMENT
db2set DB2_NUM_CKPW_DAEMONS=0
```

The *Sterling Selling and Fulfillment Foundation: Performance Management Guide* section on recommended UDB dbset registry variables describes the effects of these parameters with Sterling Selling and Fulfillment Foundation.

DB2 User Privileges

DB2 Administrator Privileges

The DBADM authority must be granted to the Sterling Selling and Fulfillment Foundation administrative user for performing administrative operations in the DB2 database.

DB2 Privileges for Application Users

The following list contains basic privileges that should be granted to users who will only run the application.

- CONNECT
- DATAACCESS

Manually Creating Objects on DB2-Setting Up Scripts

About this task

To set up the scripts:

Procedure

1. Create tablespaces where the Sterling Selling and Fulfillment Foundation tables and indexes reside.
2. Only complete this step if you are manually creating database tables after installation (instead of having installation create them automatically). Modify the `<INSTALL_DIR>/repository/scripts/EFrame_TableChanges.sql` file to reference your newly created tablespaces.

Results

The DDLs in the Sterling Selling and Fulfillment Foundation scripts create a standard set of indexes. You may need to create additional indexes according to your business practice.

Manually Creating Objects on DB2-Running Scripts

About this task

To run the scripts:

Procedure

1. Log into the DB2 server manager as the database administrator.
2. Create the user that is the designated schema owner.
3. Grant the privileges listed in "DB2 User Privileges" to the newly created user.
4. Log out of the DB2 Server Manager and log back in as the newly created user.
5. Verify the database as described in "Verifying the Database Schema" on page 96.
6. Load the Sterling Selling and Fulfillment Foundation database factory defaults as described in "Loading the Sterling Selling and Fulfillment Foundation Database Factory Defaults After Installation" on page 94.

7. Check for the degree of parallelism, using information from the *Sterling Selling and Fulfillment Foundation: Performance Management Guide*.

Manually Creating Views on DB2

About this task

To configure your DB2 database for your production environment, you must set up and run a series of scripts to create the tables, indexes, sequences, and so forth for your schema.

You must run these scripts only if you are manually creating the views after installation (`REINIT_DB=no`). In the normal installation mode (`REINIT_DB=yes`), the views will be applied automatically.

These script files reside in the `<INSTALL_DIR>/database/db2/scripts/` directory.

This is the list of scripts to be edited using a SQL tool:

- CustomDBViews/transaction/ImportExport_View.sql
- CustomDBViews/transaction/Interop_Views.sql
- CustomDBViews/transaction/InvSnapshot_vw.sql
- CustomDBViews/transaction/yfs_cross_reference_vw.sql
- CustomDBViews/transaction/yfs_iba_ord_demand_vw.sql
- CustomDBViews/transaction/yfs_iba_resv_demand_vw.sql
- CustomDBViews/transaction/yfs_invtdmddtl_vw.sql
- CustomDBViews/transaction/yfs_noPendMove_nodeInventoryDtl_vw.sql
- CustomDBViews/transaction/yfs_onlyLPN_nodeInventoryDtl_vw.sql
- CustomDBViews/transaction/yfs_onlyLPN_NoPendMove_nodeInventoryDtl_vw.sql
- CustomDBViews/transaction/yfs_order_release_line.sql
- CustomDBViews/transaction/yfs_order_release_line_vw.sql
- CustomDBViews/transaction/yfs_nodeInventoryDtl_vw.sql
- CustomDBViews/configuration/yfs_wave_item_vol_config_vw.sql
- CustomDBViews/master/ycm_pricelist_vw.sql
- CustomDBViews/master/ypm_category_item_vw.sql
- CustomDBViews/master/ypm_item_vw.sql

Enabling the Text Search Feature on DB2

About this task

To enable the text search feature on DB2 database:

Procedure

1. Make sure that the DB2 database is configured with the Net Search Extender plug-in.
2. Log in to the DB2 server using the Command Editor or Command Line Processor with a user ID having DBA privileges.
3. Verify that the text search index creation was successful.

   Use the `customer_overrides.properties` file that is located in the `<INSTALL_DIR>/properties` directory to set the `yfs.db.textsearch` property to `Y`.

Results

The text search indexes that are created on DB2 database using the `<INSTALL_DIR>/repository/scripts/EFrame_TextIndexAdds.sql` script are automatically updated every 6 hours. The DBA can modify this script to change this frequency, if necessary. Before running the `EFrame_TextIndexAdds.sql` script, the DBA must update the "/*Database*/" string in the `EFrame_TextIndexAdds.sql` script and specify the database name.

For information on how to create the text search indexes, see the *Sterling Selling and Fulfillment Foundation: Customizing Console JSP Interface for End-User* .

Chapter 6. Installing and Configuring Database Tier Software on Windows

Database Configuration on Windows-Overview

This chapter describes how to install and configure the database tier software to run Sterling Selling and Fulfillment Foundation in a Windows environment.

Before installing your database server, verify that you have the applicable software versions. For more information, go to the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning.

If you are planning a multischema installation, you must deploy the same database vendor and version across all deployments.

Configuring a Database Server on Windows

You must install, create, and configure a database so that each Sterling Selling and Fulfillment Foundation instance has a dedicated schema and login for the database.

Configuring a Database for a Multiple Schema Environment

If you are planning a multischema installation, you must deploy the same database vendor and version across all deployments.

Generating Data Backup and Restore Scripts

Procedure

To generate the backup and restore scripts, run the `backupScriptGen.xml` script located in the `<INSTALL_DIR><INSTALL>/bin` directory using the following command:
`sci_ant.cmd -f backupScriptGen.xml -DdbType=<database_type>`
`<INSTALL><INSTALL>`This script generates sample backup and restore scripts in the `<INSTALL_DIR><INSTALL>/bin/sample` directory.
You can rename and customize the scripts to suit your business needs. For example, you can modify the script to add your custom configuration tables and modify the path where the data files are stored. These scripts depend on utilities provided by the database vendors.
The `backupScriptGen.xml` script accepts the following arguments:

Table 2. backupScripGen.xml Arguments

Argument	Purpose	Accepted Values
`-Dos=`	Determines what kind of script is generated. If "windows" is selected, a .cmd file is created. If "linux" or "unix" is selected, a .sh file is created. If "all" is selected, both .cmd and .sh files are created.	• windows • unix • linux • all
`-DdbType=`	Determines for which databases the scripts will be generated.	• oracle • db2 • all
`-DtableType=`	Determines which entities to generate scripts for. Valid values are any TableType attribute defined for an entity.	• `ALL` • `CONFIGURATION` • `MASTER` • `METADATA` • `STATISTICS` • `TRANSACTION` **Note:** The value `ALL` is supported only for single-schema mode.

Results

Running the `backupScriptGen.xml` script creates both backup and restore scripts for the selected operating systems and databases.

Oracle scripts depend on `export`, `import`, or `sqlplus` utilities. You can modify and use the following scripts:

- `backup_config_oracle.cmd`
- `restore_config_oracle.cmd`
- `delete_configuration_oracle.sql`

Configuring an Oracle Database on Windows

About this task

You can use an Oracle database for maintaining information on Sterling Selling and Fulfillment Foundation. To configure an Oracle database for use in a Sterling Selling and Fulfillment Foundation production environment, you must:

Procedure

1. Create the database. Refer to the Oracle documentation for information about creating the database, including creating a schema repository, login, and tablespace.
2. Size the database by estimating the required disk space.
3. Set the database connection properties.

4. Set the Oracle database parameters.
5. Grant Permissions in Oracle.

Setting Database Parameters in Oracle

For information about required parameter settings in your Oracle database, see the *Sterling Selling and Fulfillment Foundation: Performance Management Guide*.

Rolling Back or Undoing Changes in Oracle

You can roll back or undo changes in Oracle using the AUTO UNDO management option. It is recommended that you use this option. This avoids manual monitoring of UNDO segments.

Configuring Oracle for Single or Multiple Byte Characters

About this task

To set up Oracle for use with single or multiple byte characters:

Procedure

1. Run the Oracle create instance procedure. Use a character set appropriate for your desired language. For example:

   ```
   CHARACTER SET "UTF8"
   ```

2. Configure the `INIT<INSTANCE_NAME>.ORA` file for Oracle as follows:

   ```
   open_cursors= <set to appropriate value>
   ```

   For example, the minimum value for WebLogic equals number of threads (across all application servers) + (connection pool size X prepared statement pool size)

   ```
   cursor_sharing=FORCE
   compatible=<set to appropriate value, or remove to default to current release>
   timed_statistics=true
   db_block_size=8192
   optimizer_mode=ALL_ROWS
   ```

   If you are using multi-byte character set, set the following and restart Oracle:

   ```
   nls_length_semantics=CHAR
   ```

   Alternatively you can run the following prior to running any create table scripts:

   ```
   alter session set nls_length_semantics = CHAR
   ```

   Setting this attribute ensures that the field sizes are not impacted by the number of bytes a data type can store. For example, Varchar(40) would now be able to store 40 Japanese characters instead of 40/3 bytes in the UTF-8 character set.

   For the Japanese locale, the AL32UTF-8 character set or the UTF-16 character set must be used.

   When you change the multi-byte character set to CHAR by setting nls_length_semantics = CHAR, Oracle reserves space equivalent to 'n' chars, which is more than 'n' bytes. Therefore, when you run the `dbverify.cmd` command, the reduced entries in table columns are printed in the `EFrame_Drops.lst` file.

3. Download the Oracle JDBC driver for the version of Oracle you are using from the Oracle Web site and copy it to a well known location for reference during installation.

   The Oracle JDBC driver can be found in the JDBC Driver Downloads section at:

   ```
   http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html
   ```

For Oracle database supported version information, go to the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning

Results

To create the Oracle database to handle multiple byte characters, do not modify the Sterling Selling and Fulfillment Foundation DDL.

Oracle Database User Privileges

Unless specifically stated for a given task, the Sterling Selling and Fulfillment Foundation user does not require database administrator (DBA) privileges.

Administrator Privileges

The following list includes some of the basic privileges that should be granted to the Sterling Selling and Fulfillment Foundation administrative user who creates or modifies the Oracle database:

- Alter any sequence
- Alter session
- Create any sequence
- Create procedure
- Create sequence
- Create session
- Create synonym
- Create table
- Create trigger
- Create type
- Create view
- Delete any table
- Execute any procedure
- Execute any type
- Connect
- Insert any table
- Select any dictionary
- Select any sequence
- Select any table
- Select catalog role
- Update any table

Assign the user responsible for creating and modifying the Oracle database a specified quota (extent) in the tablespace, even if that user was assigned an unlimited tablespace when created. Otherwise, the installer will generate a "ORA-01950: no privileges on tablespace name" error.

Application User Privileges

The following list contains basic privileges that should be granted to users who will only run the application.

The Privileges/Grant is only for the tables owned by the user. The following is an example script to create table update grants:

```
select 'grant update on '||table_name||' to [user];'
from user_tables
```

You can write additional scripts to create similar permissions for the following privileges:

- Alter session
- Create session
- Delete table
- Execute procedure
- Insert table
- Select sequence
- Select table
- Update table
- If you are using text indexes, you must also have privileges for CTXAPP or CTXCAT, depending on the type of text indexes you are using.

Manually Creating Views on Oracle

To configure your Oracle database for your production environment, you must set up and run two scripts, Interop_Views.sql and ImportExport_View.sql, to create the views for your schema.

These script files reside in the <INSTALL_DIR>\database\oracle\scripts\CustomDBViews\transaction directory.

Table, index, and sequence create DDLs are created during installation. These reside in the install_dir\repository\scripts directory.

If you plan to use the silent install method and set the -nodbverify option to true, you will also need to create the tables, indexes, sequences, and so forth for your schema.

Refer to "Enabling the Oracle Database Text Search Feature - Overview" on page 94 for information about text search features on the Oracle database.

Enabling Failover in a Multiple Node Oracle RAC Database Cluster in Windows

About this task

To enable failover in a multiple node Oracle RAC database cluster in Windows, do the following:

Procedure

1. Navigate to the `<INSTALL_DIR>/properties` directory, where you will modify the `sandbox.cfg` and `customer_overrides.properties` files.

   Note: You might need to create the `customer_overrides.properties` file, which is just for customizations and is not automatically created during an installation. For additional information about the `customer_overrides.properties` file, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.

2. In the sandbox.cfg file, add a new property for ORACLE_JDBC_URL which contains the Oracle RAC connection URL.

   The following example shows the suggested URL form. This example shows how the information is organized, but the property value must be one string of text, starting with ORACLE_JDBC_URL=. Your database administrator (DBA) can modify this URL as needed.

```
jdbc:oracle:thin:@
   (DESCRIPTION=
      (ADDRESS_LIST=
         (FAILOVER=ON)
         (LOAD_BALANCE=ON)
         (ADDRESS=(PROTOCOL=TCP)(HOST=myhost1)(PORT=1521))
         (ADDRESS=(PROTOCOL=TCP)(HOST=myhost2)(PORT=1521))
        )
        (CONNECT_DATA = (SERVER = DEDICATED)(SERVICE_NAME = myservicename))
)
```

3. In the customer_overrides.properties file, add the readTimeout property to all Oracle database pools. These values override the corresponding values in the jdbc.properties file.
   - jdbcService.oraclePool.prop_jdbc.readTimeout=90000
   - jdbcService.oraclePool_local.prop_jdbc.readTimeout=90000
   - jdbcService.oraclePool_NoTrans.prop_jdbc.readTimeout=90000
   - jdbcService.oraclePool_NoTrans.prop_jdbc.readTimeout=90000

   The readTimeout value will require tuning. If the value is too low, long-running queries in the system will be interrupted. If the value is too high, recovery when a RAC node fails will be delayed.
   For additional information about overriding properties using the customer_overrides.properties file, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.
4. Run the setupfiles.sh command from the <INSTALL_DIR>/bin directory.
5. Set the propagation delay on the RAC server to 0.

Chapter 7. Installing Sterling Selling and Fulfillment Foundation in a Windows Environment

Use the topics in this section to prepare for installation, run the installation, and as a reference for post-installation tasks.

High Level Overview of Installation Process

After completing the previously described security, database, and application tier software tasks, the installation process going forward consists of the tasks that are described in brief here. Each task is covered in more detail in subsequent sections.

- Download purchases from Passport Advantage®, or gather the product DVDs. See the Passport Advantage site for more information: http://www-01.ibm.com/software/howtobuy/passportadvantage/
- Ensure that you have the installation packages uncompressed on the hard drive of the machine where you are going to install Sterling Selling and Fulfillment Foundation, or have the DVDs available.
- Download IBM Installation Manager and install it. See "Install the IBM Installation Manager" on page 40.
- Create an Installation Manager repository for the Sterling Selling and Fulfillment Foundation files.
- Use the Installation Manager to install Sterling Selling and Fulfillment Foundation.
- Complete any necessary post-installation tasks. See "Post Installation Tasks for Windows" on page 53.
- Configure the application server and create and deploy the EAR file for Sterling Selling and Fulfillment Foundation. See Chapter 14, "Deploying Sterling Selling and Fulfillment Foundation," on page 109.
- The documentation for this release of Sterling Selling and Fulfillment Foundation is located in an IBM Information Center: http://www.ibm.com/support/entry/portal/Planning. You can also download the entire information center as a compressed file and deploy it on a server at your location. This is useful if employees who will use Sterling Selling and Fulfillment Foundation do not have access to the internet. See Chapter 11, "Using a Local Version of the Information Center," on page 87 for more information.

Installation Worksheet for Windows

Use this worksheet as a single place to collect all the information you will need while doing an installation of Sterling Selling and Fulfillment Foundation in a Windows environment.

Table 3. Installation Worksheet for Windows

Description	References	Your Notes
Memory parameter settings for your operating system: ADDITIONAL_ANT_JAVA_TASK_ARGS ADDITIONAL_ANT_COMPILER_ TASK_ARGS	"Memory Parameter Settings" on page 39 For information about other system requirements, go to the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning.	
Location of Installation Manager on your system	IBM Installation Manager information center:http://publib.boulder.ibm.com/infocenter/install/v1r5/index.jsp	
Name and location of product repository and packages to be installed	IBM Installation Manager information center:http://publib.boulder.ibm.com/infocenter/install/v1r5/index.jsp and the Sterling Selling and Fulfillment Foundation Support page: http://www.ibm.com/support/entry/portal/Planning	
Location to use for shared resources directory, if default would cause issues with previously installed products or versions (see the Installation Manager documentation for more information)	IBM Installation Manager information center:http://publib.boulder.ibm.com/infocenter/install/v1r5/index.jsp	
Path to and name of directory where you plan to install Sterling Selling and Fulfillment Foundation	"Installation Directory Information for Windows" on page 40	
Do you want to install for a multiple schema environment?	*Sterling Selling and Fulfillment Foundation: Multi-Tenant Enterprise Guide*	
Are you doing an upgrade from a prior version of Sterling Selling and Fulfillment Foundation?	If you are doing an upgrade, stop and review the *Sterling Selling and Fulfillment Foundation: Upgrade Guide* before proceeding with the upgrade.	
Are you going to manually or automatically apply database definition language (DDL) statements (schema) to the database?	"About Applying Database Definition Language (DDL) Statements After a Windows Installation" on page 41	
Location of JDK on your system		
Record the following information about your Oracle or DB2 database. This information may be case sensitive.		
Database user name and password (you should keep the password in a separate place for security purposes, instead of writing it here)		

Table 3. Installation Worksheet for Windows (continued)

Description	References	Your Notes
Database catalog name	This is typically the database name; also known as SERVICE_NAME or SID in some versions of Oracle.	
Database host name (or IP address)		
Oracle: Absolute path and file name for the JDBC driver file.		
DB2: Absolute paths and file names for the JDBC driver file and license file.		
Database schema owner		
DB2 only: Do you want to use multibyte support?		
If you are planning a multischema deployment, determine and record the database information for four separate schemas: Metadata, Statistics, System Configuration, and Transaction/Master Data	*Sterling Selling and Fulfillment Foundation: Multi-Tenant Enterprise Guide*	
Are you planning to run a silent installation?	"About Recording Silent Installation Files for Windows" on page 41	

Memory Parameter Settings

During an interactive installation, you are prompted for memory parameters for your system. These parameter values are written to the sandbox.cfg property file. After installation, you can edit the values as needed for performance and tuning.

The minimum memory requirements for Sterling Selling and Fulfillment Foundation are based on the products you are installing.

Table 4. Minimum Memory Parameter Settings by Operating System

Operating System	Parameters
Solaris 10 on SPARC processor (64-bit AMD)	-J-Xms256m -J-Xmx1408m -XX:MaxPermSize=768m
Windows 2008	-J-Xms1024m -J-Xmx2048m -XX:MaxPermSize=512m
IBM AIX® 6.1 TL5 POWER System	-J-Xms1024m -J-Xmx1536m
IBM AIX 7.1 POWER System	-J-Xms1024m -J-Xmx1536m
HP-UX 11i v3 on Itanium	-J-Xms256m -J-Xmx1408m -XX:MaxPermSize=512m
Red Hat Enterprise Linux 6.0/AP 64-bit Xeon or AMD processor	-J-Xms1024m -J-Xmx1664m -XX:MaxPermSize=512m
SUSE Linux Enterprise 11 on 64-bit Xeon or AMD processor	-J-Xms1024m -J-Xmx1664m -XX:MaxPermSize=512m

Note: For information about how the memory parameters relate to the ADDITIONAL_ANT_JAVA_TASK_ARGS and ADDITIONAL_ANT_COMPILER_TASK_ARGS properties, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*

Installation Directory Information for Windows

If installing in a Windows environment, read the following points to help you plan your installation.

- It is not recommended to install Sterling Selling and Fulfillment Foundation as an administrator because this would cause folders that are created by the installation to be inaccessible to anyone but the administrator.
- The silent installation creates an *INSTALL_DIR**install* folder. Subdirectories are installed directly under it.
- The interactive installation process installs subdirectories and files directly into the *INSTALL_DIR* you specify while running the Installation Manager.

Installation Methods on Windows

Interactive Installation Using IBM Installation Manager

The IBM Installation Manager steps through the Sterling Selling and Fulfillment Foundation installation for a Windows system in one of three ways:

- A GUI-based installation wizard.
- A text-based installation wizard.
- A silent installation file, which is written if you record a GUI or text-based installation.

Before performing any of these installation methods, you must install the IBM Installation Manager.

Install the IBM Installation Manager

About this task

You must download and install the IBM Installation Manager before installing Sterling Selling and Fulfillment Suite.

Important:

You must install and launch Installation Manager as a normal user not as a root user. Before you begin installation, review the documentation for the IBM Installation Manager. The Installation Manager has multiple installation and deployment scenarios, so read the documentation carefully and decide which scenario fits your needs before proceeding. The Installation Manager information center is available here: http://publib.boulder.ibm.com/infocenter/install/v1r5/index.jsp.

Procedure

1. Determine the version of Installation Manager supported for Sterling Selling and Fulfillment Suite. For more information, go to the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning.
2. Go to the Installation Manager downloads page: http://www.ibm.com/support/entry/portal/All_download_links/Software/Rational/IBM_Installation_Manager
3. Locate and download the correct package for your operating system.
4. Extract the contents of the compressed file to a temporary directory.
5. Browse to the InstallerImage folder and run the installation command file for your operating system:

userinst.exe

6. Follow the prompts in the wizard to install the Installation Manager.

Add a Repository for Sterling Selling and Fulfillment Foundation in IBM Installation Manager

About this task

After Installation Manager is installed, you must add a repository that contains the files for the Sterling Selling and Fulfillment Foundation installation.

Procedure

1. Start Installation Manager.
2. Select File > Preferences and then select the Repositories tab from the left pane.
3. Click Add Repository.
4. Browse to the output folder that contains the files for Sterling Selling and Fulfillment Foundation.
5. Select the repository.config file and click OK.
6. Verify that the new repository is shown in the list of repositories and that its checkbox is selected.
7. Click Apply, then OK.

About Recording Silent Installation Files for Windows

About this task

If you prefer, you can invoke the IBM Installation Manager from a command line so that it steps through the Sterling Selling and Fulfillment Foundation installation and records a silent installation file. You can modify and use this file to run the installation again.

Optionally, you can also record multiple silent installation files.

The topics "Running a GUI-Based Installation on Windows" on page 42 and "Running the Text-Based Interactive Installation Program on Windows" on page 44 explain how to record a silent file for installation. See "Running a Silent Installation on Windows" on page 47 for information about running a silent installation file.

For information about recording and running multiple silent installations, see "Running Multiple Silent Installations on the Same Machine" on page 47.

About Applying Database Definition Language (DDL) Statements After a Windows Installation

When you install Sterling Selling and Fulfillment Foundation, you can manually apply database definition language (DDL) statements to your database tables instead of requiring the installation process to do it directly. This enables you to apply DDL statements for database creation separately from the installation.

This feature increases database security by reducing the database permissions of the Sterling Selling and Fulfillment Foundation database user. The rights to create

tables, indexes, etc. can be reserved for a secure user like a database administrator (DBA). A business can require that only a DBA with the proper permissions can make database changes.

If you choose to manually apply the DDL, the installation process will provide the location of the DDL scripts. The installation process will then continue the rest of the installation. The installation process may validate the database with the dbverify script and warn you if there are differences, but it will not exit. It will allow the processing of the packages to continue normally.

If you do not choose to manually apply the DDL, the installation will apply both the DDL and the resources.

During an interactive installation, you indicate whether or not you want to set up DDLs and run factory setup when selecting features. Select Enable Upgrade if you are upgrading from a previous release **or** if you are doing a new installation, but want to apply DDL manually after installation. Do not select this option if you are doing a new multischema installation. If you are not upgrading and you select this option, you must manually create your database tables and load factory setup after the installation process. However, if you are upgrading, the DDLs are applied as part of the upgrade process.

See the section about configuring your database in Chapter 6, "Installing and Configuring Database Tier Software on Windows," on page 31 for information about running view scripts after the initial installation and Chapter 13, "Configuring Utilities," on page 93 for information about manually installing the database DDLs and factory setup.

Running an Installation on Windows

Use the topics in this section to install Sterling Selling and Fulfillment Foundation.

Running a GUI-Based Installation on Windows

About this task

Before you can complete this task, you must install the IBM Installation Manager and create a repository. If you are downloading your installation files from PassportAdvantage, you must also complete that task first.

To prepare for installing Sterling Selling and Fulfillment Foundation in a Windows environment, refer to your "Installation Worksheet for Windows" on page 38 and follow the steps below.

Procedure

1. Close all open Windows programs and any command prompt windows.
2. If you wish to record the installation to a silent file rather than perform an actual installation, open a command prompt window and issue the following command line. Otherwise, skip to Step 3.
   - Navigate to the **\IBM\Installation Manager\eclipse** folder.
   - Enter `IBMIM.exe -record response.xml -skipInstall <IM_data_dir>`

     ...where response.xml is the silent installation file to be generated and *<IM_data_dir>* is a new directory generated for the Installation Manager to write and store data into. For example:

     ```
     IBMIM.exe -record response.xml -skipInstall C:\SterlingInstallData
     ```

The IBM Installation Manager opens and begins the installation process. Skip to Step 4.

3. To perform an installation without recording, open the IBM Installation Manager by clicking on the icon or entering the following command line:
   - `IBMIM.exe`

   The Installation Manager main menu is displayed.
4. Click Install.
5. Select the packages to be installed.
6. Review and accept the license agreement.
7. Accept the default location for the shared resources directory, or browse to a new location.
8. Accept the default location for the product to be installed, or enter a different location. If you specify an existing folder, it must be empty.
9. Select the features to install:
   - Product Files - selected by default.
   - Enable Multi-Schema - select if doing an install with multiple database schemas
   - Enable Upgrade - select if you are upgrading from a previous release **or** if you are doing a new installation, but want to apply DDL manually after installation. Do not select this option if you are doing a new multischema installation.
10. Enter the location and name of the JDK home directory. Copying the JDK to your installation directory is selected by default. If you do not want a copy of the JDK made under your installation directory, clear the checkbox.
11. Select your database, Oracle or DB2, from the list, then enter the information for the remaining database fields.
    a. If doing a single schema installation, enter information for the following fields:
       - Database user name
       - Database password/Confirm database password
       - Database catalog name
       - Database host name (or IP address)
       - Database port
       - Database schema name

         Note: The default schema names are always the same as the user name, except that the schema names are all uppercase. This applies to both single and multischema scenarios.
       - Click Add Jars to browse to the JDBC driver file for Oracle or DB2. Repeat to add the license file for DB2 (required).

       To test the connection to the database, click Validate Connection. If a message saying the connection failed is displayed, verify the information you entered in all fields and try validating again.
    b. If doing a multischema installation, click Add Jars to browse to the JDBC driver file for Oracle or DB2. Repeat to add the license file for DB2 (required). For each schema type (Metadata, Statistics, Configuration, Transaction), enter the database information:
       - Database user name
       - Database password/Confirm database password

- Database catalog name
- Database host name (or IP address)
- Database port
- Database schema name

  Note: The default schema names are always the same as the user name, except that the schema names are all uppercase. This applies to both single and multischema scenarios.

  For each schema, you can test the connection to the database by clicking Validate Connection. If a message saying the connection failed is displayed, verify the information you entered in all fields and try validating again.

12. On the Database Details screen:
    - If using DB2, choose whether to use multibyte support.
    - If using Oracle, there are no additional database properties to configure; click Next.
13. Enter the ADDITIONAL_ANT_JAVA_TASK_ARGS and ADDITIONAL_ANT_COMPILER_TASK_ARGS memory parameter settings for your operating system.
14. Click Install to start the installation. A progress bar is displayed at the bottom of the screen.
15. When the installation is complete, a status screen is displayed. Click Finish. You can also view the installation log from this screen.

    Note: If the installed_data directory is not removed by the post installation cleanup process, you can delete it. It is only used for installation.
16. Exit Installation Manager.
17. See "Post Installation Tasks for Windows" on page 53 for configuration procedures that you must complete prior to running Sterling Selling and Fulfillment Foundation.

Running the Text-Based Interactive Installation Program on Windows

About this task

Before you can complete this task, you must install the IBM Installation Manager and create a repository. If you are downloading your installation files from PassportAdvantage, you must also complete that task first. To prepare for installing Sterling Selling and Fulfillment Foundation in a Windows environment, refer to your "Installation Worksheet for Windows" on page 38 and follow the steps below.

If you are upgrading from a previous release, stop any running instances of the previous installation.

Procedure

1. Open and command prompt window and navigate to the **\IBM\Installation Manager\eclipse\tools** folder.
2. If you wish to record the installation to a silent file rather than perform an actual installation, issue the following command line. Otherwise, skip to Step 3.

- Enter `imcl.exe -c -record response.xml -skipInstall <IM_data_dir>`

  ...where response.xml is the silent installation file to be generated and *<IM_data_dir>* is the directory generated for the Installation Manager to write and store data into. For example:

  ```
  imcl.exe -c -record response.xml -skipInstall C:\SterlingInstallData
  ```

  The IBM Installation Manager opens and begins the installation process. Skip to Step 4.

3. To perform an installation without recording, enter the following command: `imcl.exe –c`
4. Type 1 to select Installation and press Enter.
5. Type 1 to select the IBM Sterling Selling and Fulfillment Suite 9.2.0 packages to install and press Enter.
6. Type 1 to choose version 9.2.0 for installation and press Enter.
7. Press Enter on the confirmation screen to move to the next screen.
8. Read each license agreement by typing the option number (1 or 2). After reading, type R to return to the prompt. When ready to accept the license agreements, type A and press Enter.
9. Choose a location for the Shared Resources directory. The default is IBM\SDPShared. To choose a different directory, type M and press Enter. Type the path and directory name and press Enter.
10. Press Enter on the confirmation screen to move to the next screen.
11. Choose an installation location. The default is the IBM directory. If you select a directory that already exists, an error message is displayed. To choose a different directory, type M and press Enter. Type the path and name of the new directory and press Enter.
12. Press Enter on the confirmation screen to move to the next screen.
13. Select the features to install:
    - 1. Product Files - selected by default.
    - 2. Enable Multi-Schema - select if doing an install with multiple database schemas
    - 3. Enable Upgrade - select if you are upgrading from a previous release **or** if you are doing a new installation, but want to apply DDL manually after installation. Do not select this option if you are doing a new multischema installation.

    To select Multi-Schema, type 2 and press Enter. To select Upgrade, type 3 and press Enter. After you have selected one option, the prompt is displayed again. You can either choose another option (2 or 3) or press Enter to accept the default (N) and continue to the next step.
14. Enter the path and name for the JDK home directory and press Enter.
15. Choose whether to have the installer create a copy of the JDK folders under your installation directory. The default is Yes (create the copy). Press Enter to accept the default and continue. If you do not want a copy of the JDK made, type N and press Enter.
16. Type the option number for the database, DB2 or Oracle, and press Enter.
17. Enter the information for the remaining database fields, depending on whether you chose single or multischema earlier:
    a. If doing a single schema installation, enter information for the following fields:
       - Database user name

- Database password (note that the password is displayed in clear text)
- Database catalog name
- Database host name (or IP address)
- Database port number
- Database schema name

  Note: The default schema names are always the same as the user name, except that the schema names are all uppercase. This applies to both single and multischema scenarios.
- Enter the location and name of the JDBC driver file for Oracle or DB2. Type A to add the license file for DB2 (required). After adding the license file and the prompt is displayed again, type D (done) to continue to the next step.

To validate the database connection, type Y (validate the connection), or N (do not attempt to connect). If a message saying the connection failed is displayed, verify the information you entered in all fields and try validating again.

b. If doing a multischema installation, enter the location and name of the JDBC driver file for Oracle or DB2. Type A to add the license file for DB2 (required). After adding the license file and the prompt is displayed again, type D (done) to continue to the next step. For each schema type (Metadata, Statistics, Configuration, Transaction), enter the database information:
   - Database user name
   - Database password (note that the password is displayed in clear text)
   - Database catalog name
   - Database host name (or IP address)
   - Database port number
   - Database schema name

     Note: The default schema names are always the same as the user name, except that the schema names are all uppercase. This applies to both single and multischema scenarios.

   To validate the database connection for each schema, type Y (validate the connection), or N (do not attempt to connect). If a message saying the connection failed is displayed, verify the information you entered in all fields and try validating again. Once all information is entered for a schema, you can choose B (back) if you want to change information, or choose N (next) to continue to the next schema or step.

18. If using DB2, choose whether to enable database multibyte support (Y or N) and press Enter.
19. Enter the ADDITIONAL_ANT_JAVA_TASK_ARGS memory parameter settings for your operating system and press Enter..
20. Enter the ADDITIONAL_ANT_COMPILER_TASK_ARGS memory parameter settings for your operating system and press Enter. Press Enter again to confirm your choices.
21. Type G (Generate an Installation response file) to have an XML file that contains the install prompts and your responses created.
22. When prompted, type the location and name for the file and press Enter. Use "xml" as the file extension.

23. Once a message saying that the file was successfully created is displayed, press Enter. The previous screen is displayed again, with the same options. This time, type I (Install) to start the installation. A progress bar is displayed at the bottom of the screen.
24. When the installation is complete, a status screen is displayed.
25. See "Post Installation Tasks for Windows" on page 53 for configuration procedures that you must complete prior to running Sterling Selling and Fulfillment Foundation, as well as optional tasks, and post installation tips.

Running a Silent Installation on Windows

About this task

Use the following instructions to install in a Windows environment from a command line, using the response.xml silent installation file. For information about recording a silent file, see "About Recording Silent Installation Files for Windows" on page 41.

Procedure

1. Open a command prompt window and navigate to the **\IBM\Installation Manager\eclipse\tools** folder.
2. Enter `imcl.exe input response.xml -log <log_filename>log`

   ...where response.xml is the silent installation file to be used for installation and <*log_filename*>.log is the log file to which installation logging data will be written. For example:

   ```
   imcl.exe input response.xml -log C:\Installation.log
   ```
3. See "Post Installation Tasks for Windows" on page 53 for configuration procedures that you must complete prior to running Sterling Selling and Fulfillment Foundation.

Running Multiple Silent Installations on the Same Machine

About this task

You can perform multiple silent installations on the same machine by using the following steps.

Procedure

1. Complete an initial installation recording in GUI or text mode, as described in "Running a GUI-Based Installation on Windows" on page 42 or "Running the Text-Based Interactive Installation Program on Windows" on page 44.
2. Modify the silent installation file, `response.xml`, by editing the permissible values as needed. "Windows Silent Install File Parameters" on page 48 contains information about the parameters that you can modify.
3. You can create multiple silent response files in either one of two ways:
   - **Choice 1:** You can save `response.xml` to create multiple variations of it, as long as they are named differently; for example, `response1.xml`, `response2.xml`, and so forth. You must also ensure that they have different profile IDs in them. Edit each `response.xml` so that it has a unique profile ID number, such as 1, 2, 3, and so forth. Edit both instances of the profile ID that are in the file. For example:

     ```
     <profile id='IBM Sterling Selling And Fulfillment Suite_1'
     installLocation=...>
     ```

<offering id='...' version='...' profile='**IBM Sterling Selling And Fulfillment Suite_1**' features='...' installFixes='none'/>

- **Choice 2:** If you prefer to create silent file variations by recording them to different response<*n*>.xml files by using the IBM Installation Manager multiple times instead of manually editing the profile ID, you will achieve the same outcome of having multiple silent installation files. This is because each time you record a silent installation, the IBM Installation Manager increments the profile ID number in the response.xml file so that each installation file is unique.

  To use this option, launch the IBM Installation Manager again and record this installation, specifying a new silent file; for example,response3.xml.

  For a GUI installation, from the **\IBM\Installation Manager\eclipse** folder, enter: IBMIM.exe -record response3.xml -skipInstall <*IM_data_dir*>

  For a text installation, from the **\IBM\Installation Manager\eclipse\tools** folder, enter: imcl.exe -c -record response3.xml -skipInstall <*IM_data_dir*>

  Note: <*IM_data_dir*> must be the same directory for all recordings.

  During the recording, the Installation Manager will see that Sterling Selling and Fulfillment Foundation has already been installed. Click Continue to indicate that you want to install it again. This appends a number to the existing profile ID.

4. Using either method, you can create multiple silent installation files.

   For information about installing with a silent file, see "Running a Silent Installation on Windows" on page 47

Windows Silent Install File Parameters

Important: The following silent installation parameters are the only ones that you can change. If you change the values for parameters that are not listed in this table, the installation will fail.

Key Name	Description
user.sbx.INSTALL_JAR	The JAR which contains the installation files. Example: user.sbx.INSTALL_JAR value=SSFS_INSTALL.JAR
user.sbx.DB_CLEAN	Valid values are true/false. If true, this property deletes all objects from the database. The installation sets this property to false.
user.sbx.JVM_LOC	Source of the downloaded JDK files, external to Sterling Selling and Fulfillment Foundation. Example: user.sbx.JVM_LOC value=<JDK_DIR>
user.sbx.OVERRIDE_LOAD_DEFAULTS_PK_GEN	To generate unique primary keys for entities across deployments, include this property in your silent installation file and set the property to true. Default is false. See the documentation about the Change Project Management feature for additional information. Example: user.sbx.OVERRIDE_LOAD_DEFAULTS_PK_GEN value=true

Important: The following silent installation parameters are the only ones that you can change. If you change the values for parameters that are not listed in this table, the installation will fail.

Key Name	Description
user.sbx.DB_VENDOR	(Required) The database vendor to use (Oracle, DB2). **Note:** In a multischema installation, you must deploy the same database vendor and version across deployments. Example: `user.sbx.DB_VENDOR value=<db_vendor>`
user.sbx.DB_USER	(Required) Database login ID with which to connect. In a multischema deployment, this must be the username for the Metadata schema. **Note:** The default schema names are always the same as the user name, except that the schema names are all uppercase. This applies to both single and multischema scenarios. Example: `user.sbx.DB_USER value=<db_user_name>`
user.sbx.DB_DATA	(Required) Database name to connect with. Example: `user.sbx.DB_DATA value=<db_data_catalog>`
user.sbx.DB_PASS	(Required) Database password with which to connect. In a multischema deployment, this must be the password for the Metadata schema. Example: `user.sbx.DB_PASS value=<db_password>`
user.sbx.DB_HOST	(Required) Host for database (for example, server or IP address). Example: `user.sbx.DB_HOST value=<db_host>`
user.sbx.DB_PORT	(Required) Database listener port to which to connect. Example: `user.sbx.DB_PORT value=<db_listener_port>`

Important: The following silent installation parameters are the only ones that you can change. If you change the values for parameters that are not listed in this table, the installation will fail.

Key Name	Description

user.sbx.DB_DRIVERS	(Required) Full path to the JDBC driver file(s).
	• Oracle requires one file.
	• DB2 requires two files: the license file and the driver file.
	If specifying more than one file, use a pipe character (|) for a separator between files.
	Examples:
	`user.sbx.DB_DRIVERS value=<absolute_path_to_driver_jar>`
	`user.sbx.DB_DRIVERS value=<JDBC_driver_dir>/jdbc.jar`
	`user.sbx.DB_DRIVERS value=<JDBC_driver_dir>db2_1_jdbc.jar|<JDBC_driver_dir>/db2_2_jdbc.jar`

user.sbx.DB_DRIVERS_VERSION	(Required) Free form version string for JDBC driver. This is informational only. Example: `user.sbx.DB_DRIVERS_VERSION value=<db_driver_version>`
user.sbx.DB_SCHEMA_OWNER	(Required for multischema mode) Default schema/schema-owner for the provided login ID. If you wish to change this value to an alternate schema, consult your database administrator, as this is considered an expert installation scenario and can be performed only through the silent installation.
user.sbx.LOAD_FACTORY_SETUP	Indicates whether you want to load factory setup defaults during installation (true) or manually after installation (false). If you are performing an installation in upgrade mode, set this property to false. Example: `user.sbx.LOAD_FACTORY_SETUP value=true` For information about manually loading the factory defaults, see "Loading the Sterling Selling and Fulfillment Foundation Database Factory Defaults After Installation" on page 94.
user.sbx.NO_DBVERIFY	Valid values are true or false. When set to true during installation and installservice, dbverify will not be run. This means that Sterling Selling and Fulfillment Foundation will not generate DDL to make the database like the XML entity repository. If you are performing an installation in upgrade mode, set this property to true. Example: `user.sbx.NO_DBVERIFY value=true`

Important: The following silent installation parameters are the only ones that you can change. If you change the values for parameters that are not listed in this table, the installation will fail.

Key Name	Description
user.sbx.REINIT_DB	Valid values are true or false. By default, the value is set to true. If the value is set to false, the Sterling Selling and Fulfillment Foundation installation will complete successfully, but no database operation will be performed as part of the installation process. If you are performing an installation in upgrade mode, set this value to false. Example: `user.sbx.REINIT_DB value=true`
user.sbx.multischema.applyddl	Valid values are true/false. If set to true (default), enables the DBVerify script to generate and apply database DDLs automatically. If set to false, allows DBVerify script to generate the DDLs but does not apply them. This property is set to false in the GUI installer, by default. If you are using the GUI Installer and do not want to apply DDLs, ensure that this property to set to false in the sandbox.cfg file. If this property is set to true or is absent in the sandbox.cfg file, the DBVerify script generates the DDLs and applies them. Otherwise, the scripts are generated and not applied. Example: `user.sbx.multischema.applyddl value=true`
user.sbx.multischema.enabled	Valid values are true/false. If true, this attribute indicates that this is a multischema installation. The silent file, response.xml, specifies database information for the Configuration, Metadata, Transaction, and Statistics schemas. **Note:** This attribute is case-sensitive. Example: `user.sbx.multischema.enabled value=true` See “About Running a Silent Installation in Multischema Mode” on page 52 for more information.
user.sbx.multischema.version	(Required only if you enable multischema.) This attribute indicates which version is being installed. For the Sterling Selling and Fulfillment Foundation Release 9.2.0, you must enter 9.2.0. **Note:** This attribute is case-sensitive. Example: `user.sbx.multischema.version value=9.2.0`
user.sbx.multischema.file	(Required if you enable multischema.) This attribute indicates the name of the user-defined XML file that contains multischema database information. **Note:** This attribute is case-sensitive. Example: `user.sbx.multischema.file value=<`*`filename`*`>.xml`

Important: The following silent installation parameters are the only ones that you can change. If you change the values for parameters that are not listed in this table, the installation will fail.

Key Name	Description

user.sbx.ADDITIONAL_ANT_JAVA_TASK_ARGS
 	
user.sbx
ADDITIONAL_ANT_COMPILER_TASK_ARGS
	Go to the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning for memory parameter values based on your operating system.
	
	Examples:
	
	`user.sbx.ADDITIONAL_ANT_JAVA_TASK_ARGS value= -XX:MaxPermSize=nnnm`
	
	`user.sbx ADDITIONAL_ANT_COMPILER_TASK_ARGS value=-J-Xmsnnnm -J-Xmxnnnm`
	
	These parameter values are written to the sandbox.cfg file during installation. After installation, you can tune them if you are seeing Out-of-Memory errors.
	
	See the *Sterling Selling and Fulfillment Foundation: Properties Guide* for more information about these parameters.

user.sbx.COPYJVM	Valid values are true/false. By default, the installation creates a copy of the JDK under your installation folder.
user.sbx.ENTITY_GEN_LOGLEVEL	Specify VERBOSE to enable logging of verbose messages during entity class generation. Default=INFO Example: `user.sbx.ENTITY_GEN_LOGLEVEL value=VERBOSE`
user.sbx.GENERATE_ALL_DBCLASSES	Specify true to generate all DBClasses. Default=false Example: `user.sbx.GENERATE_ALL_DBCLASSES value=true`
user.sbx.SUPPORT_MULTIBYTE	If you are installing on DB2 and need to localize your database using a multibyte character set, set this flag to Y. This ensures that the database column sizes are large enough to handle the multibyte characters correctly. Default is N. Example: `user.sbx.SUPPORT_MULTIBYTE=Y`
user.sbx.ORACLE_JDBC_URL	Required only if Oracle RAC is being used as the database. Include the complete URL without any blank spaces. Example: `user.sbx.ORACLE_JDBC_URL=<url>`

About Running a Silent Installation in Multischema Mode

The response.xml silent file is written by the installation process when recording a multischema installation. The file specifies the database account information for

each of the four allowable four data tables that you use: Metadata, Statistics, System Configuration, and Transaction/Master.

It also shows parameters for specifying multiple passwords and their effective dates so that you can predefine passwords for a given pool months in advance. These passwords will change automatically without a server restart.

In a multischema installation, you must deploy the same database vendor and version across deployments.

"Windows Silent Install File Parameters" on page 48 explains the multischema parameters in response.xml.

For information about recording a silent response.xml file during installation, see "Running a GUI-Based Installation on Windows" on page 42 for GUI installations, or "Running the Text-Based Interactive Installation Program on Windows" on page 44 for text-based installations.

Post Installation Tasks for Windows

Security

- See "Web Security Planning-Post installation Recommendations" on page 9 for information about the security measures that are recommended for your consideration.

Database

- Views must be created manually. Instructions to create views vary depending on what database is used. All database view related scripts are located at <INSTALL_DIR>\database\<db_type>\scripts. For more information, refer to the section about configuring your database type (Oracle or DB2) for production in Chapter 6, "Installing and Configuring Database Tier Software on Windows," on page 31 and Chapter 13, "Configuring Utilities," on page 93.

Changing to Multischema Mode after Installation

- If you did not install or upgrade in multischema mode but you want to create the appropriate tables for multischema mode and update these tables at a later time, you can:
  - Set the following properties in the `sandbox.cfg` file:

    ```
    multischema.enabled=true
    multischema.version=9.2.0
    ```

  - Run the dbverify script on multischema colonies, as described in "Verifying the Database Schema" on page 96.

    If you want to create or add colonies to your deployment, see the *Sterling Selling and Fulfillment Foundation: Multi-Tenant Enterprise Guide* for instructions.

Starting and Stopping the Information Center

Procedure

- The Sterling Selling and Fulfillment Suite information center has to be started before users can access product help topics. To start the information center, run the script `install_dir\bin\startHelp.cmd`.

  Note: Ensure that the application server (which hosts the application EAR) and Sterling Selling and Fulfillment Suite Information Center are not running on the same machine. If they are running on the same machine, access the application

using the computer name (for example, http://abc.com:8001/sbc/sbc/login.do) and context-sensitive help using the IP address (for example, http://127.0.0.1:8002/help/index.jsp) or vice-versa. To change the IP address or Hostname for the context-sensitive help, edit the `EXT_HOST_ADDR` property in the `sandbox.cfg` file and run the setupfiles script.

- The information center has to be stopped prior to updating it, then restarted once updates are complete. To stop the information center, run the script `install_dir\bin\stopHelp.cmd`.

  Note: If an error occurs on running the stopHelp script, delete the workspace folder under `install_dir\xapidocs\ibm_help\eclipse`.

Web Security Planning-Post installation Recommendations

About this task

After the installation of Sterling Selling and Fulfillment Foundation, be sure to complete the following for ensured security:

Procedure

1. Change the password of the default user (admin).
2. Change permissions on *INSTALL_DIR*/bin/migrator.* files to non-executable.
3. API security is enabled during installation. After installation, you may want to reset the api.security.mode property and carefully consider your API security configuration. For more information about API security modes, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.
4. Ensure that the required permissions to access the API resources are defined for an application. For more information about configuring API security, see the *Sterling Selling and Fulfillment Foundation: Configuration Guide*.

Upgrade Information for Windows

If installing in upgrade mode, use the topics in this section for reference.

Upgrade Installation Considerations for Windows

Silent Installation

You can run an upgrade using the silent installation process. For information about running the silent installation in upgrade mode, see "Running a Silent Upgrade on Windows" on page 55. For information about creating a silent upgrade file, see "About Recording Silent Installation Files for Windows" on page 41.

Multischema Environments

- In a multischema installation, you must deploy the same database vendor and version across deployments.
- An installation in upgrade mode for multischema is supported only if you are upgrading from a previous release of multischema.

Database Considerations

If you are reinstalling Sterling Selling and Fulfillment Foundation, be aware that data in your existing database will be deleted. To prevent this, either back up the

existing database or save it under a different name. For more information about backing up and restoring data, see “Generating Data Backup and Restore Scripts” on page 31.

After creating and configuring your database, recycle the database. Then stop and restart Sterling Selling and Fulfillment Foundation to apply the changes.

Running a Silent Upgrade on Windows

About this task

Important: If performing an upgrade from a prior version of Sterling Selling and Fulfillment Foundation, multischema mode is supported only if you are upgrading from a multischema environment. You cannot use the multischema option when upgrading from a single schema instance of Sterling Selling and Fulfillment Foundation. The same is true if you are upgrading with a silent installation file.

Use the following instructions to start a silent upgrade in a Windows environment. Before running the command, ensure that:

- You have thoroughly reviewed the *Sterling Selling and Fulfillment Foundation: Upgrade Guide* and completed all necessary tasks from it.
- You have recorded a silent upgrade file. For information, see “About Recording Silent Installation Files for Windows” on page 41
- The silent upgrade file, `response.xml`, includes these settings: user.sbx.LOAD_FACTORY_SETUP value=false, user.sbx.NO_DBVERIFY value=true, and user.sbx.REINIT_DB value=false. “Windows Silent Install File Parameters” on page 48 contains information about these silent file parameters.
- If the application is running, stop the instance before proceeding.

Note:
If the `installed_data` directory is not removed by the post installation cleanup process, you can delete it. It is used only for installation.

To run the silent upgrade:

1. Navigate to the **\IBM\Installation Manager\eclipse\tools** folder.
2. Enter `IBMIM.exe input response.xml -log <log_filename>log`.

...where response.xml is the silent upgrade file to be used for upgrade and <*log_filename*>.log is the log file to which upgrade logging data will be written. For example:

```
IBMIM.exe input response.xml -log C:\Upgrade.log
```

Proceed with the silent upgrade.

The installation/upgrade dialogs are described in “Running a GUI-Based Installation on Windows” on page 42 and“Running the Text-Based Interactive Installation Program on Windows” on page 44.

Post Upgrade Tasks for Windows

If you have installed in upgrade mode and you want to create the appropriate tables for multischema mode and update these tables at a later time, you can:

- Set the following properties in the `sandbox.cfg` file:

```
multischema.enabled=true
multischema.version=9.2.0
```

- Run the dbverify script on multischema colonies, as described in "Verifying the Database Schema" on page 96.

  If you want to create or add colonies to your deployment, see the *Sterling Selling and Fulfillment Foundation: Multi-Tenant Enterprise Guide* for instructions.

Starting and Stopping the Information Center

Procedure

- The Sterling Selling and Fulfillment Suite information center has to be started before users can access product help topics. To start the information center, run the script `install_dir\bin\startHelp.cmd`.

  Note: Ensure that the application server (which hosts the application EAR) and Sterling Selling and Fulfillment Suite Information Center are not running on the same machine. If they are running on the same machine, access the application using the computer name (for example, http://abc.com:8001/sbc/sbc/login.do) and context-sensitive help using the IP address (for example, http://127.0.0.1:8002/help/index.jsp) or vice-versa. To change the IP address or Hostname for the context-sensitive help, edit the `EXT_HOST_ADDR` property in the `sandbox.cfg` file and run the setupfiles script.
- The information center has to be stopped prior to updating it, then restarted once updates are complete. To stop the information center, run the script `install_dir\bin\stopHelp.cmd`.

  Note: If an error occurs on running the stopHelp script, delete the workspace folder under `install_dir\xapidocs\ibm_help\eclipse`.

Chapter 8. Installing Sterling Selling and Fulfillment Foundation in UNIX and Linux Environments

Use the topics in this section to prepare for installation, run the installation, and as a reference for post-installation tasks.

High Level Overview of Installation Process on UNIX/Linux

After completing the previously described security, database, and application tier software tasks, the installation process going forward consists of the tasks that are described in brief here. Each task is covered in more detail in subsequent sections.

- Download purchases from Passport Advantage, or gather the product DVDs. See the Passport Advantage site for more information: http://www-01.ibm.com/software/howtobuy/passportadvantage/.
- Ensure that you have the installation packages uncompressed on the hard drive of the machine where you are going to install Sterling Selling and Fulfillment Foundation, or have the DVDs available.
- Download IBM Installation Manager and install it. See "Install the IBM Installation Manager" on page 40.
- Create an Installation Manager repository for the Sterling Selling and Fulfillment Foundation files.
- Use the Installation Manager to install Sterling Selling and Fulfillment Foundation.
- Complete any necessary post-installation tasks. See "Post Installation Tasks for UNIX/Linux" on page 75.
- Configure the application server and create and deploy the EAR file for Sterling Selling and Fulfillment Foundation. See Chapter 14, "Deploying Sterling Selling and Fulfillment Foundation," on page 109.
- The documentation for this release of Sterling Selling and Fulfillment Foundation is located in an IBM Information Center: http://www.ibm.com/support/entry/portal/Planning. You can also download the entire information center as a compressed file and deploy it on a server at your location. This is useful if employees who will use Sterling Selling and Fulfillment Foundation do not have access to the internet. See Chapter 11, "Using a Local Version of the Information Center," on page 87 for more information.

Installation Worksheet for UNIX and Linux

Use this worksheet as a single place to collect all the information you will need while doing an installation of Sterling Selling and Fulfillment Foundation in a UNIX or Linux environment.

Table 5. Installation Worksheet for UNIX and Linux

Description	Reference	Your Notes
Memory parameter settings for your operating system: ADDITIONAL_ANT_JAVA_TASK_ARGS ADDITIONAL_ANT_COMPILER_ TASK_ARGS	"Memory Parameter Settings" on page 39 For information about other system requirements, go to the IBM Support Portal at http://www.ibm.com/ support/entry/portal/Planning.	
Location of Installation Manager on your system	IBM Installation Manager information center:http://publib.boulder.ibm.com/ infocenter/install/v1r5/index.jsp	
Name and location of product repository and packages to be installed	IBM Installation Manager information center:http://publib.boulder.ibm.com/ infocenter/install/v1r5/index.jsp and the Sterling Selling and Fulfillment Foundation Support page: **http://www.ibm.com/support/entry/ portal/Planning**	
Location to use for shared resources directory, if default would cause issues with previously installed products or versions (see the Installation Manager documentation for more information)	IBM Installation Manager information center:http://publib.boulder.ibm.com/ infocenter/install/v1r5/index.jsp	
Path to and name of installation directory	"Installation Directory Information for UNIX and Linux" on page 59	
Do you want to install for a multiple schema environment?	*Sterling Selling and Fulfillment Foundation: Multi-Tenant Enterprise Guide*	
Are you doing an upgrade from a prior version of Sterling Selling and Fulfillment Foundation?	If you are doing an upgrade, stop and review the *Sterling Selling and Fulfillment Foundation: Upgrade Guide* before proceeding with the upgrade.	
Are going to manually or automatically apply database definition language (DDL) statements (schema) to the database.	"About Applying Database Definition Language (DDL) Statements After a Windows Installation" on page 41	
Location of JDK on your system		
Record the following information about your Oracle or DB2 database. This information may be case sensitive.		
Database user name and password (you should keep the password in a separate place for security purposes, instead of writing it here)		
Database catalog name	This is typically the database name; also known as SERVICE_NAME or SID in some versions of Oracle.	
Database host name (or IP address)		

Table 5. Installation Worksheet for UNIX and Linux (continued)

Description	Reference	Your Notes
Oracle: Absolute path and file name for the JDBC driver file.		
DB2: Absolute paths and file names for the JDBC driver file and license file.		
Database schema owner		
DB2 only: Do you want to use multibyte support?		
If you are planning a multischema deployment, determine and record the database information for four separate schemas: Metadata, Statistics, System Configuration, and Transaction/Master Data	*Sterling Selling and Fulfillment Foundation: Multi-Tenant Enterprise Guide*	
Are you planning to run a silent installation?	"About Recording Silent Installation Files in UNIX or Linux" on page 62	

Memory Parameter Settings

During an interactive installation, you are prompted for memory parameters for your system. These parameter values are written to the sandbox.cfg property file. After installation, you can edit the values as needed for performance and tuning.

The minimum memory requirements for Sterling Selling and Fulfillment Foundation are based on the products you are installing.

Table 6. Minimum Memory Parameter Settings by Operating System

Operating System	Parameters
Solaris 10 on SPARC processor (64-bit AMD)	-J-Xms256m -J-Xmx1408m -XX:MaxPermSize=768m
Windows 2008	-J-Xms1024m -J-Xmx2048m -XX:MaxPermSize=512m
IBM AIX 6.1 TL5 POWER System	-J-Xms1024m -J-Xmx1536m
IBM AIX 7.1 POWER System	-J-Xms1024m -J-Xmx1536m
HP-UX 11i v3 on Itanium	-J-Xms256m -J-Xmx1408m -XX:MaxPermSize=512m
Red Hat Enterprise Linux 6.0/AP 64-bit Xeon or AMD processor	-J-Xms1024m -J-Xmx1664m -XX:MaxPermSize=512m
SUSE Linux Enterprise 11 on 64-bit Xeon or AMD processor	-J-Xms1024m -J-Xmx1664m -XX:MaxPermSize=512m

Note: For information about how the memory parameters relate to the ADDITIONAL_ANT_JAVA_TASK_ARGS and ADDITIONAL_ANT_COMPILER_TASK_ARGS properties, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*

Installation Directory Information for UNIX and Linux

Before you install, verify that the system and directory where you plan to install Sterling Selling and Fulfillment Foundation meets the following criteria.

- It is not recommended to install Sterling Selling and Fulfillment Foundation as a root user because this would cause folders that are created by the installation to be inaccessible to anyone but the root user.
- The file system must have adequate free disk space.
- The name of the directory is case-sensitive.
- If you use the silent installation method, you cannot install into a pre-existing directory. The silent installation process will fail if a pre-existing directory is specified.
- If you use the GUI or text-based installation methods, you can install into either a pre-existing directory or a new directory to be created by the installation process.

Installation Methods on UNIX/Linux

Interactive Installation Using IBM Installation Manager

The IBM Installation Manager steps through the Sterling Selling and Fulfillment Foundation installation for a UNIX or Linux in one of three ways:

- A GUI-based installation wizard , on a UNIX or Linux system locally in an X Windows environment.
- Text-based installation wizard or remotely, in a text-based console environment.
- A silent installation file, which is written if you record a GUI or text-based installation.

Before performing any of these installation methods, you must install the IBM Installation Manager.

Setting Up X Windows for a GUI Installation

About this task

Using a GUI-supported operating system, perform the following actions:

Procedure

1. Use a connectivity client to connect to your UNIX/Linux account.
2. Set the display to use your X server as a client using the following command:

   ```
   export DISPLAY=<server>:0.0
   ```

   (or the appropriate Display identifier)

   In the above command, `:0.0` can be a different value, such as `:8.0`.

About the JDK for UNIX/Linux

- To determine which JDK version and patches you need, see the System Requirements on the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning.
- Edit the JAVA_HOME environment variable to point to the directory where you installed the JDK.
- After you install the JDK, record the absolute path to its location on your system. You must supply the absolute path when you install IBM Sterling Selling and Fulfillment Foundation.
- During the installation, you will be prompted whether to have the installer create a copy of the JDK under your install folder. If you do not want the JDK

copied, download version 2.7.1 of xerces and xalan from Apache prior to running the installation. Copy the following files into the `jre/lib/endorsed` directory of your JDK:

- serializer.jar
- xalan.jar
- xercesImpl.jar
- xml-apis.jar

Install the IBM Installation Manager

About this task

You must download and install the IBM Installation Manager before installing Sterling Selling and Fulfillment Suite.

Important:

You must install and launch Installation Manager as a normal user not as a root user. Before you begin installation, review the documentation for the IBM Installation Manager. The Installation Manager has multiple installation and deployment scenarios, so read the documentation carefully and decide which scenario fits your needs before proceeding. The Installation Manager information center is available here: http://publib.boulder.ibm.com/infocenter/install/v1r5/index.jsp.

Procedure

1. Determine the version of Installation Manager supported for Sterling Selling and Fulfillment Suite. For more information, go to the IBM Support Portal at http://www-947.ibm.com/support/entry/portal/Planning.
2. Go to the Installation Manager downloads page: http://www-947.ibm.com/support/entry/portal/All_download_links/Software/Rational/IBM_Installation_Manager
3. Locate and download the correct package for your operating system.
4. Extract the contents of the compressed file to a temporary directory.
5. Browse to the InstallerImage folder and run the installation command file for your operating system:

   ./userinst
6. Follow the prompts in the wizard to install the Installation Manager.

Add a Repository for Sterling Selling and Fulfillment Foundation in IBM Installation Manager

About this task

After Installation Manager is installed, you must add a repository that contains the files for the Sterling Selling and Fulfillment Foundation installation.

Procedure

1. Start Installation Manager.
2. Select File > Preferences and then select the Repositories tab from the left pane.
3. Click Add Repository.

4. Browse to the output folder that contains the files for Sterling Selling and Fulfillment Foundation.
5. Select the repository.config file and click OK.
6. Verify that the new repository is shown in the list of repositories and that its checkbox is selected.
7. Click Apply, then OK.

About Recording Silent Installation Files in UNIX or Linux

About this task

If you prefer, you can invoke the IBM Installation Manager from a command line so that it steps through the Sterling Selling and Fulfillment Foundation installation and records a silent installation file. You can modify and use this file to run the installation again.

Optionally, you can also record multiple silent installation files.

The topics "Running a GUI-Based Installation on UNIX or Linux" on page 64 and "Running the Text-Based Interactive Installation Program in UNIX or Linux" on page 66 explain how to record a silent file for installation. See "Running a Silent Installation in UNIX or Linux" on page 68 for information about running a silent installation file.

For information about recording and running multiple silent installations, see "Running Multiple Silent Installations on the Same Machine" on page 69.

Environment Specific Notes for UNIX and Linux

-
- On Linux, do not use any soft/symbolic links in the path to the jar file. Make sure that you specify the full path to the jar file.
- Create a UNIX Account

  In a UNIX or Linux environment, you must create a UNIX administrative account on the host server for each installation of Sterling Selling and Fulfillment Foundation. For example, if you want to create a test environment and a production environment, you need to create two UNIX accounts on the host server, one for the test and one for the production environment. For more information about creating UNIX accounts, see your operating system documentation.
- A root user cannot install Sterling Selling and Fulfillment Foundation.
- When creating a name, such as an account name, permissions name, profile name, or database name, follow these conventions:
  - Use any valid alphanumeric characters and _ (underscore).
  - Do not use spaces or apostrophes.
- JMS messages are by default encoded with UTF-8. This is controlled by the jms.message.encoding property in yfs.properties, not by the LANG setting in UNIX or Linux environments. Therefore, if using different encoding as part of the LANG setting for LINUX/UNIX machines, you must also override the jms.message.encoding property in yfs.properties to override the encoding of jms messages.

Before AIX Installations Only

About this task

During the installation process, you specify the name of the directory to be created for Sterling Selling and Fulfillment Foundation. The installation process creates the directory and uses it as the Home folder for the Sterling Selling and Fulfillment Foundation files and subdirectories. Throughout this book, this directory is referred to as <INSTALL_DIR>.

To ensure that <INSTALL_DIR> has the necessary permissions, AIX users must run the following command on the parent directory of <INSTALL_DIR> before installation:

```
chmod -R a-s <absolute path>/install_dir_parent
```

Here, `install_dir_parent` is the directory in which <INSTALL_DIR> will be created.

For example, to specify `AIX_1/applications/test1/my_install` as your installation directory, you could run the command from the `AIX_1/applications` directory (directly above the test1 directory):

```
chmod -R a-s test1
```

or from another location on the file system:

```
chmod -R a-s /AIX_1/applications/test1
```

This ensures that when the `my_install` directory is created during installation, it inherits the correct permissions from `test1`.

Before Linux Installations Only

About this task

This task applies only to application systems that use a Linux operating system.

Procedure

If the base locale for the system is English, make the following system change: set the LANG environment variable to `en_US`.

Before RedHat Enterprise Linux Installations Only

About this task

Perform this procedure for application systems that use the RedHat Enterprise Linux operating system only.

Make the following system changes:

Procedure

1. If the base locale for the system is English, edit the /etc/sysconfig/i18n file by changing the SUPPORTED variable from `en_US.utf8` to `en_US`. You can also allow multiple support using the following format:

   ```
   en_US.utf8:en_US
   ```

   Save and close the /etc/sysconfig/i18n file.
2. Edit the /etc/security/limits.conf file by adding the following lines:

```
*   hard   nofile   8196
*   soft   nofile   2048
*   hard   memlock   3000000
*   soft   memlock   4000000
*   hard   nproc   16000
*   soft   nproc   16000
*   hard   stack   512000
*   soft   stack   512000
```

This updates the system ulimits.

Save and close the /etc/security/limits.conf file.

3. Reboot the system.

Running an Installation in a UNIX or Linux Environment

Use the topics in this section to install Sterling Selling and Fulfillment Foundation

Running a GUI-Based Installation on UNIX or Linux

About this task

Use the following instructions to install in a UNIX or Linux environment from a command line, using a graphical user interface (GUI) in an X Windows client.

If you are upgrading from a previous release, stop any running instances of the previous installation.

Procedure

1. If you wish to record the installation to a silent file rather than perform an actual installation, issue the following command line. Otherwise, skip to Step 2.
   - Navigate to the **/IBM/Installation Manager/eclipse** folder.
   - Enter `./IBMIM -record response.xml -skipInstall <IM_data_dir>`

     ...where <*silent_filename*>.xml is the silent installation file to be generated and <*IM_data_dir*> is a new directory generated for the Installation Manager to write and store data into. For example:

     ```
     ./IBMIM -record response.xml -skipInstall C:/SterlingInstallData
     ```

     The IBM Installation Manager opens and begins the installation process. Skip to Step 3.
2. To perform an installation without recording, open the IBM Installation Manager by entering the following command line:
   - `./IBMIM`

   The IBM Installation Manager main menu is displayed.
3. Click Install.
4. Select the packages to be installed.
5. Review and accept the license agreement.
6. Accept the default location for the shared resources directory, or browse to a new location.
7. Accept the default location for the product to be installed, or enter a different location. If you specify an existing folder, it must be empty.
8. Select the features to install:
   - Product Files - selected by default.

- Enable Multi-Schema - select if doing an install with multiple database schemas
- Enable Upgrade - select if you are upgrading from a previous release **or** if you are doing a new installation, but want to apply DDL manually after installation. Do not select this option if you are doing a new multischema installation.

9. Enter the location and name of the JDK home directory. Copying the JDK to your installation directory is selected by default. If you do not want a copy of the JDK made under your installation directory, clear the checkbox.
10. Select your database, Oracle or DB2, from the list, then enter the information for the remaining database fields.
    a. If doing a single schema installation, enter information for the following fields:
       - Database user name
       - Database password/Confirm database password
       - Database catalog name
       - Database host name (or IP address)
       - Database port
       - Database schema name

         Note: The default schema names are always the same as the user name, except that the schema names are all uppercase. This applies to both single and multischema scenarios.
       - Click Add Jars to browse to the JDBC driver file for Oracle or DB2. Repeat to add the license file for DB2 (required).

       To test the connection to the database, click Validate Connection. If a message saying the connection failed is displayed, verify the information you entered in all fields and try validating again.
    b. If doing a multischema installation, click Add Jars to browse to the JDBC driver file for Oracle or DB2. Repeat to add the license file for DB2 (required). For each schema type (Metadata, Statistics, Configuration, Transaction), enter the database information:
       - Database user name
       - Database password/Confirm database password
       - Database catalog name
       - Database host name (or IP address)
       - Database port
       - Database schema name

         Note: The default schema names are always the same as the user name, except that the schema names are all uppercase. This applies to both single and multischema scenarios.

       For each schema, you can test the connection to the database by clicking Validate Connection. If a message saying the connection failed is displayed, verify the information you entered in all fields and try validating again.
11. On the Database Details screen:
    - If using DB2, choose whether to use multibyte support.
    - If using Oracle, there are no additional database properties to configure; click Next.

12. Enter the ADDITIONAL_ANT_JAVA_TASK_ARGS and ADDITIONAL_ANT_COMPILER_TASK_ARGS memory parameter settings for your operating system.
13. Click Install to start the installation. A progress bar is displayed at the bottom of the screen.
14. When the installation is complete, a status screen is displayed. Click Finish. You can also view the installation log from this screen.

    Note: If the installed_data directory is not removed by the post installation cleanup process, you can delete it. It is only used for installation.
15. Exit Installation Manager.
16. See "Post Installation Tasks for UNIX/Linux" on page 75 for configuration procedures that you must complete prior to running Sterling Selling and Fulfillment Foundation.

Running the Text-Based Interactive Installation Program in UNIX or Linux

About this task

Use the following instructions to install in a UNIX or Linux environment from a command line, using a text-based (non-GUI) interface.

If you are upgrading from a previous release, stop any running instances of the previous installation.

Procedure

1. Navigate to the **/IBM/Installation Manager/eclipse/tools** folder.
2. If you wish to record the installation to a silent file rather than perform an actual installation, issue the following command line. Otherwise, skip to Step 3.
   - Enter `./imcl -c -record response.xml -skipInstall <IM_data_dir>`

     ...where response.xml is the silent installation file to be generated and *<IM_data_dir>* is the directory generated for the Installation Manager to write and store data into. For example:

     ```
     ./imcl -c -record response.xml -skipInstall C:/SterlingInstallData
     ```

     The IBM Installation Manager opens and begins the installation process. Skip to Step 4.
3. To perform an installation without recording, enter the following command: `./imcl -c`
4. Type 1 to select Installation and press Enter.
5. Type 1 to select the IBM Sterling Selling and Fulfillment Suite 9.2.0 packages to install and press Enter.
6. Type 1 to choose version 9.2.0 for installation and press Enter.
7. Press Enter on the confirmation screen to move to the next screen.
8. Read each license agreement by typing the option number (1 or 2). After reading, type R to return to the prompt. When ready to accept the license agreements, type A and press Enter.
9. Choose a location for the Shared Resources directory. The default is IBM/SDPShared. To choose a different directory, type M and press Enter. Type the path and directory name and press Enter.
10. Press Enter on the confirmation screen to move to the next screen.

11. Choose an installation location. The default is the IBM directory. If you select a directory that already exists, an error message is displayed. To choose a different directory, type M and press Enter. Type the path and name of the new directory and press Enter.
12. Press Enter on the confirmation screen to move to the next screen.
13. Select the features to install:
    - 1. Product Files - selected by default.
    - 2. Enable Multi-Schema - select if doing an install with multiple database schemas
    - 3. Enable Upgrade - select if you are upgrading from a previous release **or** if you are doing a new installation, but want to apply DDL manually after installation. Do not select this option if you are doing a new multischema installation.

    To select Multi-Schema, type 2 and press Enter. To select Upgrade, type 3 and press Enter. After you have selected one option, the prompt is displayed again. You can either choose another option (2 or 3) or press Enter to accept the default (N) and continue to the next step.
14. Enter the path and name for the JDK home directory and press Enter.
15. Choose whether to have the installer create a copy of the JDK folders under your installation directory. The default is Yes (create the copy). Press Enter to accept the default and continue. If you do not want a copy of the JDK made, type N and press Enter.
16. Type the option number for the database, DB2 or Oracle, and press Enter.
17. Enter the information for the remaining database fields, depending on whether you chose single or multischema earlier:
    a. If doing a single schema installation, enter information for the following fields:
       - Database user name
       - Database password (note that the password is displayed in clear text)
       - Database catalog name
       - Database host name (or IP address)
       - Database port number
       - Database schema name

       Note: The default schema names are always the same as the user name, except that the schema names are all uppercase. This applies to both single and multischema scenarios.
       - Enter the location and name of the JDBC driver file for Oracle or DB2. Type A to add the license file for DB2 (required). After adding the license file and the prompt is displayed again, type D (done) to continue to the next step.

       To validate the database connection, type Y (validate the connection), or N (do not attempt to connect). If a message saying the connection failed is displayed, verify the information you entered in all fields and try validating again.
    b. If doing a multischema installation, enter the location and name of the JDBC driver file for Oracle or DB2. Type A to add the license file for DB2 (required). After adding the license file and the prompt is displayed again,

type D (done) to continue to the next step. For each schema type (Metadata, Statistics, Configuration, Transaction), enter the database information:

- Database user name
- Database password (note that the password is displayed in clear text)
- Database catalog name
- Database host name (or IP address)
- Database port number
- Database schema name

Note: The default schema names are always the same as the user name, except that the schema names are all uppercase. This applies to both single and multischema scenarios.

To validate the database connection for each schema, type Y (validate the connection), or N (do not attempt to connect). If a message saying the connection failed is displayed, verify the information you entered in all fields and try validating again. Once all information is entered for a schema, you can choose B (back) if you want to change information, or choose N (next) to continue to the next schema or step.

18. If using DB2, choose whether to enable database multibyte support (Y or N) and press Enter.
19. Enter the ADDITIONAL_ANT_JAVA_TASK_ARGS memory parameter settings for your operating system and press Enter..
20. Enter the ADDITIONAL_ANT_COMPILER_TASK_ARGS memory parameter settings for your operating system and press Enter. Press Enter again to confirm your choices.
21. Type G (Generate an Installation response file) to have an XML file that contains the install prompts and your responses created.
22. When prompted, type the location and name for the file and press Enter. Use "xml" as the file extension.
23. Once a message saying that the file was successfully created is displayed, press Enter. The previous screen is displayed again, with the same options. This time, type I (Install) to start the installation. A progress bar is displayed at the bottom of the screen.
24. When the installation is complete, a status screen is displayed.
25. See “Post Installation Tasks for UNIX/Linux” on page 75 for configuration procedures that you must complete prior to running Sterling Selling and Fulfillment Foundation, as well as optional tasks, and post installation tips.

Running a Silent Installation in UNIX or Linux

About this task

Use the following instructions to install in a UNIX or Linux environment from a command line, using the response.xml silent installation file. For information about recording a silent file, see “About Recording Silent Installation Files in UNIX or Linux” on page 62.

Procedure

1. Navigate to the **/IBM/Installation Manager/eclipse/tools** folder.
2. Enter `./imcl input response.xml -log <log_filename>log`.

...where response.xml is the silent installation file to be used for installation and <*log_filename*>.log is the log file to which installation logging data will be written. For example:

```
./imcl input response.xml -log C:/Installation.log
```

Note: See "Post Installation Tasks for UNIX/Linux" on page 75 for configuration procedures that you must complete prior to running Sterling Selling and Fulfillment Foundation.

Running Multiple Silent Installations on the Same Machine

About this task

You can perform multiple silent installations on the same machine by using the following steps.

Procedure

1. Complete an initial installation recording in GUI or text mode, as described in "Running a GUI-Based Installation on UNIX or Linux" on page 64 or "Running the Text-Based Interactive Installation Program in UNIX or Linux" on page 66.
2. Modify the silent installation file, `response.xml`, by editing the permissible values as needed. "UNIX/Linux Silent Install File Parameters" on page 70 for information about the parameters you can modify.
3. You can create multiple silent response files in either one of two ways:
   - **Choice 1:** You can save `response.xml` to create multiple variations of it, as long as they are named differently; for example, `response1.xml`, `response2.xml`, and so forth. You must also ensure that they have different profile IDs in them. Edit each `response.xml` so that it has a unique profile ID number, such as 1, 2, 3, and so forth. Edit both instances of the profile ID that are in the file. For example:

     ```
     <profile id='IBM Sterling Selling And Fulfillment Suite_1'
     installLocation=...>
     ```

     ```
     <offering id='...' version='...' profile='IBM Sterling Selling And
     Fulfillment Suite_1' features='...' installFixes='none'/>
     ```
   - **Choice 2:** If you prefer to create silent file variations by recording them to different `response<n>.xml` files by using the IBM Installation Manager multiple times instead of manually editing the profile ID, you will achieve the same outcome of having multiple silent installation files. This is because each time you record a silent installation, the IBM Installation Manager increments the profile ID number in the `response.xml` file so that each installation file is unique.

     To use this option, launch the IBM Installation Manager again and record this installation, specifying a new silent file; for example,`response3.xml`.

     For a GUI installation, from the **/IBM/Installation Manager/eclipse** folder, enter: `./IBMIM -record response3.xml -skipInstall <IM_data_dir>`

     For a text installation, from the **/IBM/Installation Manager/eclipse/tools** folder, enter: `./imcl -c -record response3.xml -skipInstall <IM_data_dir>`

     Note: <*IM_data_dir*> must be the same directory for all recordings.

     During the recording, the Installation Manager will see that Sterling Selling and Fulfillment Foundation has already been installed. Click `Continue` to indicate that you want to install it again. This appends a number to the existing profile ID.
4. Using either method, you can create multiple silent installation files.

For information about installing with a silent file, see "Running a Silent Installation in UNIX or Linux" on page 68

UNIX/Linux Silent Install File Parameters

Important: The following silent installation parameters are the only ones that you can change. If you change the values for parameters that are not listed in this table, the installation will fail.

Example Entry	Description
user.sbx.INSTALL_JAR	The JAR which contains the installation files. Example: `user.sbx.INSTALL_JAR value=SSFS_INSTALL.JAR`
user.sbx.DB_CLEAN	Valid values are true/false. If true, this property deletes all objects from the database. The installation sets this property to false.
user.sbx.JVM_LOC	Source of the downloaded JDK files, external to Sterling Selling and Fulfillment Foundation. Example: `user.sbx.JVM_LOC value=<JDK_DIR>`
user.sbx.OVERRIDE_LOAD_DEFAULTS_PK_GEN	To generate unique primary keys for entities across deployments, include this property in your silent installation file and set the property to true. Default is false. See the documentation about the Change Project Management feature for additional information. Example: `user.sbx.OVERRIDE_LOAD_DEFAULTS_PK_GEN value=true`
user.sbx.DB_VENDOR	(Required) The database vendor to use (Oracle, DB2). **Note:** In a multischema installation, you must deploy the same database vendor and version across deployments. Example: `user.sbx.DB_VENDOR value=<db_vendor>`
user.sbx.DB_USER	(Required) Database login ID with which to connect. In a multischema deployment, this must be the username for the Metadata schema. **Note:** The default schema names are always the same as the user name, except that the schema names are all uppercase. This applies to both single and multischema scenarios. Example: `user.sbx.DB_USER value=<db_user_name>`
user.sbx.DB_DATA	(Required) Database name to connect with. Example: `user.sbx.DB_DATA value=<db_data_catalog>`

Important: The following silent installation parameters are the only ones that you can change. If you change the values for parameters that are not listed in this table, the installation will fail.

Example Entry	Description
user.sbx.DB_PASS	(Required) Database password with which to connect. In a multischema deployment, this must be the password for the Metadata schema. Example: `user.sbx.DB_PASS value=<db_password>`
user.sbx.DB_HOST	(Required) Host for database (for example, server or IP address). Example: `user.sbx.DB_HOST value=<db_host>`
user.sbx.DB_PORT	(Required) Database listener port to which to connect. Example: `user.sbx.DB_PORT value=<db_listener_port>`

user.sbx.DB_DRIVERS	(Required) Full path to the JDBC driver file(s).
	• Oracle requires one file.
	• DB2 requires two files: the license file and the driver file.
	
	If specifying more than one file, use a pipe character (|) for a separator between files.
	
	Examples:
	
	`user.sbx.DB_DRIVERS value=<absolute_path_to_driver_jar>`
	
	`user.sbx.DB_DRIVERS value=<JDBC_driver_dir>/jdbc.jar`
	
	`user.sbx.DB_DRIVERS value=<JDBC_driver_dir>db2_1_jdbc.jar|<JDBC_driver_dir>/db2_2_jdbc.jar`

user.sbx.DB_DRIVERS_VERSION	(Required) Free form version string for JDBC driver. This is informational only. Example: `user.sbx.DB_DRIVERS_VERSION value=<db_driver_version>`
user.sbx.DB_SCHEMA_OWNER	(Required for multischema mode) Default schema/schema-owner for the provided login ID. If you wish to change this value to an alternate schema, consult your database administrator, as this is considered an expert installation scenario and can be performed only through the silent installation.

Important: The following silent installation parameters are the only ones that you can change. If you change the values for parameters that are not listed in this table, the installation will fail.

Example Entry	Description
user.sbx.LOAD_FACTORY_SETUP	Indicates whether you want to load factory setup defaults during installation (true) or manually after installation (false). If you are performing an installation in upgrade mode, set this property to false. Example: `user.sbx.LOAD_FACTORY_SETUP value=true` For information about manually loading the factory defaults, see "Loading the Sterling Selling and Fulfillment Foundation Database Factory Defaults After Installation" on page 94.
user.sbx.NO_DBVERIFY	Valid values are true or false. When set to true during installation and installservice, dbverify will not be run. This means that Sterling Selling and Fulfillment Foundation will not generate DDL to make the database like the XML entity repository. If you are performing an installation in upgrade mode, set this property to true. Example: `user.sbx.NO_DBVERIFY value=true`
user.sbx.REINIT_DB	Valid values are true or false. By default, the value is set to true. If the value is set to false, the Sterling Selling and Fulfillment Foundation installation will complete successfully, but no database operation will be performed as part of the installation process. If you are performing an installation in upgrade mode, set this value to false. Example: `user.sbx.REINIT_DB value=true`
user.sbx.multischema.applyddl	Valid values are true/false. If set to true (default), enables the DBVerify script to generate and apply database DDLs automatically. If set to false, allows DBVerify script to generate the DDLs but does not apply them. This property is set to false in the GUI installer, by default. If you are using the GUI Installer and do not want to apply DDLs, ensure that this property to set to false in the sandbox.cfg file. If this property is set to true or is absent in the sandbox.cfg file, the DBVerify script generates the DDLs and applies them. Otherwise, the scripts are generated and not applied. Example: `user.sbx.multischema.applyddl value=true`

Important: The following silent installation parameters are the only ones that you can change. If you change the values for parameters that are not listed in this table, the installation will fail.

Example Entry	Description
user.sbx.multischema.enabled	Valid values are true/false. If true, this attribute indicates that this is a multischema installation. The silent installation file, response.xml, specifies database information for the Configuration, Metadata, Transaction, and Statistics schemas. **Note:** This attribute is case-sensitive. Example: `user.sbx.multischema.enabled value=true` See "About Running a Silent Installation in Multischema Mode" on page 52 for more information.
user.sbx.multischema.version	(Required only if you enable multischema.) This attribute indicates which version is being installed. For the Sterling Selling and Fulfillment Foundation Release 9.2.0, you must enter 9.2.0. **Note:** This attribute is case-sensitive. Example: `user.sbx.multischema.version value=9.2.0`
user.sbx.multischema.file	(Required if you enable multischema.) This attribute indicates the name of the user-defined XML file that contains multischema database information. **Note:** This attribute is case-sensitive. Example: `user.sbx.multischema.file value=<`*`filename`*`>.xml`

user.sbx.ADDITIONAL_ANT_JAVA_TASK_ARGS
 	
user.sbx ADDITIONAL_ANT_COMPILER_TASK_ARGS
	Go to the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning for memory parameter values based on your operating system.
	
	Examples:
	
	`user.sbx.ADDITIONAL_ANT_JAVA_TASK_ARGS value= -XX:MaxPermSize=`*`nnn`*`m`
	
	`user.sbx ADDITIONAL_ANT_COMPILER_TASK_ARGS value=-J-Xms`*`nnn`*`m -J-Xmx`*`nnn`*`m`
	
	These parameter values are written to the sandbox.cfg file during installation. After installation, you can tune them if you are seeing Out-of-Memory errors.
	
	See the *Sterling Selling and Fulfillment Foundation: Properties Guide* for more information about these parameters.

user.sbx.COPYJVM	Valid values are true/false. By default, the installation creates a copy of the JDK under your installation folder.

Important: The following silent installation parameters are the only ones that you can change. If you change the values for parameters that are not listed in this table, the installation will fail.

Example Entry	Description
user.sbx.ENTITY_GEN_LOGLEVEL	Specify VERBOSE to enable logging of verbose messages during entity class generation. Default=INFO Example: `user.sbx.ENTITY_GEN_LOGLEVEL value=VERBOSE`
user.sbx.GENERATE_ALL_DBCLASSES	Specify true to generate all DBClasses. Default=false Example: `user.sbx.GENERATE_ALL_DBCLASSES value=true`
user.sbx.SUPPORT_MULTIBYTE	If you are installing on DB2 and need to localize your database using a multibyte character set, set this flag to Y. This ensures that the database column sizes are large enough to handle the multibyte characters correctly. Default is N. Example: `user.sbx.SUPPORT_MULTIBYTE=Y`
user.sbx.ORACLE_JDBC_URL	Required only if Oracle RAC is being used as the database. Include the complete URL without any blank spaces. Example: `user.sbx.ORACLE_JDBC_URL=<url>`

About Running a Silent Installation in Multischema Mode on UNIX or Linux

The response.xml file is written by the installation process when recording a multischema installation. The file specifies the database account information for each of the four allowable four data tables that you use: Metadata, Statistics, System Configuration, and Transaction/Master.

It also shows parameters for specifying multiple passwords and their effective dates so that you can predefine passwords for a given pool months in advance. These passwords will change automatically without a server restart.

In a multischema installation, you must deploy the same database vendor and version across deployments.

"UNIX/Linux Silent Install File Parameters" on page 70 explains the multischema parameters in response.xml.For information about recording a silent response.xml file during installation, see "Running a GUI-Based Installation on UNIX or Linux" on page 64 for GUI installations, or "Running the Text-Based Interactive Installation Program in UNIX or Linux" on page 66 for text-based installations.

Post Installation Tasks for UNIX/Linux

This topic contains configuration procedures that you must complete prior to running Sterling Selling and Fulfillment Foundation, as well as optional tasks and post installation tips.

Security

- See "Web Security Planning-Post installation Recommendations" on page 9 for information about the security measures that are recommended for your consideration.

Database

- Views must be created manually. Instructions to create views vary depending on what database is used. All database view related scripts are located at <INSTALL_DIR>/database/<db_type>/scripts. For more information, see the information about configuring your database type in "Database Configuration on UNIX/Linux-Overview" on page 21 and Chapter 13, "Configuring Utilities," on page 93.

Changing to Multischema Mode after Installation

- If you did not install or upgrade in multischema mode but you want to create the appropriate tables for multischema mode and update these tables at a later time, you can:
  - Set the following properties in the `sandbox.cfg` file:

    ```
    multischema.enabled=true
    multischema.version=9.2.0
    ```

  - Run the dbverify script on multischema colonies, as described in "Verifying the Database Schema" on page 96.

    If you want to create or add colonies to your deployment, see the *Sterling Selling and Fulfillment Foundation: Multi-Tenant Enterprise Guide* for instructions.

Temporary Directories Left by Installer

- After installation, you may see temporary directories similar to the following example:

  ```
  tmpSterlingInstall.12345678/
  ```

  You can delete these directories after the installation has completed.

Starting and Stopping the Information Center

Procedure

- The Sterling Selling and Fulfillment Suite information center has to be started before users can access product help topics. To start the information center, run the script `install_dir\bin\startHelp.cmd`.

  Note: Ensure that the application server (which hosts the application EAR) and Sterling Selling and Fulfillment Suite Information Center are not running on the same machine. If they are running on the same machine, access the application using the computer name (for example, http://abc.com:8001/sbc/sbc/login.do) and context-sensitive help using the IP address (for example, http://127.0.0.1:8002/help/index.jsp) or vice-versa. To change the IP address or Hostname for the context-sensitive help, edit the `EXT_HOST_ADDR` property in the `sandbox.cfg` file and run the setupfiles script.

- The information center has to be stopped prior to updating it, then restarted once updates are complete. To stop the information center, run the script `install_dir\bin\stopHelp.cmd`.

  Note: If an error occurs on running the stopHelp script, delete the workspace folder under `install_dir\xapidocs\ibm_help\eclipse`.

Web Security Planning-Post installation Recommendations

About this task

After the installation of Sterling Selling and Fulfillment Foundation, be sure to complete the following for ensured security:

Procedure

1. Change the password of the default user (admin).
2. Change permissions on *INSTALL_DIR*/bin/migrator.* files to non-executable.
3. API security is enabled during installation. After installation, you may want to reset the api.security.mode property and carefully consider your API security configuration. For more information about API security modes, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.
4. Ensure that the required permissions to access the API resources are defined for an application. For more information about configuring API security, see the *Sterling Selling and Fulfillment Foundation: Configuration Guide*.

Upgrade Information for UNIX/Linux

If installing in upgrade mode, use the topics in this section for reference.

Running a Silent Upgrade in UNIX or Linux

About this task

Important: If performing an upgrade from a prior version of Sterling Selling and Fulfillment Foundation, multischema mode is supported only if you are upgrading from a multischema environment. You cannot use the multischema option when upgrading from a single schema instance of Sterling Selling and Fulfillment Foundation. The same is true if you are upgrading with a silent installation file. Use the following instructions to start a silent upgrade in a UNIX or Linux environment. Before running the command, ensure that:

- You have thoroughly reviewed the *Sterling Selling and Fulfillment Foundation: Upgrade Guide* and completed all necessary tasks from it.
- You have recorded a silent upgrade file. See "About Recording Silent Installation Files in UNIX or Linux" on page 62.
- The silent upgrade file includes these settings: `user.sbx.LOAD_FACTORY_SETUP value=false`, `user.sbx.NO_DBVERIFY value=true`, and `user.sbx.REINIT_DB value=false`. "UNIX/Linux Silent Install File Parameters" on page 70 contains information about these silent file parameters.
- If the application is running, stop the instance before proceeding.

Note: If the `installed_data` directory is not removed by the post installation cleanup process, you can delete it. It is used only for installation.

To run the silent upgrade:

1. Navigate to the **/IBM/Installation Manager/eclipse/tools** folder.

2. Enter `./imcl input response.xml -log <log_filename>log`.

...where response.xml is the silent upgrade file to be used for upgrade and <*log_filename*>.log is the log file to which upgrade logging data will be written. For example:

```
./imcl input response.xml -log C:/Upgrade.log
```

Proceed with the silent upgrade.

The installation/upgrade dialogs are described in "Running a GUI-Based Installation on UNIX or Linux" on page 64 and "Running the Text-Based Interactive Installation Program in UNIX or Linux" on page 66.

Post Upgrade Tasks for UNIX/Linux

Creating Tables for Multischema Mode After Installation

If you did not install or upgrade in multischema mode but you want to create the appropriate tables for multischema mode and update these tables at a later time, you can:

- Set the following properties in the `sandbox.cfg` file:

  ```
  multischema.enabled=true
  multischema.version=9.2.0
  ```

- Run the dbverify script on multischema colonies, as described in "Verifying the Database Schema" on page 96.

  If you want to create or add colonies to your deployment, see the *Sterling Selling and Fulfillment Foundation: Multi-Tenant Enterprise Guide* for instructions.

Temporary Directories Left by Installer

After installation, you may see temporary directories similar to the following example:

```
tmpSterlingInstall.12345678/
```

You can delete these directories after the installation has completed.

Starting and Stopping the Information Center

Procedure

- The Sterling Selling and Fulfillment Suite information center has to be started before users can access product help topics. To start the information center, run the `install_dir/bin/startHelp.sh` script.

  Note: Ensure that the application server (which hosts the application EAR) and Sterling Selling and Fulfillment Suite Information Center are not running on the same machine. If they are running on the same machine, access the application using the computer name (for example, http://abc.com:8001/sbc/sbc/login.do) and context-sensitive help using the IP address (for example, http://127.0.0.1:8002/help/index.jsp) or vice-versa. To change the IP address or Hostname for the context-sensitive help, edit the `EXT_HOST_ADDR` property in the `sandbox.cfg` file and run the setupfiles script.

- The information center has to be stopped prior to updating it, then restarted once updates are complete. To stop the information center, run the `install_dir/bin/stopHelp.sh` script.

Note: If an error occurs on running the stopHelp script, delete the workspace folder under `install_dir/xapidocs/ibm_help/eclipse`.

Chapter 9. Installing the IBM Sterling Sensitive Data Capture Server

Installing the IBM Sterling Sensitive Data Capture Server

Before installing the IBM Sterling Sensitive Data Capture Server (SSDCS) application, you must read the *PA-DSS Implementation Guide* for information about how to configure IBM Sterling Sensitive Data Capture Server securely.

The IBM Sterling Sensitive Data Capture Server is an application that integrates with IBM Sterling Selling and Fulfillment Suite to ensure that credit card numbers and stored value card numbers are secure by tokenizing them. IBM Sterling Sensitive Data Capture Server is a system-critical application that must be installed, configured, and deployed before IBM Sterling Selling and Fulfillment Suite can capture payment information for credit cards and stored value cards.

The IBM Sterling Sensitive Data Capture Server application is packaged as a compressed file with Sterling Selling and Fulfillment Foundation. This compressed file is located in `<INSTALL_DIR>/repository/external/ssdcs.zip`.

For information about how to install, configure, extend, and deploy IBM Sterling Sensitive Data Capture Server, refer to the *Sterling Sensitive Data Capture Server: Configuration Guide*.

Chapter 10. Installing the Sterling Selling and Fulfillment Foundation Language Pack

Installing the Sterling Selling and Fulfillment Foundation Language Pack

Before installing the language pack, ensure that you have successfully installed the current release of Sterling Selling and Fulfillment Foundation.

Language packs are compressed files that are compatible with the UNIX, Linux, and Windows operating systems. To install the Sterling Selling and Fulfillment Foundation language pack, extract the contents of the compressed language pack file into your <INSTALL_DIR> directory.

Loading the Sterling Selling and Fulfillment Foundation Language Pack Factory Defaults-Overview

Prior to loading the Sterling Selling and Fulfillment Foundation Language Pack factory defaults, ensure that you have successfully completed all the instructions provided in Chapter 5, "Installing and Configuring Database Tier Software on UNIX or Linux," on page 21 or Chapter 6, "Installing and Configuring Database Tier Software on Windows," on page 31, as the case may be.

Note: The English language factory defaults must be loaded prior to loading the language-specific factory defaults.

Loading the Sterling Selling and Fulfillment Foundation Language Pack Factory Defaults for Windows

Procedure

1. To load the language-specific factory defaults, run the loadDefaults.cmd script for Windows that is available in the <INSTALL_DIR>\bin directory. Pass the locale-specific installer file and the directory path of the associated XML files.

   For example:

   ```
   loadDefaults.cmd <INSTALL_DIR> \repository\factorysetup\
   complete_installation\<language>_<country>_locale_installer.xml
   <INSTALL_DIR>\repository\factorysetup\complete_installation\XMLS
   ```

   For more information about the configuration steps involved in loading the factory defaults, see "Loading the Sterling Selling and Fulfillment Foundation Database Factory Defaults After Installation" on page 94.
2. Create the resource JAR using the following command:

   For Windows®: <INSTALL_DIR>/bin/deployer.cmd -t resourcejar

   For Unix: <INSTALL_DIR>/bin/deployer.sh -t resourcejar

Loading the Sterling Selling and Fulfillment Foundation Language Pack Factory Defaults for UNIX/Linux

Procedure

1. To load the language-specific factory defaults, run the `loadDefaults.sh` script for UNIX and Linux that is available in the `<INSTALL_DIR>/bin` directory. Pass the locale-specific installer file and the directory path of the associated XML files.

   For example:

   ```
   loadDefaults.sh <INSTALL_DIR> /repository/factorysetup/complete_installation
   /<language>_<country>_locale_installer.xml <INSTALL_DIR>/repository/
   factorysetup/complete_installation/XMLS
   ```

   For more information about the configuration steps involved in loading the factory defaults, see "Loading the Sterling Selling and Fulfillment Foundation Database Factory Defaults After Installation" on page 94.

2. Create the resource JAR using the following command:

   For Windows®: `<INSTALL_DIR>/bin/deployer.cmd -t resourcejar`

   For Unix: `<INSTALL_DIR>/bin/deployer.sh -t resourcejar`

Loading the Language Pack Translations in Windows

About this task

Prior to loading the Sterling Selling and Fulfillment Foundation Language Pack factory defaults, ensure that you have successfully installed and configured your database tier software.

Important: Verify that your locale settings, such as currency, time format, date, and so on are correct.

Procedure

1. To load the language pack translation with custom localization literals, run the Localized String Reconciler tool in the IMPORT mode from the `<INSTALL_DIR>\bin` directory as follows:

   ```
   sci_ant.cmd -f localizedstringreconciler.xml import -Dsrc=<INSTALL_DIR>
   \repository\factorysetup\complete_installation\XMLS -Dbasefilename=
   ycplocalizedstrings -Dvariablefilename=resources\ycd_fc_variable.properties
   ```

   where `<INSTALL_DIR>` refers to the installation directory.

   The `basefilename` refers to the file present in the following directory, for which the translations are to be imported into the database.

   ```
   <INSTALL_DIR>\repository\factorysetup\complete_installation\XMLS
   ```

   The default value for the `basefilename` parameter is `ycplocalizedstrings`.

   For example, to import translations for the Sterling Call Center or Sterling Store language pack, the base file is `ycdlocalizedstrings`. The base file is derived from the `xx_XX_ycdlocalizedstrings_yy_YY.properties` file. This file inserts the values specified in the properties file into the database.

   The Localized String Reconciler tool inserts the values specified in the following file...

   ```
   <from_language>_<from_country>_<basefilename>_<to_language>_<to_country>.properties
   ```

   ... from the following directory...

   ```
   <INSTALL_DIR>\repository\factorysetup\complete_installation\XMLS\<language>_<country>
   ```

... into the database.

2. If you need to localize the factory setup of add-in, run the following command:

   ```
   sci_ant.cmd -f localizedstringreconciler.xml import -Dsrc=<INSTALL_DIR>
   \repository\factorysetup\complete_installation\XMLS -Dbasefilename=
   ycdfcaddinliterals2translate -Dvariablefilename=resources
   \ycd_fc_variable.properties
   ```

   To localize Sterling Business Center, enter the following:

   ```
   sci_ant.cmd -f localizedstringreconciler.xml import -Dsrc=<INSTALL_DIR>
   \repository\factorysetup\complete_installation\XMLS -Dbasefilename=
   sbcliterals2translate -Dvariablefilename=resources
   \ycd_fc_variable.properties
   ```

Loading the Language Pack Translations in UNIX/Linux

About this task

Prior to loading the Sterling Selling and Fulfillment Foundation Language Pack factory defaults, ensure that you have successfully configured your database tier software.

Important: Verify that your locale settings, such as currency, time format, date, and so on are correct.

Procedure

1. To load the language pack translation with custom localization literals, run the Localized String Reconciler tool in the IMPORT mode from the <INSTALL_DIR>/bin directory as follows:

   ```
   sci_ant.sh -f localizedstringreconciler.xml import -Dsrc=<INSTALL_DIR>
   /repository/factorysetup/complete_installation/XMLS -Dbasefilename=
   ycplocalizedstrings -Dvariablefilename=resources/ycd_fc_variable.properties
   ```

   where <INSTALL_DIR> refers to the installation directory.

   The basefilename refers to the file present in the following directory, for which translations are to be imported into the database:

   ```
   <INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS
   ```

   The default value for the basefilename parameter is ycplocalizedstrings.

   For example, to import translations for the Sterling Call Center or Sterling Store language pack, the base file is ycdlocalizedstrings. The base file is derived from the xx_XX_ycdlocalizedstrings_yy_YY.properties file. This file inserts the values specified in the properties file into the database.

   The Localized String Reconciler tool inserts the values specified in the following file...

   ```
   <from_language>_<from_country>_<basefilename>_<to_language>_<to_country>.properties
   ```

   ... from the following directory...

   ```
   <INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS/<language>_<country>
   ```

   ...into the database.

2. If you need to localize the factory setup of add-in, run the following command:

   ```
   sci_ant.cmd -f localizedstringreconciler.xml import -Dsrc=<INSTALL_DIR>
   /repository/factorysetup/complete_installation/XMLS -Dbasefilename=
   ycdfcaddinliterals2translate -Dvariablefilename=resources
   /ycd_fc_variable.properties
   ```

   To localize Sterling Business Center, enter the following:

```
sci_ant.cmd -f localizedstringreconciler.xml import -Dsrc=<INSTALL_DIR>
/repository/factorysetup/complete_installation/XMLS -Dbasefilename=
sbcliterals2translate -Dvariablefilename=resources
/ycd_fc_variable.properties
```

Switching the Sterling Selling and Fulfillment Foundation Base Language

The base language for the Applications Manager can be switched only once. For more information about switching the base language and performing the switch test, see the *Sterling Selling and Fulfillment Foundation: Localization Guide*.

Note: After switching the base locale from en_US to any other locale (for example, fr_FR), CatalogSearchConfigProperties.xml should be extended to include the switched locale code (which is fr_FR) in order to build the search index successfully. For more information about extending catalog search, see *Sterling Selling and Fulfillment Foundation: Extending the Database*. Also see the *Sterling Selling and Fulfillment Foundation: Catalog Management Concepts Guide* for more general information about catalog searches.

Note: If you switch the base locale from en_US to any other locale (for example, fr_FR):

- CatalogSearchConfigProperties.xml should be extended to include the switched locale code (which is fr_FR) in order to build the search index successfully. For more information about extending catalog search, see *Sterling Selling and Fulfillment Foundation: Extending the Database*. Also see the *Sterling Selling and Fulfillment Foundation: Catalog Management Concepts Guide* for more general information about catalog searches.
- If the yfs.install.localecode property is set to the new base locale, you must append en_US to all out-of-the-box English property files if you want the UI to display for the English user's locale. To determine the names of the out-of-the-box property files that you need to update with en_US, refer to all files in your language pack. These files have identical names to the files that you need to update, except they have your specific locale (for example, fr_FR) appended to each file name.

Creating and Deploying the Enterprise Archive

If you are installing both Sterling Selling and Fulfillment Foundation and the language pack together, it is sufficient if you create and deploy the EAR once. If you have already deployed your application and are installing the language pack after this, you need to re-create and redeploy the EAR file.

For more information about creating and deploying the EAR file for your chosen application server, see Chapter 14, "Deploying Sterling Selling and Fulfillment Foundation," on page 109.

Configuring Locales in UNIX/Linux

About this task

Sterling Selling and Fulfillment Foundation runs on any locale that Java supports. If you want to run Sterling Selling and Fulfillment Foundation on a non-default locale, configure your environment to the specific locale that you want to use.

Note: To configure your operating system as a non-English environment, refer to your operating system's documentation.

To determine and set the locale in a UNIX or Linux environment:

Procedure

1. Enter locale -a. A list of locales is displayed.
2. Set your locate by entering:
   - export LANG=<*locale*>
   - export LC_ALL=<*locale*>

   Example to set the locale to Japanese (on Solaris):
   - export LANG=ja_JP
   - export LC_ALL=ja_JP

   Note: Some UNIX shells require the `setenv` command instead of the `export` command.

   To determine and set your locale in a Windows environment:
3. Select Control Panel > Regional Options > General tab.
4. From the Your locale (location) list, select the language and location.
5. Click Set Default and select the locale from the Set the appropriate locale list.

Configuring Locales in Windows

About this task

Sterling Selling and Fulfillment Foundation runs on any locale that Java supports. If you want to run Sterling Selling and Fulfillment Foundation on a non-default locale, configure your environment to the specific locale that you want to use.

Note: To configure your operating system as a non-English environment, refer to your operating system's documentation.

To determine and set your locale in a Windows environment:

Procedure

1. Select Control Panel > Regional Options > General tab.
2. From the Your locale (location) list, select the language and location.
3. Click Set Default and select the locale from the Set the appropriate locale list.

About Localized Content in the Information Center

When you install a language pack, any topics that have a localized version are displayed in the specified language. If no localized version of a topic exists, the topic is displayed in English.

The language setting that is used depends on how you access the help:

- If you access the help from within the application, localized topics are displayed in the language identified by the current user's locale.
- If you access the help from outside of the application (for example, by typing the URL for the help system into a browser), localized topics are displayed in the language identified by the browser setting.

Chapter 11. Using a Local Version of the Information Center

About this task

By default, the Sterling Selling and Fulfillment Foundation installs a small number of help topics with the product and accesses most of the product documentation on an information center residing on the ibm.com® Web site. Users who do not have access to the Internet can display only the limited number of installed help topics and cannot view the bulk of the documentation that resides on the Web site. To address this problem, you can optionally download the information center to your local environment so that users can access it.

If you download the information center, you can update it when changes are available.

Note: Sterling Configurator Visual Modeler does not support access to an information center that resides in your local environment. Users of Sterling Configurator Visual Modeler must have access to the Internet to display help from within the application.

Procedure

1. Update the host address for the information center, so that it points to your local server instead of the IBM remote location.
   a. On the server where Sterling Selling and Fulfillment Foundation is installed, browse to the `install_dir/properties/` folder. Open the `customer_overrides.properties` file in a text editor. If the customer_overrides.properties file does not exit, create it.
   b. Add this line to the `customer_overrides.properties` file: `yfs.yfs.online.help.url=http://<Hostname/IPaddress>:9999/help/index.jsp`. Here, *<Hostname/IPaddress>* refers to the Hostname or IPaddress of the local server that is used for hosting the Sterling Selling and Fulfillment Suite Information Center.

      Example: `yfs.yfs.online.help.url=http://10.23.23.25:9999/help/index.jsp`

      Note: Ensure that the application server (which hosts the application EAR) and Sterling Selling and Fulfillment Suite Information Center are not running on the same machine. If they are running on the same machine, access the application using the computer name (for example, http://abc.com:8001/sbc/sbc/login.do) and general help using the IP address (for example, http://127.0.0.1:9999/help/index.jsp) or vice-versa.
   c. Rebuild the product ear file and redeploy it to the application server.
2. Download, extract, and set up the information center on your local server where Sterling Selling and Fulfillment Foundation is installed.
   a. Download the latest version of the information center compressed file (SSFSInformationCenter9.2.0.x.zip) from this FTP site:

      ftp://public.dhe.ibm.com/software/commerce/doc/ssfs/92/
   b. On the server where Sterling Selling and Fulfillment Foundation is installed, extract the compressed file to this folder:

      `install_dir/xapidocs/ibm_help/eclipse/plugins`

c. Stop the information center system by running the script `install_dir/bin/stopHelp.sh` (for UNIX) or `install_dir\bin\stopHelp.cmd` (for Windows).

d. Delete the `install_dir/xapidocs/ibm_help/eclipse/workspace` directory.

e. Delete the subdirectories and files in the `install_dir/xapidocs/ibm_help/eclipse/configuration` directory except for the `config.ini` file. Do not delete the `config.ini` file.

f. Restart the information center by running the script `install_dir/bin/startHelp.sh` (for UNIX) or `install_dir\bin\startHelp.cmd` (for Windows).

g. Users can now access the context-sensitive help and general help without being connected to the Internet.

Updating a Local Version of the Information Center

About this task

The Sterling Selling and Fulfillment Suite information center content is updated periodically. If you have deployed the information center on a server in your local environment, you can download the updates and deploy them.

Procedure

1. Download the latest version of the information center compressed file (SSFSInformationCenter9.2.0.x.zip) from this FTP site:

   ftp://public.dhe.ibm.com/software/commerce/doc/ssfs/92/
2. On the server where Sterling Selling and Fulfillment Foundation is installed, extract the compressed file to the `install_dir/xapidocs/ibm_help/eclipse/plugins` folder.
3. Stop the information center system by running the script `install_dir/bin/stopHelp.sh` (for UNIX) or `install_dir\bin\stopHelp.cmd` (for Windows).
4. Delete the `install_dir/xapidocs/ibm_help/eclipse/workspace` directory.
5. Delete the subdirectories and files in the `install_dir/xapidocs/ibm_help/eclipse/configuration` directory except for the `config.ini` file. Do not delete the `config.ini` file.
6. Restart the information center by running the script `install_dir/bin/startHelp.sh` (for UNIX) or `install_dir\bin\startHelp.cmd` (for Windows).
7. Users can now access the context-sensitive help and general help without being connected to the Internet.

Changing the Port Number Used for Context-Sensitive Help Topics

About this task

When Sterling Selling and Fulfillment Suite is installed, a small number of context-sensitive help topics are installed with it. To enable users to access these local help files without being connected to the Internet, a copy of the information center application is also installed with the product. This local information center uses the port number 9999 by default. If this port is already in use or reserved in your environment, you must change the port number for the local information center in the sandbox.cfg file.

Procedure

1. On the server where Sterling Selling and Fulfillment Foundation is installed, browse to the *install_dir*/properties/ folder. Open the sandbox.cfg file in a text editor.
2. Locate the IC_PORT parameter. Change the port from the default, 9999, to the desired port number.
3. Run the script *install_dir*\bin\setupfiles.cmd (for Windows) or *install_dir*/bin/setupfiles.sh (for UNIX).
4. Rebuild the product EAR file and redeploy it to the application server.

Chapter 12. Installing a Print Server

Installing a Print Server - Overview

This section explains how to install and configure the Loftware Label Manager (LLM) and Loftware Print Server (LPS).

For more information about configuring the Loftware Label Manager and Print Server, see the *Loftware Label Manager User's Guide* and the *Loftware Print Server User's Guide*, available from Loftware, Inc.

For more information about Performance Considerations for setting up the Loftware Print Server (LPS) see the *Loftware Print Server User's Guide*, available from Loftware, Inc.

Note: Loftware Label Manager and Loftware Print Server are third party software products that can be purchased directly from the vendor, Loftware, Inc.

Installing Loftware Components

The Loftware Print Server manages bar code label print requests between applications and hundreds of networked printers. As a general guideline, you should configure a maximum of 200 printers for each Loftware Print Server you install. For more information about server requirements and installation instructions, see the *Loftware Print Server User's Guide*. Contact your Loftware support representative for additional sizing and configuration support.

The Loftware Label Manager, used for designing labels, may be installed on any compatible PC. For more information about server requirements and installation guidelines, see the *Loftware Label Manager User's Guide*.

Sterling Selling and Fulfillment Foundation supports printing in the following modes:

- File Copy Mode
- TCP/IP Sockets Mode

  The `yfs.loftware.tcpip.sockets` attribute in the `yfs.properties.in` file determines the mode used for printing. By default this boolean property is set to 'N' for File Copy Mode.

  To configure the Loftware printing in the TCP/IP Sockets Mode, use the `<INSTALL_DIR>/properties/customer_overrides.properties` file to set the `yfs.loftware.tcpip.sockets` property to `Y`. For more information about overriding properties using the `customer_overrides.properties` file, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.

Sterling Selling and Fulfillment Foundation requires the following settings in the Loftware Print Server Configuration Utility:

- In Directory Set up, ensure that the 'Pass Files' option is selected.
- When using File Copy Mode: In Directory Set up, ensure that the 'Enable Polling (Disable Event File Trigger)' option is selected. It is recommended that the Poll Interval value is set to 500 Milliseconds.

Note: In File Copy Mode, SAMBA should be configured when using a UNIX version of the application server.

Note: The Drop Directories of the printers configured in Loftware need to be mounted on to the UNIX server using SAMBA.

Defining Printers on Loftware

Configure printers on Loftware using the Loftware Design 32 tool. For more information about configuring printers using the Loftware Design 32 tool, see the *Loftware Label Manager User's Guide*.

Defining Printers for the Sterling Store Inventory Management Installation

For more information about configuring printers for Sterling Store Inventory Management, see the *Sterling Store Inventory Management: Implementation Guide*.

Installing JasperReports

JasperReports is a third-party, open source Java reporting tool that delivers rich content on the screen, to the printer or in the format of a PDF, HTML etc,. You can use JasperReports with Sterling Selling and Fulfillment Foundation for printing or generating PDF objects for order reports, labels and so forth. The installation procedure and sample files are located in `<INSTALL_DIR>/xapidocs/code_examples/jasperreports` directory.

Note: For more information about JasperReports and supporting jars and components, go to the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning and the `alert_report_readme.html` file, located in the `<INSTALL_DIR>/xapidocs/code_examples/jasperreports` directory.

Chapter 13. Configuring Utilities

Configuring Sterling Selling and Fulfillment Foundation Utilities - Overview

Sterling Selling and Fulfillment Foundation provides script files (.sh for UNIX and .cmd for Windows) that you must customize after installation.

This section describes all the utilities supplied by Sterling Selling and Fulfillment Foundation, organized in the order in which you are likely to use them. It describes generic customizations that apply to most or all utilities. Further details specific to each utility are provided throughout the rest of this guide.

Installation Utilities

Installation utilities enable you to install Sterling Selling and Fulfillment Foundation. These utilities are present in the `<INSTALL_DIR>/bin` directory. Some of the utilities used for installing the various configurations of Sterling Selling and Fulfillment Foundation are "loadFactoryDefaults" and "dbverify".

loadFactoryDefaults

This utility loads the standard installation database configuration, known as the "factory defaults".

dbverify

This utility performs database schema creation, verification, and correction. Dbverify is used to ensure database schema integrity. When run, it invokes a Java class to compare a database with the entity XMLs and generates the SQL statements that would make the database match the entity repository; it generates SQL statements for any differences between the two.

During the installation process, dbverify is used to generate SQL scripts to create the database schema or tables and indexes. These SQL statements are then run against the database, unless you choose to manually create database schemas after installation.

install3rdparty

This utility copies supplied resources into the Sterling Selling and Fulfillment Foundation directory structure, and can append added jar files to the global classpath, agent classpath, or application server EAR file.

installService

This utility installs programs for specific tasks, like a regression test jar file or a fix pack jar file.

setupfiles

This utility checks the various initial product settings files (`*.in`) files for variables and updates the corresponding files with the values defined in the `sandbox.cfg` file

to create the final files that are used by the product during runtime.

Creating Database Schemas and Loading Factory Defaults After Installation

By default, the database schemas are created and factory defaults are automatically loaded during installation. However, you can tell the installation process to skip these tasks, then perform the tasks manually after installation. To have the installation process skip creating the database schemas and loading factory defaults, do one of the following, depending on the type of installation you choose:

- If you are using the GUI Installer on UNIX/Linux or on Windows, check the Upgrade feature on the Installation Features screen.
- If you are using the text-based installation process on UNIX/Linux, type the option number for Upgrade on the Installation Features screen.
- If you are using the silent installation method, set the LOAD_FACTORY_SETUP parameter in your silent installation file to false and the NO_DBVERIFY parameter in your silent installation file to True prior to running the installation.

Updating Properties Files After Installation

Procedure

After installing Sterling Selling and Fulfillment Foundation in Upgrade mode, reset the following properties in `<INSTALL_DIR>/properties/sandbox.cfg` as shown here:

- REINIT_DB=true
- LOAD_FACTORY_SETUP=true
- NO_DBVERIFY=false
- DB_SCHEMA_OWNER=<YOUR_DATABASE_SCHEMA_OWNER> (entry is required to be all upper-case)

  After setting the properties, you must re-run `setupfiles.sh/cmd` from the `<INSTALL_DIR>/bin` folder.

  For more information about editing the sandbox.cfg file, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.

Enabling the Oracle Database Text Search Feature - Overview

Sterling Selling and Fulfillment Foundation supports two types of text search indexes on Oracle databases: CTXCAT and CONTEXT. The CTXCAT index supports automatic updating of text search indexes, whereas, the CONTEXT index does not support automatic updating of text search indexes. IBM recommends that you use the CTXCAT index.

For information on how to create the text search indexes, refer to the *Sterling Selling and Fulfillment Foundation: Extending the Database*.

Loading the Sterling Selling and Fulfillment Foundation Database Factory Defaults After Installation

About this task

To load the Sterling Selling and Fulfillment Foundation database factory defaults after the product installation, load the defaults using the script applicable to your operating system.

Procedure

From the command line, run the <INSTALL_DIR>/bin/loadFactoryDefaults.sh command on UNIX and Linux or the <INSTALL_DIR>\bin\loadFactoryDefaults.cmd command on Windows.

Note: If the factory default installation stops before it is finished, each package under <INSTALL_DIR>/repository/factorysetup contains a file named "installer.xml.restart". This file records the location where the installation was stopped, and it is used the next time the factory defaults are installed.

Results

You can also generate audits when running loadFactoryDefaults script by overriding the value of the AUDIT_LOAD_DEFAULTS property and setting it to true. By default, this property is set to false. To override the value of this property, add an entry for it in the sandbox.cfg file. For more information about modifying properties and sandbox.cfg file, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.

Enabling the Text Search Feature for CONTEXT Index

About this task

The CONTEXT index does not automatically update text search indexes. Therefore, the DBA has to manually update text search indexes by running the EFrame_TextIndexUpdates.sql script.

To enable the text search feature on Oracle database using the CONTEXT index:

Procedure

1. Make sure that the Oracle database is configured with the Oracle Text feature.
2. Log in to the Oracle server with a user ID having the CTXAPP privilege.

   Note: The CONTEXT type text search indexes that are created on Oracle database using the EFrame_TextIndexAdds.sql script are not updated automatically. The DBA has to run the EFrame_TextIndexUpdates.sql script to update the CONTEXT type text search indexes whenever required using scheduled jobs. The frequency of these scheduled jobs can be decided by the DBA.
3. Verify that the text search index creation was successful.
4. Edit the customer_overrides.properties file that is located in the <INSTALL_DIR>/properties/ directory to add the following entries:

   ```
   yfs.yfs.db.textsearch=Y
   yfs.yfs.db.textsearch.oracle.contexttype=context
   ```

   For additional information about overriding properties using the customer_overrides.properties file, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.

Enabling the Text Search Feature for CTXCAT Index

About this task

The CTXCAT index automatically updates text search indexes. Therefore, the DBA need not manually run the EFrame_TextIndexUpdates.sql script to update text search indexes.

To enable the text search feature on an Oracle database using the CTXCAT index:

Procedure

1. Make sure that the Oracle database is configured with the Oracle Text feature.
2. Log in to the Oracle server with a user ID having the CTXAPP privilege.
3. Verify that the text search index creation was successful.
4. Edit the customer_overrides.properties file that is located in the <INSTALL_DIR>/properties directory to add the following entries:

   ```
   yfs.yfs.db.textsearch=Y
   yfs.yfs.db.textsearch.oracle.contexttype=ctxcat

   For additional information about overriding properties using the
   customer_overrides.properties file, see the
   Sterling Selling and Fulfillment Foundation:
   Properties Guide.
   ```

Verifying the Database Schema

About this task

You can run the dbverify utility to verify the database schema as follows:

Procedure

1. (Oracle only) If you are using the Oracle database, add an ORA_TS_CONTEXT entry to the <INSTALL_DIR>/properties/sandbox.cfg file. This entry determines the text search index type for Oracle.

   Assign one of the following values to ORA_TS_CONTEXT:

   - CONTEXT

     For fast retrieval of unstructured text.
   - CTXCAT (default)

     For retrieval of structured text like numbers and dates.
2. (Oracle only) If you are using the Oracle database, add an ORACLE_NLS_LENGTH_SEMANTICS entry to the sandbox.cfg file in the properties subdirectory of your installation directory. This entry determines the type of length semantic to be used for Oracle database, when using the dbverify tool.

   Assign one of the following values to ORACLE_NLS_LENGTH_SEMANTICS:

   - CHAR
   - BYTE (default)

   Note: If the database or the specific session in which database was created has length semantic as CHAR, this property must be set to CHAR before running the dbverify tool.
3. You may want to prevent dbverify from generating "IndexAdds" SQLs for indexes that have been dropped from the database; for example, you dropped an index in an earlier release and are now using dbverify to make a correction to the database. To prevent dbverify from generating "IndexAdds" SQLs for indexes that have been dropped from the database, use a text editor to create the indexes_not_created.txt file and place the file in the <INSTALL_DIR>/extensions/schemagenerator/ directory. This is the default name and location of the index-creation-suppression file. When creating the file, ensure that you specify dropped indexes on separate lines of the file. If you want to use a different name and location for the file, provide the INDEXES_NOT_CREATED

property in the <INSTALL_DIR>/properties/sandbox.cfg file and set the property's value to the full path where you want the index-creation-suppression file to be read from.

4. If you have a single-schema deployment, skip this step and go to Step 4.

   If you have a multischema deployment and are installing in upgrade mode, run the dbverify script from the <INSTALL_DIR>/bin/ folder and pass the multischema.xml file, which specifies database information and sets up multischema table types:

   ```
   dbverify.sh -colonyxml <INSTALL_DIR>/multischema.xml (on UNIX
   and Linux)
   ```

   or

   ```
   dbverify.cmd -colonyxml <INSTALL_DIR>/multischema.xml (on
   Windows)
   ```

   If you run the dbverify command without passing the multischema.xml file, it runs dbverify for all colonies.

   Note: By default, the multischema.applyddl property is set to true, which enables the dbverify script to run the associated scripts automatically. If you set it to false, the dbverify script generates DDLs but does not apply them. In this case, you must run these scripts manually.

   To run dbverify on a specific colony, pass the Colony ID in the command line as follows:

   ```
   dbverify.sh -ColonyId <Colony_Id> (on UNIX and Linux)
   ```

   or

   ```
   dbverify.cmd -ColonyId <Colony_Id> (on Windows)
   ```

5. If you are installing in single-schema mode, run the dbverify script from the <INSTALL_DIR>/bin/ folder as follows:

   ```
   dbverify.sh (on UNIX and Linux)
   ```

   or

   ```
   dbverify.cmd (on Windows)
   ```

6. If you have enabled the text search feature and change the text search index type in Oracle from ctxcat to context or vice-versa, the updated create and drop SQL scripts can be found in the <INSTALL_DIR>/bin/EFrame_TextIndexUpdates.sql file.

   In multischema deployments, the updated create and drop SQL scripts can be found in the <INSTALL_DIR>/bin/EFrame_<*Pool_Id*>_<*TableType*>.sql file, where:

   - <*Pool_Id*> to the name of the jdbc pool for which the sql scripts should be run as specified in the jdbc.properties or multischema.xml file.
   - <*FileType*> refers to one of a set of hardcoded names that are used to contains different types of sql statements. Possible values include:
     - UpdateQueries
     - TableChanges
     - Sequence
     - IndexDrops
     - IndexAdds
     - TextIndexDrops
     - TextIndexAdds
     - TextIndexUpdates
     - TextIndexModify

For example, the script for table alterations for the metadata pool for an Oracle install would be called: EFrame_oraclePool_TableChanges.sql

7. The differences between the entity XMLs and the database are generated in the form of SQL scripts, which can be run against the database to rectify the differences.

   For example, if there is a mismatch in the size of a datatype for a column [varchar2(20) to varchar2(40)] that has an associated index, dbverify generates SQL statements for:

   - Dropping the Index
   - Changing the size of the datatype for the column
   - Creating the new Index

   The three SQL statements described in the previous list appear in different *.sql files. The appropriate *.sql files must be run in the proper order as follows:

   a. Run the <INSTALL_DIR>/bin/EFrame_IndexDrops.sql for dropping the index.
   b. Run the <INSTALL_DIR>/bin/EFrame_TableChanges.sql for altering the size of the datatype for a column.
   c. Run the <INSTALL_DIR>/bin/EFrame_IndexAdds.sql for creating a new index.

   If the SQL statements are not run in the sequence as mentioned above, it results in script failure.

   The scripts shown in Table 7 are generated.

   Note: All scripts listed below can be found in the <INSTALL_DIR>/bin directory.

Table 7. Generated Scripts

Single Schema Script Name	Multischema Script Name	Description of the script
EFrame_Sequence.sql	EFrame_<Pool_Id>_<TableType>_Sequence.sql	Contains all the additional sequences that need to be created.
EFrame_TableChanges.sql	EFrame_<Pool_Id>_<TableType>_TableChanges.sql	Contains all the table column differences that need to be applied on the database schema. Modify this file to reference your tablespaces.
EFrame_Drops.lst	EFrame_<Pool_Id>_<TableType>_Drops.lst	This file contains sample and/or informational changes that are not applied to the database by the entity deployer because they could cause data loss. Review the file thoroughly and take action on the entries as necessary for your environment. **Note:** Sterling Selling and Fulfillment Foundation does not provide a .sql file for removing tables from the database but does provide the drop statements in the .lst file. If you want to drop these tables, you must do it manually. Review the entries in the file carefully before taking action.

Table 7. Generated Scripts (continued)

Single Schema Script Name	Multischema Script Name	Description of the script
EFrame_IndexAdds.sql	EFrame_<Pool_Id>_<TableType>_IndexAdds.sql	Adds all of the indexes that need to be created in the database. Modify this file to reference your tablespaces.
EFrame_IndexDrops.sql	EFrame_<Pool_Id>_<TableType>_IndexDrops.sql	Removes any extra indexes in the database.
EFrame_TextIndexAdds.sql	EFrame_<Pool_Id>_<TableType>_TextIndexAdds.sql	Adds new text search indexes that need to be created in the database.
EFrame_TextIndexDrops.sql	EFrame_<Pool_Id>_<TableType>_TextIndexDrops.sql	Removes text search indexes from the database.
EFrame_TextIndexModify.sql	EFrame_<Pool_Id>_<TableType>_TextIndexModify.sql	Updates the text search indexes in the database.
EFrame_TextIndexUpdates.sql	EFrame_<Pool_Id>_<TableType>_TextIndexUpdates.sql	When executed, updates the content of the text indexes.
EFrame_UpdateQueries.sql	EFrame_<Pool_Id>_<TableType>_UpdateQueries.sql	For upgrades, updates the table column values in order to apply other changes made to the columns. For example, if a table column is changed from nullable to not nullable in the installation of a previous release, the column values must be updated before the column can be made not null in the current release because the column default values for the current release may contain null values.

Note: In single-schema deployments, the <INSTALL_DIR>/bin/EFrame_Drops.lst indicates extra objects in the database. In multischema deployments, this file name is <INSTALL_DIR>/bin/EFrame_<Pool_Id>_<TableType>_Drops.lst.

These extra objects could be custom objects or objects that are dropped as the result of a schema change or an upgrade. Please look through this script carefully.

This script may also contain reduced columns. These are columns that were changed to have a smaller size in the newer version. These changes are suppressed because:

- Not all databases will allow you to apply the changes.
- Databases that do allow you to apply the changes can behave unpredictably if the table already contains values that are longer than the new length.

8. Run the scripts specified for your database type, as shown in the following topics. You must run these scripts only if you are manually creating the views

after installation (REINIT_DB=no). In the normal installation mode (REINIT_DB=yes), the views will be applied automatically.

Running DB2 Scripts in a Single-Schema Deployment Procedure

Run the following scripts in the <INSTALL_DIR>/database/db2/scripts directory individually:

- CustomDBViews/transaction/ImportExport_View.sql
- CustomDBViews/transaction/Interop_Views.sql
- CustomDBViews/transaction/InvSnapshot_vw.sql
- CustomDBViews/transaction/yfs_cross_reference_vw.sql
- CustomDBViews/transaction/yfs_iba_ord_demand_vw.sql
- CustomDBViews/transaction/yfs_iba_resv_demand_vw.sql
- CustomDBViews/transaction/yfs_invtdmddtl_vw.sql
- CustomDBViews/transaction/yfs_noPendMove_nodeInventoryDtl_vw.sql
- CustomDBViews/transaction/yfs_onlyLPN_nodeInventoryDtl_vw.sql
- CustomDBViews/transaction/yfs_onlyLPN_NoPendMove_nodeInventoryDtl_vw.sql
- CustomDBViews/transaction/yfs_order_release_line.sql
- CustomDBViews/transaction/yfs_order_release_line_vw.sql
- CustomDBViews/transaction/yfs_nodeInventoryDtl_vw.sql
- CustomDBViews/configuration/yfs_wave_item_vol_config_vw.sql
- CustomDBViews/master/ycm_pricelist_vw.sql
- CustomDBViews/master/ypm_category_item_vw.sql
- CustomDBViews/master/ypm_item_vw.sql

Running DB2 Scripts in a Multischema Deployment Procedure

1. Run all of the scripts within each directory for each schema:
   <INSTALL_DIR>/database/db2/scripts/CustomDBViews/<tabletype>
   where <tabletype> is configuration, transaction, and master.
2. Run the configuration table type scripts for the Configuration schema, the transaction table type scripts for the Transaction schema, and the master table type scripts for the Master/Transaction schema.
3. Additionally, for a multischema deployment, run yfs_addnl_index.sql in
   <INSTALL_DIR>/database/db2/scripts/CustomDBIndexes
   where <tabletype> is configuration and transaction.

Running Oracle Scripts in a Single-Schema Deployment Procedure

Run the following script: <INSTALL_DIR>/database/oracle/scripts/yfs_master_db_script.sql

Running Oracle Scripts in a Multischema Deployment Procedure

1. Run all of the scripts within each directory for each schema:
   <INSTALL_DIR>/database/oracle/scripts/CustomDBViews/<tabletype>
   where <tabletype> is configuration, transaction, and master.

2. Run the configuration table type scripts for the Configuration schema, the transaction table type scripts for the Transaction schema, and the master table type scripts for the Master/Transaction schema.
3. Additionally, for a multischema deployment, run yfs_addnl_index.sql in <INSTALL_DIR>/database/oracle/scripts/CustomDBIndexes

   where <tabletype> is configuration and transaction.

Populating U.S. Zip Codes and Region Schemas After Installation

To make use of U.S. zip codes and region schemas for delivery plan maps and other location-dependent tasks, run the following scripts after installation:

```
<INSTALL_DIR>/database/FactorySetup/Optional/<dbtype>/RegionSchema-US/
RegionSchema-US.sql
```

This script populates the YFS_REGION and YFS_REGION_DETAIL tables.

```
<INSTALL_DIR>/database/FactorySetup/Optional/<dbtype>/ZipCodeLocation/
US_ZipcodeLocation.sql
```

This script populates the YFS_ZIP_CODE_LOCATION table.

Installing Third-Party JAR Files

You can use the install3rdParty utility to add third-party custom jars to the classpath of various utilities and Enterprise Archive (EAR) files.

Following is the syntax for the install3rdParty script:

```
./install3rdParty.sh vendorName vendorVersion <-d | -j | -l | -p | -r > filelist
[-targetJVM EVERY | NOWHERE | DCL | APP | AGENT | [-uninstall]
```

Here

- <vendorName> refers to the name of the vendor such as WebLogic, WebSphere, and JBoss.
- <vendorVersion> refers to the version of the vendor's product.
- [-uninstall] is used to remove a JAR from the JAR directory or the classpath files.

  For example, ./install3rdParty.sh jboss 4_2_0 -j /ais_local/share/vbhat/sandbox/fairlopmaint/install/jar/jboss/4_2_0/jboss-j2ee.jar -targetJVM APP –uninstall. In this example, the jboss-j2ee.jar file will be removed from both the JAR directory, and the APPDynamicClasspath.cfg file.

Pass the appropriate argument based on the file type. You can pass the following arguments:

- -d for database jar/compressed files
- -j for jar/compressed files
- -l for shared libraries
- -p for properties files
- -r for resource properties files

<filelist> refers to the path to your custom file.

For example, to install the wlclient.jar to the dynamic classpath of the agent, use the following command:

```
<INSTALL_DIR>/bin/install3rdParty.sh weblogic 10 -j
<BEA_HOME>/wlserver_10.0/server/lib/wlclient.jar -targetJVM AGENT
```

This command causes the wlclient.jar file to be copied from the WebLogic installation location into the product installation location (<INSTALL_DIR>/jar/weblogic/10/wlclient.jar). The utility then updates the <INSTALL_DIR>/properties/AGENTDynamicclasspath.cfg.in file with the new jar file and invokes the <INSTALL_DIR>/bin/setupfiles.sh utility to regenerate the AGENTDynamicclasspath.cfg file from the modified ".in" file.

If you want to make this new JAR available to the Application Server and Agents when running the install3rdParty utility, pass the following arguments based on your requirements:

Argument
Description

EVERY
Adds the new JAR to all the dynamic classpath files (for example, APPDynamicclasspath.cfg, AGENTDynamicclasspath.cfg, and dynamicclasspath.cfg.

NOWHERE
Adds the new JAR to the <INSTALL_DIR>/jar directory and do not want to update any of the dynamic classpath files

DCL Adds the new JAR to the main Dynamicclasspath.cfg file only

APP Adds the new JAR to the EAR file

AGENT
Adds the new JAR to the AgentDynamicclasspath.cfg file

Note: Sterling Selling and Fulfillment Foundation supports only the options listed above for the install3rdParty utility. Any other options that are displayed with the -help message command are not supported.

If the argument for -targetJVM is not specified, the new jar file is then added to the Dynamicclasspath.cfg file.

Keep the following in mind when using the install3rdParty utility to update a classpath:

- The order of lines in the dynamic classpath files determine the order of the classpath for the application server or agent.
- Whatever is in the beginning of the file is analogous to the jar being in the beginning of the classpath.

For help in using install3rdParty, enter the command, including the -help option, on the command line. The install3rdParty utility prints a usage message.

Development Utilities - Overview

Development utilities enable you to customize Sterling Selling and Fulfillment Foundation to suit your business needs. They are for use while running Sterling Selling and Fulfillment Foundation in development mode.

About the Configuration Deployment Tool

The Configuration Deployment Tool enables you to migrate configuration data from your development environment to your production environment.

For more information about the configuration deployment tool, see the *Sterling Selling and Fulfillment Foundation: Configuration Deployment Tool Guide*.

Truncating Transaction Data

About this task

When deploying Sterling Selling and Fulfillment Foundation to a production environment, you may not want to include all of your transaction data. Sterling Selling and Fulfillment Foundation provides a utility through which you can generate a script to remove transaction data prior to moving into your production environment.

To truncate transaction data:

Procedure

1. From the `<INSTALL_DIR>/bin` directory use the following command appropriate for your database:

   For Oracle and SQL Sever:

   ```
   ./sci_ant.sh(cmd) -f generateTruncateTransactionData.xml
   ```

   For DB2:

   ```
   ./sci_ant.sh(cmd) -Ddbtype=DB2 -f generateTruncateTransactionData.xml
   ```

2. The `TruncateTransactionTables.sql` script is generated and placed in the current directory.
3. To truncate your transaction data, run the newly generated `TruncateTransactionTables.sql` script against your database.

Run-Time Utilities

These utilities start processes that run in the background. The setup of these utilities is described in detail in the *Sterling Selling and Fulfillment Foundation: Properties Guide*.

Integration Server

An integration Server is a process that manages asynchronous services, such as messages to and from external systems. You can run the integration server using the `<INSTALL_DIR>/bin/startIntegrationServer` script.

The Sterling Selling and Fulfillment Foundation Integration Server allows Sterling Selling and Fulfillment Foundation to collaborate with different systems, organizations, and businesses all through a standard, uniform interface to all systems. The Sterling Selling and Fulfillment Foundation Integration Server runs in its own Java Virtual Machine (JVM) environment, separate from your application server.

Agent Server

The agent server utility starts processes responsible for processing transactions generated by the time-triggered transactions (agents). You can start multiple instances of an agent server using the `<INSTALL_DIR>/bin/agentserver.sh <server_name>` script as many times as needed.

Note: If you configure agents to use TIBCO JMS, ensure the prefetch parameter for queues is set to 1.

Note: If you configure agents or integration servers to use JBoss Messaging, uncomment the following "Attribute" xml element in remoting-bisocket-service.xml under:

```
<PATH>/jboss-eap-5.1/jboss-as/server/all/deploy/messaging directory: <attribute name="ping
Frequency" isParam="true">214748364</attribute>
```

Note: Agent and integration servers implement their own, self-managing connection pools. However, the following two properties in the `<INSTALL_DIR>/properties/jdbc.properties` file define the minimum and maximum number of connections:

- `<databasePool>.initsize` - Initial size of the database pool. This is the minimum number of database connections to keep in the pool.
- `<databasePool>.maxsize` - Maximum size of the database pool. This is the maximum number of connections to keep in the pool.

For more information about database pool properties, read the commented descriptions in your `jdbc.properties` file.

Trigger Agent

The trigger agent utility is used for scheduling time-triggered transactions.

You can override the agent criteria attributes only in the Real-time Availability Monitor and Inventory Monitor. The command for triggering the Real-time Availability Monitor and Inventory Monitor with override abilities is:

`triggeragent.sh <criteriaID> -<AgentCriteriaAttribute> <OverriddenValue>` (or `.cmd` on Windows)

To enable this override, you should pass the `AgentCriteriaAttribute` and `OverriddenValue` as additional parameters to the java class in the `triggeragent.sh` (or `.cmd` on Windows) file as follows:

```
java com.yantra.ycp.agent.server.YCPAgentTrigger -criteria %*
```

Therefore, when you invoke:

```
triggerAgent.sh CustomCriteria -MyOverriddenParam DynamicValue
```

all the values are passed to the java class.

However, do not modify the parameters passed to the java class in the default `triggeragent.sh` (or `.cmd` on Windows) file. Make these changes in the file that you have copied and renamed from the `triggeragent.sh` (or `.cmd` on Windows). Also, the agent criteria XML code must have the `AllowedOverriddenCriteria` flag set to `Y`.

sender.sh Utility

The sender.sh utility invokes TestClientSender and is used for testing. It enables you to execute an API or service from the command line, as follows:

```
java
com.yantra.integration.adapter.client.TestClientSender<flowNam
e/systemApiName> <is firstParameter Flow (Y/N)> <xmlFileName>
```

Here,

- The first argument takes the name of a service/sdf (*flowName*) or an API name (*systemApiName*).
- The second argument (*is firstParameterFlow (Y/N)*) determines whether the first argument is a service or an API. Valid values are Y and N. If the first argument is a service, use Y; if it is an API, use N.
- The third argument (*xmlFileName*) takes the path and name of the XML input file you want to use as input to the API or service.

Setting Up the Runtime Utilities

You can use WebLogic, WebSphere, JBoss, and TIBCO for the Java Messaging Service (JMS).

The CLASSPATH for the `startIntegrationServer`, `agentServer` and `triggerAgent` scripts must include certain jar files in order for them to be used on WebLogic, WebSphere MQ, JBoss, or TIBCO. Use the `<INSTALL_DIR>/bin/install3rdparty` script to include these respective jar files - as they are listed in this section - in the `AGENTDynamicclasspath.cfg` dynamic classpath file.

The JDK used by the Runtime Utilities is determined by the AGENT_JAVA_HOME property in <INSTALL_DIR>/properties/sandbox.cfg. This JDK should point to the same JDK that is used to run your application server. For more information about configuring sandbox.cfg, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.

If you have developed custom Java classes (user exits, event handlers, and so forth), see the section on "Including Custom Classes" for your application server in Chapter 14, “Deploying Sterling Selling and Fulfillment Foundation,” on page 109.

About Setting Up Runtime Utilities on IBM WebSphere

Note: The classpath information provided in this section is intended as a starting point for your implementation. Depending upon the JMS, JNDI, and JDK you are using you may need to add or remove jars from the lists given here.

Both WebSphere and MQ have jars which are required for running the agent and integration servers. You can obtain these jars from the WebSphere or MQ server.

IBM WebSphere

If you are using WebSphere Default Messaging, include the following jars:

- `<WAS_HOME>/runtimes/com.ibm.ws.sib.client.thin.jms_version_number.jar`
- `<WAS_HOME>/runtimes/com.ibm.ws.ejb.thinclient_version_number.jar`
- `<WAS_HOME>/runtimes/com.ibm.ws.orb_version_number.jar` (for non-IBM JREs only)

where *version_number* is the version of WebSphere you are using.

Note: If you are using WebSphere version 7.0.0.7 or higher, set the following property to true in the WebSphere console under Application Servers > *your server name* > web container > custom properties:
`com.ibm.ws.webcontainer.dispatcherRethrowSER=true`

IBM WebSphere MQ Using "fscontext" jndi

When using WebSphere MQ you should obtain the following jars from MQ_HOME/java/lib:

- com.ibm.mq.jar
- com.ibm.mqjms.jar
- fscontext.jar
- dhbcore.jar
- jms.jar
- jta.jar

providerutil.jar

Setting Up Runtime Utilities on Oracle WebLogic

About this task

Note: The classpath information provided in this section is intended as a starting point for your implementation. Depending upon the JMS, JNDI, and JDK you are using you may need to add or remove jars from the lists given here.

Include the following jar files for WebLogic JMS:

Procedure

Use the <INSTALL_DIR>/bin/install3rdparty.sh (or .cmd) script to install the wlfullclient.jar file and include it in the AGENTDynamicclasspath.cfg dynamic classpath file.

Results

For more information about developing a WebLogic Full Client, see:

http://download.oracle.com/docs/cd/E12840_01/wls/docs103/client/jarbuilder.html

Note: For more information about using the install3rdparty script, see “Installing Third-Party JAR Files” on page 101.

If you are using WebLogic Server with WebSphere MQ JMS and you are binding queues in WebLogic JNDI, use install3rdparty to include the following files from MQ_HOME/java/lib in the AGENTDynamicclasspath or the APPDynamicclasspath:

- com.ibm.mq.jar
- com.ibm.mqjms.jar
- connector.jar
- jms.jar
- jta.jar
- wlfullclient.jar
- dhbcore.jar

Note: If you are using WebLogic JMS 10.0 or later, refer to the Web site http://download.oracle.com/docs/cd/E12840_01/wls/docs103/client/jarbuilder.html and add wlfullclient.jar to the AGENTDynamicclasspath.cfg, using the install3rdparty.sh (or .cmd) script.

If you are binding queues in File Bindings JNDI, use the following files, obtainable from MQ_HOME/java/lib:

- com.ibm.mq.jar
- com.ibm.mqjms.jar
- connector.jar
- jms.jar
- jta.jar
- fscontext.jar
- providerutil.jar
- dhbcore.jar

About Setting Up Runtime Utilities on JBoss

Note: The classpath information provided in this section is intended as a starting point for your implementation. Depending upon the JMS, JNDI, and JDK you are using you may need to add or remove jars from the lists given here.

If you are using JBoss JMS, add the following jars in the classpath using the <INSTALL_DIR>/bin/install3rdparty script:

- javassist.jar
- jboss-aop-client.jar
- jboss-common-core.jar
- jboss-logging-log4j.jar
- jboss-logging-spi.jar
- jboss-mdr.jar
- jboss-remoting.jar
- jnp-client.jar
- trove.jar
- jboss-serialization.jar
- jboss-messaging-client.jar

Do not include any *ui.jar files.

- If you are using the JBoss Messaging, add the jbossmq-client.jar in the classpath using the <INSTALL_DIR>/bin/install3rdparty script. This is for agents only - do not add JBoss jars to APPdynamicclasspath.cfg.
- If you are using JBoss application server, add the log4j.jar file from JBoss at the beginning of your CLASSPATH.
- If you are invoking a JSP page for the first time, there may be a short delay while the JBoss application server compiles the JSP page. To avoid this delay, precompile the JSP files before creating the smcfs.ear file.
- For information about precompiling JSP files, see the *Sterling Selling and Fulfillment Foundation: Performance Management Guide*.

About Setting Up Runtime Utilities on TIBCO

Note: The classpath information provided in this section is intended as a starting point for your implementation. Depending upon the JMS, JNDI, and JDK you are using you may need to add or remove jars from the lists given here.

If you are using TIBCO JMS:

- Use the <INSTALL_DIR>/bin/install3rdparty.sh(cmd)script to install the jms.jar and tibjms.jar files and include them in the AGENTDynamicclasspath.cfg dynamic classpath file.
- Some agents will require servlet classes. For these agents, use the install3rdParty script to install the <INSTALL_DIR>/jar/geronimo/2_2/geronimo-servlet_2.5_spec-1.2.jar into the AGENTDynamicclasspath.cfg file.

Chapter 14. Deploying Sterling Selling and Fulfillment Foundation

Sterling Selling and Fulfillment Foundation Deployment - Overview

After configuring Sterling Selling and Fulfillment Foundation according to your business needs, deploy it into production based on your application server. Deployment is part of the general path that you follow when installing and deploying Sterling Selling and Fulfillment Foundation:

1. Installing the application server (JBoss, WebLogic, or WebSphere). Refer to the documentation for the application server.
2. Installing Sterling Selling and Fulfillment Foundation. Refer to the installation information for the operating system (UNIX/Linux or Windows).
3. Building the Enterprise Archive (EAR).
4. Starting the application server.
5. Deploying Sterling Selling and Fulfillment Foundation.

To deploy Sterling Selling and Fulfillment Foundation in a development environment, consider the following:

- Run the development environment in exploded (non-ear) mode. See the *Sterling Selling and Fulfillment Foundation: Customization Basics*.
- Build and deploy the development-related documentation such as API Javadocs, ERDs, and XSDs as online resources in the development environment. To build and deploy documentation EAR, `smcfsdocs.ear`, along with the application EAR, refer to *Sterling Selling and Fulfillment Foundation: Customization Basics*.

Before deployment, verify if you have applied all the concepts that pertain to your environment, and have completed the Performance Recommendations Checklist as described in the *Sterling Selling and Fulfillment Foundation: Performance Management Guide*.

Tip: To enable faster loading of a JSP page, pre-compile your JSP files. For information on how to do this, see the JSP Pre-compilation section of the *Sterling Selling and Fulfillment Foundation: Performance Management Guide*.

Note: If you are planning on installing any of the Sterling Selling and Fulfillment Foundation Packaged Composite Application (PCAs), or applying any extensions, you may want to consider delaying the building of your Enterprise Archive (EAR) until all of your PCAs are installed. Building the EAR now and for each PCA or extension installation does not cause harm, but does save time if you build your EAR only once after all PCAs or extensions are installed.

Configuring an Oracle WebLogic Application Server - Overview

This section includes the tasks necessary for configuring an Oracle WebLogic application server for use with Sterling Selling and Fulfillment Foundation.

Setting Up the WebLogic Script File

About this task

If you are using HP-UX 11iv3, verify that your kernel parameters are set according to Oracle's recommendations before you set up the WebLogic application server. For these recommendations, go to: http://download.oracle.com/docs/cd/E13196_01/platform/suppconfigs/configs/hpux/hpux_11iv3_103.html

To set up the WebLogic script file:

Procedure

1. The following properties are supplied by Oracle in the <*your_WebLogic_domain*>/bin/startWebLogic.sh (or .cmd) file. Modify each property listed in Table 8 according to its description.

Table 8. startWebLogic.sh Properties

Property	Description
JAVA_OPTIONS	Java command line options for running the server. Depending on your JVM vendor, specify as follows: • For IBM, set this value to `-Xms768m -Xmx768m` • For JRockit, set this value to `-Xms768m -Xmx768m` • For HP, set this value to `-XX:MaxPermSize=512m -Xms768m -Xmx768m` For Sun, set this value to `-XX:MaxPermSize=512m -Xms768m -Xmx768m`

DBDRIVERS	Specify the paths to your data base drivers as the first item in the value of the CLASSPATH.
	
	The out-of-the-box CLASSPATH setting is:
	`CLASSPATH="${CLASSPATH}${CLASPATHSEP}${MEDREC_WEBLOGIC_CLASSPATH}"`
	
	Change this so that the path to the drivers .jars is first. For example:
	
	DBDRIVERS=/<directory_path_to_oracle_drivers>/ojdbc6.jar
	`CLASSPATH="${DBDRIVERS}${CLASSPATHSEP}${CLASSPATH}${CLASSPATHSEP} ${MEDREC_WEBLOGIC_CLASSPATH}"`

JITC_COMPILEOPTS	For AIX, specify as "NQCLSINIT"
-Dfile.encoding	To ensure that all the Sterling Selling and Fulfillment Foundation UI screens display UTF-8 characters, specify as follows after {JAVA_OPTIONS} for Java commands : `-Dfile.encoding=UTF-8` This is applicable to all the Sterling Selling and Fulfillment Foundation Java start-up scripts.
-Dvendor	System property. Specify as an argument to the Java command after {JAVA_OPTIONS}. Can be “shell” or “weblogic” depending upon whether datasource is being used or not. `-Dvendor=shell`
-DvendorFile	System property. Specify as an argument to the Java command after {JAVA_OPTIONS}. `-DvendorFile=/servers.properties`

Table 8. startWebLogic.sh Properties (continued)

Property	Description
Example: `${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} -Dweblogic.Name=${SERVER_NAME} -Djava.security.policy= ${WL_HOME}/server/lib/weblogic.policy ${JAVA_OPTIONS} -Dfile.encoding=UTF-8 -Dvendor=shell -DvendorFile=/servers.properties ${PROXY_SETTINGS} ${SERVER_CLASS}`	

2. If you are using an HTTPS transport, download the Secure Socket Extension (JSSE) 1.0.3 package from http://java.sun.com and add the following files to the <JAVA_HOME>/jre/lib/extn/ directory:
   - jnet.jar
   - jcert.jar
   - jsse.jar

Configuring Oracle WebLogic XML Registry

About this task

You must configure WebLogic to run properly with Sterling Selling and Fulfillment Foundation.

To configure WebLogic:

Procedure

1. From the WebLogic Console menu, choose **Services** > **XML Registries**.
2. Click **New**.

   Note: If the XMLs have UTF-8 specified as encoding in the header, then you do not need to specify any XML registry.
3. Click **Next**. Select the WebLogic application server or cluster to which you would like to deploy this XML Registry.
4. Click **Finish**.

Disabling Instrumented Stack Traces in WebLogic

About this task

You can eliminate additional stack traces resulting from an error on an API call in EJB mode.

To eliminate stack traces:

Procedure

1. From the WebLogic System Administration Console, select each server on which Sterling Selling and Fulfillment Foundation is deployed.
2. Select Logging.
3. Uncheck the checkbox for Instrument Stack Traces and choose Apply.

Setting Up WebLogic to Display Barcodes and Graphs

About this task

Sterling Selling and Fulfillment Foundation uses X Window functionality to display barcodes and dynamic graphical images (such as inventory supply and demand graphs) in a UNIX environment.

The following configuration is required to enable the X Window environment in UNIX systems for a WebLogic application server:

Procedure

1. If your UNIX server is also an X Window client, edit the startWebLogic.sh script, and set the DISPLAY environment variable as follows:

   ```
   export DISPLAY=IP_address_of_XWindows_server:0.0.
   ```

2. If you are using UNIX, run the `xhost +` command to remove access control for your X Window server.

   You can run X server on the same server on which you run Sterling Selling and Fulfillment Foundation. However, you need to be logged into the server console.

   Note: If the X Window server goes down or crashes while the inventory user interface is using the jbchartx.jar file, the WebLogic server also goes down.

Setting Up WebLogic to Use HTTP In-Memory Session Replication

About this task

Sterling Selling and Fulfillment Foundation supports HTTP in-memory session replication on the following configuration:

Apache 2.0.44 with the WebLogic plug-in as the proxy server with idempotent set to OFF

We advise testing session replication if you are using a different proxy.

The `weblogic.xml` file should be edited to set up WebLogic for in-memory session replication as follows:

Procedure

1. Build the EAR file.
2. Copy the `<INSTALL_DIR>/tmp/build<package_name>/WEB-INF/weblogic.xml` file to the `<INSTALL_DIR>/extensions/<package_name>` directory, where `<package_name>` is the application name; for example, `smcfs`.
3. Add the following lines to the `weblogic.xml` file:

   ```
   <session-descriptor>
   <session-param>
   <param-name>PersistentStoreType</param-name>
   <param-value>replicated</param-value>
   </session-param>
   </session-descriptor>
   ```

4. Rebuild the EAR file.

Building the Enterprise Archive (EAR) Package on WebLogic

Note: Sterling Selling and Fulfillment Foundation supports overriding the context root during EAR deployment.

When deploying Sterling Selling and Fulfillment Foundation on WebLogic, use the `smcfs.ear` file, which may contain:

- `smcfs.war` - Web module that contains all of the Sterling Selling and Fulfillment Foundation JSPs and other Web application components.
- `sma.war` - Web module that contains the System Management Administrator application components.
- `sbc.war` - Web module that contains all of the IBM Sterling Business Center Web application components.
- `yantrawebservices.war` - Web module that contains all of the Sterling Selling and Fulfillment Foundation Web services interface classes.
- `smcfsejb.jar` - The EJB module that contains all the Sterling Selling and Fulfillment Foundation EJBs. You can pass an alternate earfile name by using the `–Dearfile` option to the `buildear.sh` script. Doing this will result in a name change for the ejb jar file. For example, if you specify an EAR file as `xyz.ear`, the ejb jar becomes `xyzejb.jar`.
- `smcfswsbe.jar - The backend Web services jar file. You get this file if you expose Web services. You can pass an alternate earfile name to the ear build script. Doing this will result in a name change for the Web services backend jar file. For example, if you specify an EAR file as xyz.ear, the Web service backend jar becomes xyzwsbe.jar.`
- Jars that contain backend business logic.
- Jars that contain third-party libraries accessed by backend logic.

Each of the third-party JAR files are left as is and in the manifest of the application each file is indicated as a dependency. For example, log4j files are represented separately as `log4j-1.2.15.jar` with a dependency in the application.

Building Web Services on WebLogic

You can expose APIs and synchronous SDF services through web services. Web services require some configuration prior to running the buildear script to create the application EAR and web services WAR file. The configuration takes place in either the namedwebservices.xml file or webservicebeans.xml file.

There are two types of web services that can be created:

- EJB: The EJB web service accepts two string inputs and returns one string output. No information about the content of the strings is included in the WSDL for this web service; the caller must open the javadocs for the API to get the information about how to structure the content. This makes it very difficult to dynamically generate calls to the XAPI web services without an additional source of information beyond the WSDL. This type of web service is created using EJB beans, and is created by default with the application EAR.
- JAX-WS: The WSDL of the JAX-WS web service includes information about the input expected by the server, the output, and exceptions, which makes it easier to dynamically generate calls to the web services. JAX-WS web services are created using JAXB beans. The advantage of using JAXB beans is that they are fully self-describing and alleviate the need to have access to javadocs for the API. To facilitate securing JAX-WS based web services, they are created with

handlers. You can use the default server and client handlers that are delivered with Sterling Selling and Fulfillment Suite or create your own, custom handlers.

You can choose to build the application with one or both types of web services.

Note: You can also choose to suppress the web services build by using the -Dnowebservice=true option on the buildear command.

Note: If using EJB based web services or using JAX-WS based web services without handlers and API security is enabled, ensure that you expose the Login API. If using custom handlers, this decision will depend upon how your handlers are written. For more information about web services security, see the *Sterling Selling and Fulfillment Foundation: Secure Deployment Guide*.

EJB Web Services Overview

An EJB web service accepts two string inputs and returns one string output. No information about the content of the strings is included in the WSDL for this web service; the caller must open the javadocs for the API to get the information about how to structure the content. This makes it very difficult to dynamically generate calls to the XAPI web services without an additional source of information beyond the WSDL. This type of web service is created using EJB beans, and is created by default with the application EAR.

Defining an EJB Web Service with WebLogic

Procedure

1. In the `<INSTALL_DIR>/repository/eardata/platform/webservices` folder, locate the namedwebservices.xml.sample file. Rename (or copy) it to namedwebservices.xml. This is the file you will edit.
2. In namedwebservices.xml, specify each API you want to expose as a web service in an Api/Name attribute.

   Property
   Description

   ServiceName
   The name of the service that you configured using the Service Definition Framework (SDF).

   ExposedName
   The name that is used in the Web Services Description Language (WSDL) file. This is the name that is used to call the web service programmatically. When specifying a service name for ExposedName, choose a literal that does not match any of the standard application API names.

   The exposed name must start with a lower case letter.
3. Set the properties necessary for your application server in sandbox.cfg:

   Parameter
   Description

   BEA_DIR
   Set the value to specify the <WL_HOME> directory. (You may need to create the property.)

   Required if building EJB web services.

   For WebLogic only.

WEBSERVICES_BUILDS

Takes a comma-separated list that can include YIFWebService and SIXBeanXapiJaxWS. Defaults to YIFWebService.

Required for EJB and JAX-WS web services, on all application servers.

4. Save the file, navigate to the bin directory, and run the setupfiles command:
   - For UNIX/Linux: `<INSTALL_DIR>/bin/setupfiles.sh`
   - For Windows: `<INSTALL_DIR>\bin\setupfiles.cmd`
5. Create the EAR as described in "Creating the EAR on WebLogic" on page 122. The web services defined in the file will be built when you create the application EAR.
6. Later, if you want to add more APIs and services as EJB Web services, repeat these steps.

About EJB Web Services and Security

How Sterling Selling and Fulfillment Suite Supports EJB Web Services Security

Transport Security

The concept of transport layer security is an option that can be applied to all types of http traffic including EJB based web services.

For YIFWebServices, use transport security. Sterling Selling and Fulfillment Suite does not provide any other security mechanisms or recommendations for EJB based web services.

See your application server documentation for information about security available on the application server itself.

Client Generation

Sterling Selling and Fulfillment Suite does not provide any assistance around client generation for YIFWebService. One popular pattern is to use the Axis client generation tasks for ant provide by Apache Axis. The same rules apply here. Be sure to configure the trust-store properly and set the required system properties in your code before communicating with the https port of the server.

JAX-WS Web Services Overview

You can expose APIs as web services. Before you create the application EAR file, you need to define which APIs will be exposed as web services so that they are included in the web services WAR file in the EAR. This section provides a high-level overview of how to configure a JAX-WS web service.

The following APIs are not supported with JAX-WS web services:

- createAsyncRequest
- evaluateAdvancedCondition
- evaluateCondition
- getAgentCriteriaList
- getPage
- invokeUE
- multiApi
- printDocumentSet

Performance Considerations

Injecting handlers into the web services stack can have performance implications. Customers should take care in designing handlers that the actions they perform will not be overly costly since these handlers will get invoked (potentially multiple times) for every API call that is made.

About JAX-WS Web Services Security

Sterling Selling and Fulfillment Suite provides:

- Information about ways to implement transport security for JAX-WS based web services - see the *Sterling Selling and Fulfillment Foundation: Secure Deployment Guide* sections about web services security for more information.
- JAX-WS handlers to implement security for JAX-WS based web services. Handler-chains are similar in concept to servlet filter chains. A default implementation is provided, but you can plug in your own customized handler chain xml and classes. See the *Sterling Selling and Fulfillment Foundation: Customizing Web Services* for more information.

Defining a JAX-WS Web Service on WebLogic

Procedure

1. Set the properties necessary for your application server in sandbox.cfg:

Parameter	Description
WEBSERVICES_BUILDS	Required. Can take a comma separated list that can include YIFWebService and SIXBeanXapiJaxWS.
XBEAN_PACKAGE	Optional. By default, the xbean package names and namespace generated for JAX-WS web services include the word "documentation." Use this parameter to replace the word "documentation" with another string. This value can be any string which would form a valid java package name. **Important:** Regardless of how it is entered, Sterling Selling and Fulfillment Foundation will always convert this string to all lowercase characters to comply with Java standards.
JAXB_LOCAL_SCOPING	Required. Valid values are true/false. Set to true. This flag affects the way that JAXB generates beans. Everything becomes “toplevel” so all classes will exist in the default package rather than existing in a hierarchy. Typically, for Sterling Selling and Fulfillment Foundation APIs, not using top-level scoping will result in generated file names that are too long for the file system.

Parameter	Description
JAXB_ALWAYS_ANNOTATE_CLASSNAMES	Required. The classes that are generated by JAXB are given "1-up" numbers at the end of their names to prevent having multiple classes with the same name when top level scoping is used. By default, this flag is included and set to true (yes, annotate all classes).If you prefer to minimize the use of one-up numbers, specify the AnnotateClassNames attribute in the webservicebeans.xml file for selected APIs, as shown in the following example. `<Bean BeanName="ParticipantBean" BeanPackage="com.sterlingcommerce.jaxws.participant. webservices" > <Apis> <Api AnnotateClassNames="true" Name="getOrganizationHierarchy" ExposedName="getOrganizationHierarchy" /> <Api AnnotateClassNames="true" Name="getOrganizationList" ExposedName="getOrganizationList" /> <Api AnnotateClassNames="false" Name="getPersonInfoList" ExposedName="getPersonInfoList" /> <Api AnnotateClassNames="false" Name="login" ExposedName="login" /> </Apis> </Bean>`
SUPPRESS_JAXWS_HANDLERS	By default, Sterling Selling and Fulfillment Suite applies handlers to JAX-WS based web services as part of security enablement. If you do not want the handlers applied, set this sandbox.cfg variable to true. For more information about web services security, see the *Sterling Selling and Fulfillment Foundation: Secure Deployment Guide*.

2. Save the file, navigate to the bin directory, and run the setupfiles command:
   - For UNIX/Linux: `<INSTALL_DIR>/bin/setupfiles.sh`
   - For Windows: `<INSTALL_DIR>\bin\setupfiles.cmd`
3. In the `<INSTALL_DIR>/repository/eardata/platform/webservices` folder, locate the `webservicebeans.xml.sample` file. Copy the file to `webservicebeans.xml`. This is the file you will edit.

webservicebeans.xml attributes	Description
Bean attributes **Note:** Do not put the same API into two beans.	
BeanName	Required. Enter a descriptive name for the web service. This name is used by the WSDL generator and reflected in the client code. For example, a company named "Dave's BBQ" might use the following: `BeanName=DavesBBQJaxWS`
BeanPackage	Required. Enter a descriptive name for the package. Each bean must have a unique package name. This name is used by the WSDL generator and reflected in the client code. For example, a company named "Dave's BBQ" might use the following: `BeanPackage="com.DavesBBQ`
API element attributes	

webservicebeans.xml attributes	Description
Api Name	The name of the API that you wish to expose.
ExposedName	The name that is used in the Web Services Description Language (WSDL) file. This is the name that is used to call the web service programmatically. When specifying a service name for ExposedName, choose a literal that does not match any of the standard application API names. The exposed name must start with a lower case letter.
AnnotateClassNames	Specify the `AnnotateClassNames` attribute on the `webservicebeans.xml` to indicate which APIs need to have annotations. This flag provides a more granular way of indicating that 1-up annotations are required for specific APIs as compared to the sandbox variable ALWAYS_ANNOTATE_CLASS_NAMES.
Service element attributes	
Service Name	The name of the service that you configured using the Service Builder.
ExposedName	The name that is used in the Web Services Description Language (WSDL) file. This is the name that is used to call the web service programmatically. When specifying a service name for ExposedName, choose a literal that does not match any of the standard application service names. The exposed name must start with a lower case letter.
AnnotateClassName	Specify the `AnnotateClassNames` attribute on the `webservicebeans.xml` to indicate which APIs need to have annotations.
To expose an SDF service as a web service, you must also complete the procedure in "Exposing an SDF Service as a Web Service on WebLogic" on page 119.	

4. Save the file.
5. Create the EAR as described in "Creating the EAR on WebLogic" on page 122. The web services defined in the file will be built when you create the application EAR.

   CAUTION: When building an ear with JAX-WS Web services, you may encounter out of memory errors. JAX-WS Web services will generate a java class for each element in the XSD, which can exceed the system memory. If the ear build runs out of memory, then unnecessary APIs should be removed from the webservicebeans.xml file. Additionally, APIs with potentially large output XMLs may be exposed through services with reduced XSDs to eliminate unwanted elements.

   Tip: To a certain extent, the memory requirements for building JAX-WS beans is on a bean-by-bean basis. This means that in some cases, exposing a set of APIs through one JAX-WS bean can cause an out-of-memory error during the ear build. Exposing the same APIs by splitting them across multiple beans may prevent the out-of-memory error and enable the ear to build successfully.
6. Later, if you want to add more APIs and services as JAX-WS web services, repeat these steps.

Example of webservicesbeans.xml file - WebLogic

The following is an example of a webservicebeans.xml file:

```
<WebServices>
  <Beans>
   <Bean BeanName="TheBeans"
BeanPackage="com.sterlingcommerce.jaxws.the.webservices" >
```

```
    <Apis>
      <Api Name="login" ExposedName="theLogin" />
      <Api Name="getLocaleList" ExposedName="theGetLocaleList" />
    </Apis>
   </Bean>
   <Bean BeanName="OtherBeans"
BeanPackage="com.sterlingcommerce.jaxws.other.webservices" >
    <Apis>
         <Api Name="getLocaleList" ExposedName="otherGetLocaleList" />
    </Apis>
     <Services>
      <Service AnnotateClassNames="true" Name="testWS" ExposedName="testWS" />
      <Service Name="testWS2" ExposedName="testWS2" />
    </Services>
   </Bean>
  </Beans>
</WebServices>
```

Exposing an SDF Service as a Web Service on WebLogic

About this task

To expose an SDF service as a web service, you first have to copy the input and output xsds for the API that is exposed in the SDF service and edit them.

Note: If you want to overwrite the input or output XSD for any API, you must expose the API as an SDF service and then expose the SDF service as a Web service. JAX-WS does not support API output templates but the equivalent can be achieved by using the following procedure.

Tip: You can use the same procedure to design fully customized APIs exposed through the SDF. To do so, write your own XSD for the input and output of your API, and ensure that you avoid constructs that are not supported in JAX-B such as xsd:any.

Procedure

1. Open the SDF and create a synchronous service that has an API node.
2. Select the desired API.
3. Save the SDF service.
4. Copy the input and output XSD files for the API you exposed in the SDF from the following location:

   `<INSTALL_DIR>/xapidocs/api_javadocs/XSD`

   to:

   `<INSTALL_DIR>/extensions/webservices`
5. Rename the files to match the name and exposed name that you give them in webservicebeans.xml.

   Use the following format:

   *<name>_<exposed_name>*_input.xsd

   *<name>_<exposed_name>*_output.xsd

Results

This is also the procedure you would use to extend a table, with this additional step: after copying and renaming the files, edit them to reference the new columns/API inputs/outputs.

Running the JAX-WS Client Generator

About this task

Sterling Selling and Fulfillment Foundation provides a client generator for use with JAX-WS web services on UNIX and Linux (the client generator is not supported on Windows). If you are modifying the XSD for any API or service which has been exposed as a web service, you must rebuild the web services client. Additionally, each time you extend the database through customization or upgrade, you must run the xsdgenerator, and then rebuild the ear and client. See *Sterling Selling and Fulfillment Foundation: Customizing APIs* for more information about using the xsdgenerator, and see *Sterling Selling and Fulfillment Foundation: Extending the Database* for more information about extending the database. See "Building the Enterprise Archive (EAR) Package on WebLogic" on page 113 for information about building the ear.

To generate a client against an HTTPS url, include the trustStore and keyStore system properties when you run the script.

Note: If you are using a SUN JDK or HP-UX JDK, create the file *install_dir* /jdk/jre/lib/stax.properties (or equvalent directory if sandbox.cfg variable *JAVA_HOME* points to an external directory.) Put the following line in stax.properties:

```
javax.xml.stream.XMLInputFactory=com.sun.xml.internal.stream.
XMLInputFactoryImpl
```

Otherwise, you may see client build failures.

Procedure

To invoke the generator, run the buildjaxclient.sh script from the *install_dir* directory. Include the following system properties when running the script:

Property	Description
BEAN_NAME	Required. Matches the beanname on the server. The service name may differ, depending on the application server.
APPSERVER	Required. Valid values are websphere, weblogic, or jboss.
SERVER_URL	Required. The HTTP URL of the server up to, but not including, the bean name.
trustStore	Full path to your client truststore (required when generating client from the https url).
trustStorePassword	Password for your client truststore.
keyStore	Full path to your client keystore (required to achieve two-way ssl connection when generating a client from the https url).
keyStorePassword	Password for your client keystore.

The following code samples are examples of how the script might be set up in different scenarios.

- Without ssl:

  ```
  bin/buildjaxclient.sh -DBEAN_NAME=ParticipantBeanService
  -DAPPSERVER=weblogic -DSERVER_URL=http://00.00.00.00:/SIXBeanXapiJaxWS
  ```

- For one way ssl:

```
bin/buildjaxclient.sh -DBEAN_NAME=ParticipantBeanService
-DAPPSERVER=weblogic -DSERVER_URL=http://00.00.00.00:/SIXBeanXapiJaxWS
-DtrustStore=/home/joe_user/resources/ssl/jaxwsclienttrust.jks
-DtrustStorePassword=changeit
```

- For two way ssl:

```
bin/buildjaxclient.sh -DBEAN_NAME=ParticipantBeanService
-DAPPSERVER=weblogic -DSERVER_URL=http://00.00.00.00:/SIXBeanXapiJaxWS
-DtrustStore=/home/joe_user/resources/ssl/jaxwsclienttrust.jks
-DtrustStorePassword=changeit -DkeyStore=/home/joe_user/resources/ssl/
jaxwsclientkey.jks -DkeyStorePassword=changeit
```

When run, the JAX-WS client generator creates a new folder, jaxwsclient, that contains the compiled classes and source code. The folder structure will be similar to the following: *install_dir*/jaxwsclient/output/*beanname*

Sample Code for BeanService Classes - WebLogic

Sample code is generated in the same directory as the bean and BeanService class. You can use this sample as a reference for invoking the bean methods.

The sample file has the same name as the service class but with "Sample" appended to the end of the name.

For example, SMCFSBeanService would have an accompanying sample called SMCFSBeanServiceSample class.

```
jaxwsclient/output/ParticipantBeanService/com/sterlingcommerce
/jaxws/participant/webservices/ParticipantBeanServiceSample
```

The class generates a method for each API you expose. The content will require editing.

```
public void testlogin( ) throws Exception {
com.sterlingcommerce.documentation.ycp.login.input.Login input
= new com.sterlingcommerce.documentation.ycp.login.input.Login();
        /*
Insert custom code here to set values on the input object.
        */
com.sterlingcommerce.documentation.ycp.login.output.Login
returnValue = b.login(env,input);
        /*
Insert custom code here to retrieve values from the return object.
        */
    }
```

Including Custom Classes on WebLogic

About this task

When deploying Sterling Selling and Fulfillment Foundation as Web services on WebLogic, if you have developed custom Java classes (user exits, event handlers, and so forth) you need to deploy them in order for them to be available.

To ensure that your custom classes get invoked, do the following:

Procedure

1. Create a JAR file with all your custom classes.
2. Place this JAR file in a folder structure based on the package name. For more information about packaging and deploying jar files, see the *Sterling Selling and Fulfillment Foundation: Customization Basics*.

3. Rebuild the EAR as described in "Creating the EAR on WebLogic."

Results

The custom classes are automatically included in the `smcfs.ear` file.

Creating the EAR on WebLogic

About this task

Note: Set the number of file descriptors (`ulimit -n`) for the user creating the EAR to be greater than 8192. If you are deploying on HP set `ulimit unlimited` for the user creating the EAR.

During the ear build, when JAX-WS webservice creation occurs, Sterling Selling and Fulfillment Suite will look for customer extensions and use them if they exist. If they do not exist, default handler XML and handler classes will be built into the ear and used. Sterling Selling and Fulfillment Suite provides one single set of default handlers that will get used against all beans that are deployed and for which customers have not provided extensions. Customer extensions can be provided on a per JAX-WS bean basis.

Enterprise Archives are built using an ANT (buildEAR.xml) that accepts the following targets:

Main Target
: **Description**

create-ear
: Creates `smcfs.ear` - The Sterling Selling and Fulfillment Foundation application EAR file.

Procedure

To create an application EAR file, run the following command from the `<INSTALL_DIR>/bin directory:`

```
./buildear.sh (.cmd for Windows) -Dappserver=weblogic -Dwarfiles=smcfs,sma,sbc
 -Dearfile=smcfs.ear create-ear
```

Note: Sterling Selling and Fulfillment Foundation supports the RPC encoded or document literal style and usage of invocation for Web services. When choosing style and usage for WebLogic, the Web service uses the document literal only. Running this command creates the `smcfs.ear` file in the `<INSTALL_DIR>/external_deployments/` directory. It also puts three war files into the `smcfs.ear`:

- `smcfs.war` - The Sterling Selling and Fulfillment Foundation application war file
- `sma.war` - The System Management Administrator application war file
- `sbc.war` - The IBM Sterling Business Center application war file

  Note: You can add the following options to the end of the above buildear commands:
  - `-Dnowebservice=true` If you do not want to use Web services.
  - `-Ddevmode=true` if you want to use the HTTP API Tester in the development environment.
  - `-Dnodocear=true` if you want to skip the documentation build. Use this option when building the ear for a production environment. The doc ear does not contain end-user documentation, such as context-sensitive help files. It

contains only development-related documentation, including API Javadocs, ERDs, and XSDs and should not be deployed to a production server.

– `-Dsupport.multi.war=true` if you want to copy all the UI jars to `<WAR>/WEB-INF/lib`. The UI jars will be copied based on the entry in `DCL.xml`.

Results

For more information about the System Management Administrator (SMA) see the *Sterling Selling and Fulfillment Foundation: System Management and Administration Guide*. For more information about IBM Sterling Business Center, see the *Business Center: Item Administration Guide* and *Business Center: Pricing Administration Guide*.

For more information about WebLogic, see the documentation provided by the vendor.

Installing and Deploying Sterling Selling and Fulfillment Foundation on Different Servers

About this task

This section applies only to users who are installing and deploying Sterling Selling and Fulfillment Foundation on separate systems.

In order for this scenario to deploy successfully, you must identify a log directory on the system where you are deploying Sterling Selling and Fulfillment Foundation.

Procedure

1. Set the LOG_DIR property in sandbox.cfg to a value that is meaningful on the system where the EAR will be deployed.
2. Run the `<INSTALL_DIR>/bin/setupfiles.sh` (or `setupfiles.cmd`) script.
3. Rebuild the EAR file.
4. Edit sandbox.cfg to set the LOG_DIR value back to its original value.

Results

For more information about sandbox.cfg and changing properties, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.

Precompiling the WAR File on WebLogic

To improve the performance when initially loading UI resources, IBM recommends that you precompile the jsps that comprise the WAR file. For more information about how to precompile jsps, see the *Sterling Selling and Fulfillment Foundation: Performance Management Guide*.

Deploy the new EAR file as described in "Deploying the Enterprise Archive (EAR) on WebLogic."

Deploying the Enterprise Archive (EAR) on WebLogic

Note:

The doc ear does not contain end-user documentation, such as context-sensitive help files. It contains only development-related documentation, including API Javadocs, ERDs, and XSDs and should not be deployed to a production server. Deploy the doc ear with the application ear on test or development environments. Do not deploy the doc ear on a production environment.

Sterling Selling and Fulfillment Foundation provides support for deploying Multiple EARs (Enterprise Archives) on a single application server. On the same application server, you can:

- Deploy different customizations of the same or different versions of the application, or
- Deploy different versions of the same application

Multiple EARs or context roots require additional memory for the application server JVM. Testing has shown that the deployment of a second IBM EAR file requires 2.5 - 3.5 times the memory of a single EAR. Supporting two deployments may require up to 2.5 GB of heap space and 1.2 GB of permanent space.

During installation, you can use JVM-specific arguments to avoid out-of-memory errors. For more information, see the *Sterling Selling and Fulfillment Foundation: Properties Guide* descriptions of ADDITIONAL_ANT_JAVA_TASK_ARGS and ADDITIONAL_ANT_COMPILER_TASK_ARGS.

For information about JVM tuning on your application server, see the general and application server-specific JVM chapters in the *Sterling Selling and Fulfillment Foundation: Performance Management Guide*.

For more information about how to implement multiple EAR files on the same application server, see the *Sterling Selling and Fulfillment Foundation: Customization Basics*.

For instructions on deploying the EAR to your WebLogic application server, see your WebLogic documentation.

Verifying Installation on WebLogic

About this task

To verify the Sterling Selling and Fulfillment Foundation installation:

Procedure

1. Restart your application server.
2. Start Internet Explorer.
3. To access the Application Console:
   a. Access `http://<hostname>:<port>/smcfs/console/login.jsp`.
   b. When prompted, enter your Login ID and Password.
4. To access IBM Sterling Business Center:
   a. Access the IBM Sterling Business Center login page by setting the enterprise appropriately. For more information about setting the enterprise while logging in to IBM Sterling Business Center, see "Setting an Enterprise for Logging In to IBM Sterling Business Center" on page 159.
   b. When prompted, enter your Login ID and Password.

Configuring a WebSphere Application Server - Overview

Before configuring WebSphere, IBM recommends that you start the WebSphere administrative server with the following memory parameters:

- -Xms768 MB or higher
- -Xmx768 MB or higher

Configuring WebSphere to Avoid Direct Datasource Lookup Warning Messages at Runtime

About this task

You have the option to avoid the warning messages regarding direct datasource lookups that occur at run time. To avoid these messages, do the following:

Procedure

1. From the WebSphere Administrative Console, expand **Troubleshooting** in the left panel and click on **Logs** and **Trace**.
2. Select each server that hosts Sterling Selling and Fulfillment Foundation and choose **Change Log Detail Levels** in the **General Properties**.
3. In the Components panel, select the class, **com.ibm.ejs.j2c.ConnectionFactoryBuilderImpl**, and specify the log level as **severe**.
4. Save the changes to the Master Configuration.

Configuring WebSphere to Avoid FileNotFound Exceptions

About this task

When launching an applet, you may see multiple FileNotFound exceptions in the WebSphere Application Server logs. These exceptions are harmless and you can safely ignore them. If they occur frequently and you want to avoid having them show up in the SystemOut.log file, set the custom property on the WebSphere Application Server Web Container to false, as follows:

```
com.ibm.ws.webcontainer.modifiedfilenotfoundexceptionbehavior=false
```

Configuring the WebSphere Classloader

About this task

Ensure that the WebSphere Classloader is set correctly for Classloader policy and Class loading modes as follows:

Procedure

1. From the Administrative Console left panel, choose Servers > Application Servers.
2. Select among the servers listed.
3. Set the Classloader policy pulldown to Single and the Class loading mode pulldown to Parent first.

Setting Up Application Clients to Invoke Sterling Selling and Fulfillment Foundation EJBs

About this task

In order to make EJB calls in Sterling Selling and Fulfillment Foundation using WebSphere you need to generate EJB stubs and skeletons. The following steps outline the method for creating the JAR files using the `ejbdeploy.sh` script to generate the stubs:

Procedure

1. Set the `CLASSPATH` to include `xercesImpl.jar`, `xalan.jar`, and `xml-apis.jar` as provided in the *JRE*`/lib/endorsed` directory. Also, `CLASSPATH` must include the jar files specified in the `dynamicclasspath.cfg` file.
2. Invoke the `ejbdeploy.sh` command from the <*WAS_HOME*>`/bin` directory with the following three arguments:
   a. Specify the full path to the `smcfsejb.jar` file in `<INSTALL_DIR>/external_deployments/` directory.
   b. Specify the temporary directory that is used for the EJB deployment.
   c. Specify the full path to the desired output file, for example `smcfs_ejbstubs.jar`.

Results

Additionally set the classpath on the `ejbdeploy.sh` command line following the `-cp` argument. For example:

```
$WAS_HOME/bin/ejbdeploy.sh <INSTALL_DIR>/external_deployments/smcfsejb.jar
WAS_HOME/temp <INSTALL_DIR>/external_deployments/smcfsejb.jar -cp
$CLASSPATH
```

Configuring WebSphere JVM Settings

About this task

You need to use the WebSphere Administrative Console to specify the JVM settings. These JVM settings must be set on **all** servers in a cluster (if you are using a cluster).

To configure JVM setting on WebSphere, do the following:

Procedure

1. From the WebSphere Administrative Console, select the application server specified for Sterling Selling and Fulfillment Foundation.
2. For IBM servers with IBM JDK 6.0 SR8:
   a. Select Server Infrastructure > Java and Process Management > Process Definition > Environment Entries.
   b. Choose `New` and specify the following values and then choose OK:

Name	Value	Description
PSALLOC	early	PSALLOC
NODISCLAIM	true	NODISCLAIM

3. Select Server Infrastructure > Java and Process Management > Process Definition > Java Virtual Machine. Edit the generic JVM arguments dialog to include the following values:

Property	Value	Description
-Dvendor	`-Dvendor=shell`	System Property. If you are using App Server Connection Pooling, use -Dvendor=websphere. Otherwise, use -Dvendor=shell.
-DvendorFile	`-DvendorFile=/servers.properties`	System property. Specify as an argument to the java command.

4. Under the Custom Properties section, set the JVM settings to the following values:

Name	Value	Description
client.encoding.override	UTF-8	Enables the use of special characters.

5. Restart the application server to enable these changes to take effect.
6. Save the changes to the Master Configuration.

Setting Up WebSphere to Display Barcodes and Graphs

About this task

Sterling Selling and Fulfillment Foundation uses the X Window functionality to display barcodes and dynamic graphical images (such as inventory supply & demand graphs) in a UNIX environment.

The following configuration is required to enable the X Window environment in UNIX systems for the WebSphere application server:

Procedure

1. From the WebSphere Administrative Console, go to Servers > Application Server and select the application server specified for Sterling Selling and Fulfillment Foundation.
2. On the Configuration tab, select Java and Process Management under Server Infrastructure option.
3. Select Process Definition.
4. On the configuration, go to Additional Properties and select Environment Entries.
5. Select New.
6. On the General Properties enter the Name as DISPLAY and the value as *IP_address_of_XWindows_server*:0.0. Do make sure that the X Window server accepts requests from this client.
7. If you are using UNIX, run the xhost+ command to remove access control for your X Window server.

   You can run X server on the same server in which you run Sterling Selling and Fulfillment Foundation. However, you need to be logged to the server console.

   Restart the application server for the DISPLAY variable to take effect.

8. Save the changes to the Master Configuration.

   Note: If the X Window server goes down or crashes while the inventory user interface is using the jbchartx.jar file, the WebSphere server also goes down.

Building the Enterprise Archive (EAR) Package on WebSphere

When deploying Sterling Selling and Fulfillment Foundation on WebSphere, use the `smcfs.ear` file, which may contain:

- `smcfs.war` - Web module that contains all of the Sterling Selling and Fulfillment Foundation JSPs and other Web application components.
- `sma.war` - Web module that contains the System Management Administrator application components.
- `sbc.war` - Web module that contains all of the IBM Sterling Business Center Web application components.
- `yantrawebservices.war` - Web module that contains all of the Sterling Selling and Fulfillment Foundation Web services interface classes.
- `smcfsejb.jar` - The EJB module that contains all the Sterling Selling and Fulfillment Foundation EJBs. You can pass an alternate earfile name by using the `–Dearfile` option to the `buildear.sh` script. Doing this will result in a name change for the ejb jar file. For example, if you specify an EAR file as `xyz.ear`, the ejb jar becomes `xyzejb.jar`.
- `smcfswsbe.jar` - The backend Web services jar file. You get this file if you expose Web services. You can pass an alternate earfile name to the ear build script. Doing this will result in a name change for the Web services backend jar file. For example, if you specify an EAR file as xyz.ear, the Web service backend jar becomes xyzwsbe.jar.
- jars that contain backend business logic
- jars that contain third-party libraries accessed by backend logic

Each of the third-party JAR files is left as is and in the manifest of the application each file is indicated as a dependency. For example, log4j files are represented separately as `log4j-1.2.15.jar` with a dependency in the application.

Building Web Services on WebSphere

You can expose APIs and synchronous SDF services through web services. Web services require some configuration prior to running the buildear script to create the application EAR and web services WAR file. The configuration takes place in either the namedwebservices.xml file or webservicebeans.xml file.

There are two types of web services that can be created:

- EJB: The EJB web service accepts two string inputs and returns one string output. No information about the content of the strings is included in the WSDL for this web service; the caller must open the javadocs for the API to get the information about how to structure the content. This makes it very difficult to dynamically generate calls to the XAPI web services without an additional source of information beyond the WSDL. This type of web service is created using EJB beans, and is created by default with the application EAR.
- JAX-WS: The WSDL of the JAX-WS web service includes information about the input expected by the server, the output, and exceptions, which makes it easier to dynamically generate calls to the web services. JAX-WS web services are created using JAXB beans. The advantage of using JAXB beans is that they are fully self-describing and alleviate the need to have access to javadocs for the

API. To facilitate securing JAX-WS based web services, they are created with handlers. You can use the default server and client handlers that are delivered with Sterling Selling and Fulfillment Suite or create your own, custom handlers.

You can choose to build the application with one or both types of web services.

Note: You can also choose to suppress the web services build by using the -Dnowebservice=true option on the buildear command.

Note: If using EJB based web services or using JAX-WS based web services without handlers and API security is enabled, ensure that you expose the Login API. If using custom handlers, this decision will depend upon how your handlers are written. For more information about web services security, see the *Sterling Selling and Fulfillment Foundation: Secure Deployment Guide*.

EJB Web Services Overview

An EJB web service accepts two string inputs and returns one string output. No information about the content of the strings is included in the WSDL for this web service; the caller must open the javadocs for the API to get the information about how to structure the content. This makes it very difficult to dynamically generate calls to the XAPI web services without an additional source of information beyond the WSDL. This type of web service is created using EJB beans, and is created by default with the application EAR.

Defining an EJB Web Service with WebSphere

Procedure

1. In the `<INSTALL_DIR>/repository/eardata/platform/webservices` folder, locate the namedwebservices.xml.sample file. Rename (or copy) it to namedwebservices.xml. This is the file you will edit.
2. In namedwebservices.xml, specify each API you want to expose as a web service in an Api/Name attribute.

   Property
   Description

   ServiceName
   The name of the service that you configured using the Service Definition Framework (SDF).

   ExposedName
   The name that is used in the Web Services Description Language (WSDL) file. This is the name that is used to call the web service programmatically. When specifying a service name for ExposedName, choose a literal that does not match any of the standard application API names.

   The exposed name must start with a lower case letter.
3. Set the properties necessary for your application server in sandbox.cfg:

   Parameter
   Description

   WAS_DIR
   Set the value to specify the <WAS_HOME> directory. (You may need to create the property.)

   Required if building EJB web services.

For WebSphere only

WEBSERVICES_BUILDS
Takes a comma-separated list that can include YIFWebService and SIXBeanXapiJaxWS. Defaults to YIFWebService.

Required for EJB and JAX-WS web services, on all application servers.

4. Save the file, navigate to the bin directory, and run the setupfiles command:

   Note: If you receive a "permission denied" error when performing this step, grant the build user write permission to the subdirectory referenced in the error message.
   - For UNIX/Linux: `<INSTALL_DIR>/bin/setupfiles.sh`
   - For Windows: `<INSTALL_DIR>\bin\setupfiles.cmd`
5. Create the EAR as described in "Creating the EAR File on WebSphere" on page 137. The web services defined in the file will be built when you create the application EAR.
6. Later, if you want to add more APIs and services as EJB Web services, repeat these steps.

About EJB Web Services and Security

How Sterling Selling and Fulfillment Suite Supports EJB Web Services Security

Transport Security

The concept of transport layer security is an option that can be applied to all types of http traffic including EJB based web services.

For YIFWebServices, use transport security. Sterling Selling and Fulfillment Suite does not provide any other security mechanisms or recommendations for EJB based web services.

See your application server documentation for information about security available on the application server itself.

Client Generation

Sterling Selling and Fulfillment Suite does not provide any assistance around client generation for YIFWebService. One popular pattern is to use the Axis client generation tasks for ant provide by Apache Axis. The same rules apply here. Be sure to configure the trust-store properly and set the required system properties in your code before communicating with the https port of the server.

JAX-WS Web Services Overview

You can expose APIs as web services. Before you create the application EAR file, you need to define which APIs will be exposed as web services so that they are included in the web services WAR file in the EAR. This section provides a high-level overview of how to configure a JAX-WS web service.

The following APIs are not supported with JAX-WS web services:
- createAsyncRequest
- evaluateAdvancedCondition
- evaluateCondition

- getAgentCriteriaList
- getPage
- invokeUE
- multiApi
- printDocumentSet

Performance Considerations

Injecting handlers into the web services stack can have performance implications. Customers should take care in designing handlers that the actions they perform will not be overly costly since these handlers will get invoked (potentially multiple times) for every API call that is made.

About JAX-WS Web Services Security

Sterling Selling and Fulfillment Suite provides:

- Information about ways to implement transport security for JAX-WS based web services - see the *Sterling Selling and Fulfillment Foundation: Secure Deployment Guide* sections about web services security for more information.
- JAX-WS handlers to implement security for JAX-WS based web services. Handler-chains are similar in concept to servlet filter chains. A default implementation is provided, but you can plug in your own customized handler chain xml and classes. See the *Sterling Selling and Fulfillment Foundation: Customizing Web Services* for more information.

Defining a JAX-WS Web Service with WebSphere

About this task

Procedure

1. Set the properties necessary for your application server in sandbox.cfg:

Parameter	Description
WEBSERVICES_BUILDS	Required. Can take a comma separated list that can include YIFWebService and SIXBeanXapiJaxWS.
XBEAN_PACKAGE	Optional. By default, the xbean package names and namespace generated for JAX-WS web services include the word "documentation." Use this parameter to replace the word "documentation" with another string. This value can be any string which would form a valid java package name. **Important:** Regardless of how it is entered, Sterling Selling and Fulfillment Foundation will always convert this string to all lowercase characters to comply with Java standards.
JAXB_LOCAL_SCOPING	Required. Valid values are true/false. Set to true. This flag affects the way that JAXB generates beans. Everything becomes “toplevel” so all classes will exist in the default package rather than existing in a hierarchy. Typically, for Sterling Selling and Fulfillment Foundation APIs, not using top-level scoping will result in generated file names that are too long for the file system.

Parameter	Description
JAXB_ALWAYS_ANNOTATE_CLASSNAMES	Required. The classes that are generated by JAXB are given “1-up” numbers at the end of their names to prevent having multiple classes with the same name when top level scoping is used. By default, this flag is included and set to true (yes, annotate all classes).If you prefer to minimize the use of one-up numbers, specify the AnnotateClassNames attribute in the webservicebeans.xml file for selected APIs, as shown in the following example. `<Bean BeanName="ParticipantBean" BeanPackage="com.sterlingcommerce.jaxws.participant. webservices" > <Apis> <Api AnnotateClassNames="true" Name="getOrganizationHierarchy" ExposedName="getOrganizationHierarchy" /> <Api AnnotateClassNames="true" Name="getOrganizationList" ExposedName="getOrganizationList" /> <Api AnnotateClassNames="false" Name="getPersonInfoList" ExposedName="getPersonInfoList" /> <Api AnnotateClassNames="false" Name="login" ExposedName="login" /> </Apis> </Bean>`
SUPPRESS_JAXWS_HANDLERS	By default, Sterling Selling and Fulfillment Suite applies handlers to JAX-WS based web services as part of security enablement. If you do not want the handlers applied, set this sandbox.cfg variable to true. For more information about web services security, see the *Sterling Selling and Fulfillment Foundation: Secure Deployment Guide*.

2. Save the file, navigate to the bin directory, and run the setupfiles command:

   Note: If you receive a "permission denied" error when performing this step, grant the build user write permission to the subdirectory referenced in the error message.

   - For UNIX/Linux: `<INSTALL_DIR>/bin/setupfiles.sh`
   - For Windows: `<INSTALL_DIR>\bin\setupfiles.cmd`

3. In the `<INSTALL_DIR>/repository/eardata/platform/webservices` folder, locate the `webservicebeans.xml.sample` file. Copy the file to `webservicebeans.xml`. This is the file you will edit.

webservicebeans.xml attributes	Description
Bean attributes **Note:** Do not put the same API into two beans.	

webservicebeans.xml attributes	Description
BeanName	Required. Enter a descriptive name for the web service. This name is used by the WSDL generator and reflected in the client code. Required. For example, a company named "Dave's BBQ" might use the following: `BeanName=DavesBBQJaxWS`
BeanPackage	Required. Enter a descriptive name for the package. Each bean must have a unique package name. This name is used by the WSDL generator and reflected in the client code. For example, a company named "Dave's BBQ" might use the following: `BeanPackage="com.DavesBBQ`
API element attributes	
Api Name	The name of the API that you configured using the Service Builder.
ExposedName	The name that is used in the Web Services Description Language (WSDL) file. This is the name that is used to call the web service programmatically. When specifying a service name for ExposedName, choose a literal that does not match any of the standard application API names. The exposed name must start with a lower case letter.
AnnotateClassNames	Specify the `AnnotateClassNames` attribute on the `webservicebeans.xml` to indicate which APIs need to have annotations. This flag provides a more granular way of indicating that 1-up annotations are required for specific APIs as compared to the sandbox variable ALWAYS_ANNOTATE_CLASS_NAMES.
Service element attributes	
Service Name	The name of the service that you configured using the Service Builder.
ExposedName	The name that is used in the Web Services Description Language (WSDL) file. This is the name that is used to call the web service programmatically. When specifying a service name for ExposedName, choose a literal that does not match any of the standard application service names. The exposed name must start with a lower case letter.
AnnotateClassName	Specify the `AnnotateClassNames` attribute on the `webservicebeans.xml` to indicate which APIs need to have annotations.
To expose an SDF service as a web service, you must also complete the procedure in “Exposing an SDF Service as a Web Service on WebSphere” on page 134.	

4. Save the file.
5. Create the EAR as described in “Creating the EAR File on WebSphere” on page 137. The web services defined in the file will be built when you create the application EAR.

   CAUTION: When building an ear with JAX-WS Web services, you may encounter out of memory errors. JAX-WS Web services will generate a java class for each element in the XSD, which can exceed the system memory. If the

ear build runs out of memory, then unnecessary APIs should be removed from the webservicebeans.xml file. Additionally, APIs with potentially large output XMLs may be exposed through services with reduced XSDs to eliminate unwanted elements.

Tip: To a certain extent, the memory requirements for building JAX-WS beans is on a bean-by-bean basis. This means that in some cases, exposing a set of APIs through one JAX-WS bean can cause an out-of-memory error during the ear build. Exposing the same APIs by splitting them across multiple beans may prevent the out-of-memory error and enable the ear to build successfully.

6. Later, if you want to add more APIs and services as JAX-WS web services, repeat these steps.

Example of webservicesbeans.xml file - WebSphere

The following is an example of a webservicebeans.xml file:

```
<WebServices>
  <Beans>
   <Bean BeanName="TheBeans"
BeanPackage="com.sterlingcommerce.jaxws.the.webservices" >
    <Apis>
      <Api Name="login" ExposedName="theLogin" />
      <Api Name="getLocaleList" ExposedName="theGetLocaleList" />
    </Apis>
   </Bean>
   <Bean BeanName="OtherBeans"
BeanPackage="com.sterlingcommerce.jaxws.other.webservices" >
    <Apis>
         <Api Name="getLocaleList" ExposedName="otherGetLocaleList" />
    </Apis>
     <Services>
      <Service AnnotateClassNames="true" Name="testWS" ExposedName="testWS" />
      <Service Name="testWS2" ExposedName="testWS2" />
    </Services>
   </Bean>
  </Beans>
</WebServices>
```

Exposing an SDF Service as a Web Service on WebSphere

About this task

To expose an SDF service as a web service, you first have to copy the input and output xsds for the API that is exposed in the SDF service and edit them.

Note: If you want to overwrite the input or output XSD for any API, you must expose the API as an SDF service and then expose the SDF service as a Web service. JAX-WS does not support API output templates but the equivalent can be achieved by using the following procedure.

Tip: You can use the same procedure to design fully customized APIs exposed through the SDF. To do so, write your own XSD for the input and output of your API, and ensure that you avoid constructs that are not supported in JAX-B such as xsd:any.

Procedure

1. Open the SDF and create a synchronous service that has an API node.
2. Select the desired API.
3. Save the SDF service.

4. Copy the input and output XSD files for the API you exposed in the SDF from the following location:

   `<INSTALL_DIR>/xapidocs/api_javadocs/XSD`

   to:

   `<INSTALL_DIR>/extensions/webservices`

5. Rename the files to match the name and exposed name that you give them in webservicebeans.xml.

   Use the following format:

   *<name>_<exposed_name>*_input.xsd

   *<name>_<exposed_name>*_output.xsd

Results

This is also the procedure you would use to extend a table, with this additional step: after copying and renaming the files, edit them to reference the new columns/API inputs/outputs.

Running the JAX-WS Client Generator on WebSphere

About this task

Sterling Selling and Fulfillment Foundation provides a client generator for use with JAX-WS web services on UNIX and Linux (the client generator is not supported on Windows). If you are modifying the XSD for any API or service which has been exposed as a web service, you must rebuild the web services client. Additionally, each time you extend the database through customization or upgrade, you must run the xsdgenerator, and then rebuild the ear and client. See *Sterling Selling and Fulfillment Foundation: Customizing APIs* for more information about using the xsdgenerator, and see *Sterling Selling and Fulfillment Foundation: Extending the Database* for more information about extending the database. See "Building the Enterprise Archive (EAR) Package on WebLogic" on page 113 for information about building the ear.

To generate a client against an HTTPS url, include the trustStore and keyStore system properties when you run the script.

Note: If you are using a SUN JDK or HP-UX JDK, create the file *install_dir* /jdk/jre/lib/stax.properties (or equvalent directory if sandbox.cfg variable *JAVA_HOME* points to an external directory.) Put the following line in stax.properties:

```
javax.xml.stream.XMLInputFactory=com.sun.xml.internal.stream.XMLInputFactoryImpl
```

Otherwise, you may see client build failures.

Procedure

To invoke the generator, run the buildjaxclient.sh script from the *install_dir* directory. Include the following system properties when running the script:

Property	Description
BEAN_NAME	Required. Matches the beanname on the server. The service name may differ, depending on the application server.
APPSERVER	Required. Valid values are websphere, weblogic, or jboss.

Property	Description
SERVER_URL	Required. The HTTP URL of the server up to, but not including, the bean name.
trustStore	Full path to your client truststore (required when generating client from the https url).
trustStorePassword	Password for your client truststore.
keyStore	Full path to your client keystore (required to achieve two-way ssl connection when generating a client from the https url).
keyStorePassword	Password for your client keystore.

The following code samples are examples of how the script might be set up in different scenarios.

- Without ssl:

```
bin/buildjaxclient.sh -DBEAN_NAME=ParticipantBeanService
-DAPPSERVER=websphere -DSERVER_URL=http://00.00.00.00:/SIXBeanXapiJaxWS
```

- For one way ssl:

```
bin/buildjaxclient.sh -DBEAN_NAME=ParticipantBeanService
-DAPPSERVER=websphere -DSERVER_URL=http://00.00.00.00:/SIXBeanXapiJaxWS
-DtrustStore=/home/joe_user/resources/ssl/jaxwsclienttrust.jks
-DtrustStorePassword=changeit
```

- For two way ssl:

```
bin/buildjaxclient.sh -DBEAN_NAME=ParticipantBeanService
-DAPPSERVER=websphere -DSERVER_URL=http://00.00.00.00:/SIXBeanXapiJaxWS
-DtrustStore=/home/joe_user/resources/ssl/jaxwsclienttrust.jks
-DtrustStorePassword=changeit -DkeyStore=/home/joe_user/resources/ssl/
jaxwsclientkey.jks -DkeyStorePassword=changeit
```

When run, the JAX-WS client generator creates a new folder, jaxwsclient, that contains the compiled classes and source code. The folder structure will be similar to the following: *install_dir*/jaxwsclient/output/*beanname*

Sample Code for BeanService Classes - WebSphere

Sample code is generated in the same directory as the bean and BeanService class. You can use this sample as a reference for invoking the bean methods.

The sample file has the same name as the service class but with "Sample" appended to the end of the name.

For example, SMCFSBeanService would have an accompanying sample called SMCFSBeanServiceSample class.

```
jaxwsclient/output/ParticipantBeanService/com/sterlingcommerce
/jaxws/participant/webservices/ParticipantBeanServiceSample
```

The class generates a method for each API you expose. The content will require editing.

```
public void testlogin( ) throws Exception {
com.sterlingcommerce.documentation.ycp.login.input.Login input
= new com.sterlingcommerce.documentation.ycp.login.input.Login();
        /*
Insert custom code here to set values on the input object.
        */
com.sterlingcommerce.documentation.ycp.login.output.Login
returnValue = b.login(env,input);
```

```
        /*
Insert custom code here to retrieve values from the return object.
        */
    }
```

Including Custom Classes on WebSphere

About this task

When deploying Sterling Selling and Fulfillment Foundation as Web services on WebSphere, if you have developed custom Java classes (user exits, event handlers, and so forth) you need to deploy them in order for them to be available.

To ensure that your custom classes get invoked, do the following:

Procedure

1. Create a JAR file with all your custom classes.
2. Place this JAR file in a folder structure based on the package name. For more information about packaging and deploying jar files, see the *Sterling Selling and Fulfillment Foundation: Customization Basics*.
3. Rebuild the EAR as described in "Creating the EAR File on WebSphere."

Results

The custom classes are automatically included in the `smcfs.ear` file.

Creating the EAR File on WebSphere

About this task

During the ear build, when JAX-WS webservice creation occurs, Sterling Selling and Fulfillment Suite will look for customer extensions and use them if they exist. If they do not exist, default handler XML and handler classes will be built into the ear and used. Sterling Selling and Fulfillment Suite provides one single set of default handlers that will get used against all beans that are deployed and for which customers have not provided extensions. Customer extensions can be provided on a per JAX-WS bean basis.

Enterprise Archives are built using an ANT (buildEAR.xml) that accepts the following targets:

Main Target
: **Description**

create-ear
: Creates `smcfs.ear` - The Sterling Selling and Fulfillment Foundation application EAR file.

Procedure

To create an application EAR file, run the following command from the `<INSTALL_DIR>/bin directory:`

```
   ./buildear.sh (.cmd for Windows) -Dappserver=websphere
-Dwarfiles=smcfs,sma,sbc -Dearfile=smcfs.ear create-ear
```

Note: Sterling Selling and Fulfillment Foundation supports the RPC encoded or document literal style and usage of invocation for Web services. When choosing style and usage for WebSphere, pass the following in the ear command line:

```
-D websphere-java2wsdl-style=<rpc|document>
```

Running this command creates the `smcfs.ear` file in the `<INSTALL_DIR>/external_deployments/` directory. It also puts three war files into the `smcfs.ear`:

- `smcfs.war` - The Sterling Selling and Fulfillment Foundation application war file
- `sma.war` - The System Management Administrator application war file
- `sbc.war` - The IBM Sterling Business Center application war file

Note: You can add the following options to the end of the above buildear commands:

- `-Dnowebservice=true` If you do not want to use Web services.
- `-Ddevmode=true` If you want to use the HTTP API Tester in the development environment.
- `-Dnodocear=true` If you want to skip the documentation build. Use this option when building the ear for a production environment. The doc ear does not contain end-user documentation, such as context-sensitive help files. It contains only development-related documentation, including API Javadocs, ERDs, and XSDs and should not be deployed to a production server.
- `-Dsupport.multi.war=true` If you want to copy all the UI jars to `<WAR>/WEB-INF/lib`. The UI jars will be copied based on the entry in `DCL.xml`.

For more information about the System Management Administrator (SMA) see the *Sterling Selling and Fulfillment Foundation: System Management and Administration Guide*. For more information about IBM Sterling Business Center, see the *Business Center: Item Administration Guide* and *Business Center: Pricing Administration Guide*.

For more information about WebSphere, see the IBM WebSphere information center: http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp.

Note:

If using WebSphere 8: When building EJB-based web services, if WebSphere 8 and Sterling Selling and Fulfillment Foundation were installed with different users, you may see file permissions errors when running the ear build script. To avoid this, grant permissions to read and write in the Sterling Selling and Fulfillment Foundation directory to the WebSphere 8 installation user.

The build ear process will leave files owned by the other user and you may need to set permissions back.

Precompiling the WAR File on WebSphere

To improve the performance when initially loading UI resources, IBM recommends that you precompile the jsps that comprise the WAR file. For more information about how to pre-compile jsps, see *"JSP Pre-Compilation"* in the *Sterling Selling and Fulfillment Foundation: Performance Management Guide*.

Deploying the Enterprise Archive (EAR) Using the WebSphere Admin Console

About this task

Note:

The doc ear does not contain end-user documentation, such as context-sensitive help files. It contains only development-related documentation, including API Javadocs, ERDs, and XSDs and should not be deployed to a production server. Deploy the doc ear with the application ear on test or development environments. Do not deploy the doc ear on a production environment.

Sterling Selling and Fulfillment Foundation provides support for deploying Multiple EARs (Enterprise Archives) on a single application server. On the same application server, you can:

- Deploy different customizations of the same or different versions of the application, or
- Deploy different versions of the same application

Multiple EARs or context roots require additional memory for the application server JVM. Testing has shown that the deployment of a second IBM EAR file requires 2.5 - 3.5 times the memory of a single EAR. Supporting two deployments may require up to 2.5 GB of heap space and 1.2 GB of permanent space.

During installation, you can use JVM-specific arguments to avoid out-of-memory errors. For more information, see the *Sterling Selling and Fulfillment Foundation: Properties Guide* descriptions of ADDITIONAL_ANT_JAVA_TASK_ARGS and ADDITIONAL_ANT_COMPILER_TASK_ARGS.

For information about JVM tuning on your application server, see the general and application server-specific JVM chapters in the *Sterling Selling and Fulfillment Foundation: Performance Management Guide*.

For more information about how to implement multiple EAR files on the same application server, see the *Sterling Selling and Fulfillment Foundation: Customization Basics*.

Note: Before deploying JAX-WS web services on WebSphere, note that the WebSphere server will attempt to inspect every jar in the WEB-INF lib by default. To avoid this behavior, which can add a significant amount of time to the deployment of the JAX-WS web services, configure the WebSphere server to ignore certain jars during deployment of the JAX-WS web service. Add any jars that the server should skip to the Ignore-Scanning-Archives section in <WAS_HOME>/properties/amm.filter.properties file. The jars you include will depend on what applications you build and so forth. One way to determine what jars to filter is to extract the SIXBean war file and use the jar -tvf command on it. Put the jar files you find in the WEB-INF/lib into the amm.filter.properties file.

To deploy the EAR on WebSphere:

Procedure

1. From the WebSphere Administrative Console menu in the left pane, select Applications > Application Types > WebSphere enterprise applications.

2. The right pane is populated with a list of applications that are deployed. Click the Install button.
3. Choose Local File System or Remote File System. Click the corresponding Browse button and browse to the Enterprise Archive such as `smcfs.ear` you want to deploy. Click Next.
4. Choose Fast Path option. Click Next.
5. Check Deploy enterprise beans, and change the application name as follows:
   - Ensure that there are no spaces in the application name; otherwise, the WebSphere-provided jsp compiler script will fail.
   - Ensure that the application name is different from that of the documentation EAR; otherwise, accepting the default makes both names the same.

   If you are using Web services, check Deploy WebServices.

   Note: If you want to precompile the JSP files during deployment, check Precompile JavaServer Pages files.

   Click Next.
6. The Map Modules to Servers screen displays. Select the checkbox next to each desired module (at least two entries, `smcfsejb.jar` and `smcfs.war`, should be present). Click the Cluster/Server in the Cluster and Server pane. Click **Apply**. The screen refreshes and the server field is updated with the chosen value. Click Next.
7. Accept the default JNDI names for the EJB modules on the Provide JNDI Names for Beans screen. Click Next.
8. On the Map Virtual Hosts for Web Modules screen, select your web module and its correct virtual host. Choose Next.
9. The Ensure all Unprotected 2.x Methods screen displays. Click Next.
10. The Provide Options to perform the WebServices Deployment screen displays. Leave them as is and click Next.
11. On the summary page, choose Finish.
12. If you are deploying the IBM Sterling Field Sales application, make the following additional changes in the WebSphere Administration Console:
    a. For each one of the application servers where you deploy the IBM Sterling Selling and Fulfillment Suite application, verify that the ClassLoader Policy is set to Multiple.
    b. Navigate to Enterprise Applications > Application Name > Class Loader. Set the Class Loader Order to "Classes loaded with local class loader first (parent last)."
    c. Navigate to Enterprise Applications > Application Name > Class Loader. Set the WAR Class Loader Policy to "Class Loader for each WAR file in application."

Verifying the Installation on WebSphere

About this task

To verify the Sterling Selling and Fulfillment Foundation installation:

Procedure

1. Restart your application server.
2. Start Internet Explorer.

3. To access the Application Console:
   a. Access `http://<hostname>:<port>/smcfs/console/login.jsp`.
   b. When prompted, enter your Login ID and Password.
4. To access IBM Sterling Business Center :
   a. Access the IBM Sterling Business Center login page by setting the enterprise appropriately. For more information about setting the enterprise while logging in to IBM Sterling Business Center, see "Setting an Enterprise for Logging In to IBM Sterling Business Center" on page 159.
   b. When prompted, enter your Login ID and Password.

Configuring a JBoss Application Server

About this task

Note: The JBoss server must have the default name of "all" for the precompilation scripts to run successfully.

To set up the JBoss application server, you must set up some properties in the JBoss script file.

To set up the JBoss script file, do the following:

Procedure

1. Add the following properties to the `<JBOSS_HOME>/bin/run.conf` file supplied by JBoss. Each property and its proper syntax are described in the following list:

   Property
   Required Edits

   JAVA_OPTS
   Depending on your JVM vendor, specify as follows:
   ```
   -Xms<value> -Xmx<value>
   ```
   For example, for HP UX 11i on Itanium, set this value to `-XX:MaxPermSize=512m -Xms768m -Xmx768m`

   For information about supported JDK tiers and memory requirements for specific operating systems, go to the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning.

   -Dfile.encoding
   To ensure that all the Sterling Selling and Fulfillment Foundation screens display UTF-8 characters for java commands, specify:
   ```
   -Dfile.encoding=UTF-8
   ```
   This is applicable to all the Sterling Selling and Fulfillment Foundation Java start-up scripts.

   -Dvendor
   System Property. If you are using App Server Connection Pooling, use -Dvendor=jboss. Otherwise, use -Dvendor=shell.
   ```
   -Dvendor=shell
   ```

   -DvendorFile
   System property. Specify as an argument to the java command.
   ```
   -DvendorFile=/servers.properties
   ```

The -D properties shown above can either be included in JAVA_OPT$or passed in a command line. For example:

```
run.sh -c yantra_domain
-Djboss.partition.name=yantraPartition -b host
-Dvendor=shell -DvendorFile=/servers.properties
-DLOGFILE=logs/JBOSS_sci.log
-DSECURITY_LOGFILE=logs/JBOSS_security.log
```

2. If you are using an HTTPS transport, download the Secure Socket Extension (JSSE) 1.0.3 package from http://java.sun.com and add the following files to the <*JAVA_HOME*>/jre/lib/extn/ directory:
   - jnet.jar
   - jcert.jar
   - jsse.jar

   In addition to this setup, see "Setting Up JBoss to Display Barcodes and Graphs" for information about setting up the JBoss application server to display barcodes and graphs.

Setting Up JBoss to Display Barcodes and Graphs

About this task

Sterling Selling and Fulfillment Foundation uses the X Window functionality to display barcodes and dynamic graphical images (such as inventory supply & demand graphs) in a UNIX environment. The following configuration is required to enable the X Window environment in UNIX systems for JBoss application servers:

Procedure

1. If your UNIX server is also an X Window client, edit the run.sh script, and set the DISPLAY environment variable as follows:

   ```
   export DISPLAY=<IP_address_of_XWindows_server>:0.0
   ```

2. If you are using UNIX, run the xhost + command to remove access control for your X Window server.
3. You can run X-server on the same server in which you run Sterling Selling and Fulfillment Foundation. However, you need to be logged in to the server console.

   Note: If the X Window server goes down or crashes while the inventory user interface is using the jbchartx.jar file, the JBoss server also goes down.

Building the Enterprise Archive (EAR) Package on JBoss

When deploying Sterling Selling and Fulfillment Foundation on JBoss, use the smcfs.ear file, which may contain:

- smcfs.war - Web module that contains all of the Sterling Selling and Fulfillment Foundation JSPs and other Web application components.
- sma.war - Web module that contains the System Management Administrator application components.
- sbc.war - Web module that contains all of the IBM Sterling Business Center Web application components.
- smcfsejb.jar - The EJB module that contains all the Sterling Selling and Fulfillment Foundation EJBs. You can pass an alternate earfile name by using the

–Dearfile option to the buildear.sh script. Doing this will result in a name change for the ejb jar file. For example, if you specify an EAR file as xyz.ear, the ejb jar becomes xyzejb.jar.

- smcfswsbe.jar - The backend Web services jar file. You get this file if you expose Web services. You can pass an alternate earfile name to the ear build script. Doing this will result in a name change for the Web services backend jar file. For example, if you specify an EAR file as xyz.ear, the Web services backend jar becomes xyzwsbe.jar.
- Jars that contain backend business logic.
- Jars that contain third-party libraries accessed by backend logic.

Each of the third-party JAR files are left as is and in the manifest of the application each file is indicated as a dependency. For example, log4j files are represented separately as log4j-1.2.15.jar with a dependency in the application.

Building Web Services on JBoss

You can expose APIs and synchronous SDF services through web services. Web services require some configuration prior to running the buildear script to create the application EAR and web services WAR file. The configuration takes place in either the namedwebservices.xml file or webservicebeans.xml file.

There are two types of web services that can be created:

- EJB: The EJB web service accepts two string inputs and returns one string output. No information about the content of the strings is included in the WSDL for this web service; the caller must open the javadocs for the API to get the information about how to structure the content. This makes it very difficult to dynamically generate calls to the XAPI web services without an additional source of information beyond the WSDL. This type of web service is created using EJB beans, and is created by default with the application EAR.
- JAX-WS: The WSDL of the JAX-WS web service includes information about the input expected by the server, the output, and exceptions, which makes it easier to dynamically generate calls to the web services. JAX-WS web services are created using JAXB beans. The advantage of using JAXB beans is that they are fully self-describing and alleviate the need to have access to javadocs for the API. To facilitate securing JAX-WS based web services, they are created with handlers. You can use the default server and client handlers that are delivered with Sterling Selling and Fulfillment Suite or create your own, custom handlers.

You can choose to build the application with one or both types of web services.

Note: If using EJB based web services or using JAX-WS based web services without handlers and API security is enabled, ensure that you expose the Login API. If using custom handlers, this decision will depend upon how your handlers are written. For more information about web services security, see the *Sterling Selling and Fulfillment Foundation: Secure Deployment Guide*.

EJB Web Services Overview

An EJB web service accepts two string inputs and returns one string output. No information about the content of the strings is included in the WSDL for this web service; the caller must open the javadocs for the API to get the information about how to structure the content. This makes it very difficult to dynamically generate calls to the XAPI web services without an additional source of information beyond the WSDL. This type of web service is created using EJB beans, and is created by default with the application EAR.

Defining an EJB Web Service with JBoss

Procedure

1. In the <INSTALL_DIR>/repository/eardata/platform/webservices folder, locate the namedwebservices.xml.sample file. Rename (or copy) it to namedwebservices.xml. This is the file you will edit.
2. In namedwebservices.xml, specify each API you want to expose as a web service in an Api/Name attribute.

   Property
   Description

   ServiceName
   The name of the service that you configured using the Service Definition Framework (SDF).

   ExposedName
   The name that is used in the Web Services Description Language (WSDL) file. This is the name that is used to call the web service programmatically. When specifying a service name for ExposedName, choose a literal that does not match any of the standard application API names.

   The exposed name must start with a lower case letter.
3. Set the properties necessary for your application server in sandbox.cfg:

   Parameter
   Description

   EJB_3_ENABLED
   Set to true or false. Determines whether the EJBs are generated according to the spec version 2 or 3. JBoss supports both.

   Required if building EJB web services (JBoss only).

   JBOSS_PRECOMPILE_JSP
   Precompiles pages in the WAR file.

   Set to true (precompile jsps) or false (do not precompile jsps). There is no default set by installation, but jsps will not be precompiled unless you set this to true.

   Required if building EJB web services.

   For JBoss only.

   JBOSS_DIR
   Set to the absolute path of the JBoss installation directory.

   Required if building EJB web services.

   For JBoss only.

   WEBSERVICES_BUILDS
   Takes a comma-separated list that can include YIFWebService and SIXBeanXapiJaxWS. Defaults to YIFWebService.

   Required for EJB and JAX-WS web services, on all application servers.
4. Save the file, navigate to the bin directory, and run the setupfiles command:
   - For UNIX/Linux: <INSTALL_DIR>/bin/setupfiles.sh
   - For Windows: <INSTALL_DIR>\bin\setupfiles.cmd

5. Create the EAR as described in "Creating the EAR on JBoss" on page 152. The web services defined in the file will be built when you create the application EAR.
6. Later, if you want to add more APIs and services as EJB Web services, repeat these steps.

About EJB Web Services and Security

How Sterling Selling and Fulfillment Suite Supports EJB Web Services Security

Transport Security

The concept of transport layer security is an option that can be applied to all types of http traffic including EJB based web services.

For YIFWebServices, use transport security. Sterling Selling and Fulfillment Suite does not provide any other security mechanisms or recommendations for EJB based web services.

See your application server documentation for information about security available on the application server itself.

Client Generation

Sterling Selling and Fulfillment Suite does not provide any assistance around client generation for YIFWebService. One popular pattern is to use the Axis client generation tasks for ant provide by Apache Axis. The same rules apply here. Be sure to configure the trust-store properly and set the required system properties in your code before communicating with the https port of the server.

JAX-WS Web Services Overview

You can expose APIs as web services. Before you create the application EAR file, you need to define which APIs will be exposed as web services so that they are included in the web services WAR file in the EAR. This section provides a high-level overview of how to configure a JAX-WS web service.

The following APIs are not supported with JAX-WS web services:

- createAsyncRequest
- evaluateAdvancedCondition
- evaluateCondition
- getAgentCriteriaList
- getPage
- invokeUE
- multiApi
- printDocumentSet

Performance Considerations

Injecting handlers into the web services stack can have performance implications. Customers should take care in designing handlers that the actions they perform will not be overly costly since these handlers will get invoked (potentially multiple times) for every API call that is made.

About JAX-WS Web Services Security

Sterling Selling and Fulfillment Suite provides:

- Information about ways to implement transport security for JAX-WS based web services - see the *Sterling Selling and Fulfillment Foundation: Secure Deployment Guide* sections about web services security for more information.
- JAX-WS handlers to implement security for JAX-WS based web services. Handler-chains are similar in concept to servlet filter chains. A default implementation is provided, but you can plug in your own customized handler chain xml and classes. See the *Sterling Selling and Fulfillment Foundation: Customizing Web Services* for more information.

Defining a JAX-WS Web Service on JBoss

Procedure

1. Set the properties necessary for your application server in sandbox.cfg:

Parameter	Description
WEBSERVICES_BUILDS	Required. Can take a comma separated list that can include YIFWebService and SIXBeanXapiJaxWS.
XBEAN_PACKAGE	Optional. By default, the xbean package names and namespace generated for JAX-WS web services include the word "documentation." Use this parameter to replace the word "documentation" with another string. This value can be any string which would form a valid java package name. **Important:** Regardless of how it is entered, Sterling Selling and Fulfillment Foundation will always convert this string to all lowercase characters to comply with Java standards.
JAXB_LOCAL_SCOPING	Required. Valid values are true/false. Set to true. This flag affects the way that JAXB generates beans. Everything becomes “toplevel” so all classes will exist in the default package rather than existing in a hierarchy. Typically, for Sterling Selling and Fulfillment Foundation APIs, not using top-level scoping will result in generated file names that are too long for the file system.

Parameter	Description
JAXB_ALWAYS_ANNOTATE_CLASSNAMES	Required. The classes that are generated by JAXB are given "1-up" numbers at the end of their names to prevent having multiple classes with the same name when top level scoping is used. By default, this flag is included and set to true (yes, annotate all classes).If you prefer to minimize the use of one-up numbers, specify the AnnotateClassNames attribute in the webservicebeans.xml file for selected APIs, as shown in the following example. `<Bean BeanName="ParticipantBean" BeanPackage="com.sterlingcommerce.jaxws.participant. webservices" > <Apis> <Api AnnotateClassNames="true" Name="getOrganizationHierarchy" ExposedName="getOrganizationHierarchy" /> <Api AnnotateClassNames="true" Name="getOrganizationList" ExposedName="getOrganizationList" /> <Api AnnotateClassNames="false" Name="getPersonInfoList" ExposedName="getPersonInfoList" /> <Api AnnotateClassNames="false" Name="login" ExposedName="login" /> </Apis> </Bean>`
SUPPRESS_JAXWS_HANDLERS	By default, Sterling Selling and Fulfillment Suite applies handlers to JAX-WS based web services as part of security enablement. If you do not want the handlers applied, set this sandbox.cfg variable to true. For more information about web services security, see the *Sterling Selling and Fulfillment Foundation: Secure Deployment Guide*.

2. Save the file, navigate to the bin directory, and run the setupfiles command:
   - For UNIX/Linux: `<INSTALL_DIR>/bin/setupfiles.sh`
   - For Windows: `<INSTALL_DIR>\bin\setupfiles.cmd`
3. In the `<INSTALL_DIR>/repository/eardata/platform/webservices` folder, locate the `webservicebeans.xml.sample` file. Copy the file to `webservicebeans.xml`. This is the file you will edit.

webservicebeans.xml attributes	Description
Bean attributes **Note:** Do not put the same API into two beans.	
BeanName	Required. Enter a descriptive name for the web service. This name is used by the WSDL generator and reflected in the client code. Required. For example, a company named "Dave's BBQ" might use the following: `BeanName=DavesBBQJaxWS`

webservicebeans.xml attributes	Description
BeanPackage	Required. Required. Enter a descriptive name for the package. Each bean must have a unique package name. This name is used by the WSDL generator and reflected in the client code. For example, a company named "Dave's BBQ" might use the following: `BeanPackage="com.DavesBBQ`
API element attributes	
Api Name	The name of the API that you configured using the Service Builder.
ExposedName	The name that is used in the Web Services Description Language (WSDL) file. This is the name that is used to call the web service programmatically. When specifying a service name for ExposedName, choose a literal that does not match any of the standard application API names. The exposed name must start with a lower case letter.
AnnotateClassNames	Specify the `AnnotateClassNames` attribute on the `webservicebeans.xml` to indicate which APIs need to have annotations. This flag provides a more granular way of indicating that 1-up annotations are required for specific APIs as compared to the sandbox variable ALWAYS_ANNOTATE_CLASS_NAMES.
Service element attributes	
Service Name	The name of the service that you configured using the Service Builder.
ExposedName	The name that is used in the Web Services Description Language (WSDL) file. This is the name that is used to call the web service programmatically. When specifying a service name for ExposedName, choose a literal that does not match any of the standard application service names. The exposed name must start with a lower case letter.
AnnotateClassName	Specify the `AnnotateClassNames` attribute on the `webservicebeans.xml` to indicate which APIs need to have annotations.
To expose an SDF service as a web service, you must also complete the procedure in "Exposing an SDF Service as a Web Service on JBoss" on page 149.	

4. Save the file.
5. Create the EAR as described in "Creating the EAR on JBoss" on page 152. The web services defined in the file will be built when you create the application EAR.

   CAUTION:
   When building an ear with JAX-WS web services, you may encounter out of memory errors. JAX-WS web services will generate a java class for each element in the XSD, which can exceed the system memory. If the ear build runs out of memory, then unnecessary APIs should be removed from the webservicebeans.xml file. Additionally, APIs with potentially large output XMLs may be exposed through services with reduced XSDs to eliminate unwanted elements.

Tip: To a certain extent, the memory requirements for building JAX-WS beans is on a bean-by-bean basis. This means that in some cases, exposing a set of APIs through one JAX-WS bean can cause an out-of-memory error during the ear build. Exposing the same APIs by splitting them across multiple beans may prevent the out-of-memory error and enable the ear to build successfully.

6. Later, if you want to add more APIs and services as JAX-WS web services, repeat these steps.

Example of webservicesbeans.xml file - JBoss

The following is an example of a webservicebeans.xml file:

```
<WebServices>
  <Beans>
   <Bean BeanName="TheBeans"
BeanPackage="com.sterlingcommerce.jaxws.the.webservices" >
    <Apis>
      <Api Name="login" ExposedName="theLogin" />
      <Api Name="getLocaleList" ExposedName="theGetLocaleList" />
    </Apis>
   </Bean>
   <Bean BeanName="OtherBeans"
BeanPackage="com.sterlingcommerce.jaxws.other.webservices" >
    <Apis>
         <Api Name="getLocaleList" ExposedName="otherGetLocaleList" />
    </Apis>
     <Services>
      <Service AnnotateClassNames="true" Name="testWS" ExposedName="testWS" />
      <Service Name="testWS2" ExposedName="testWS2" />
    </Services>
   </Bean>
  </Beans>
</WebServices>
```

Exposing an SDF Service as a Web Service on JBoss

About this task

To expose an SDF service as a web service, you first have to copy the input and output xsds for the API that is exposed in the SDF service and edit them.

Note: If you want to overwrite the input or output XSD for any API, you must expose the API as an SDF service and then expose the SDF service as a Web service. JAX-WS does not support API output templates but the equivalent can be achieved by using the following procedure.

Tip: You can use the same procedure to design fully customized APIs exposed through the SDF. To do so, write your own XSD for the input and output of your API, and ensure that you avoid constructs that are not supported in JAX-B such as xsd:any.

Procedure

1. Open the SDF and create a synchronous service that has an API node.
2. Select the desired API.
3. Save the SDF service.
4. Copy the input and output XSD files for the API you exposed in the SDF from the following location:

   `<INSTALL_DIR>/xapidocs/api_javadocs/XSD`

   to:

<INSTALL_DIR>/extensions/webservices

5. Rename the files to match the name and exposed name that you give them in webservicebeans.xml.

   Use the following format:

   <*name*>_<*exposed_name*>_input.xsd

   <*name*>_<*exposed_name*>_output.xsd

Results

This is also the procedure you would use to extend a table, with this additional step: after copying and renaming the files, edit them to reference the new columns/API inputs/outputs.

Running the JAX-WS Client Generator on JBoss

About this task

Sterling Selling and Fulfillment Foundation provides a client generator for use with JAX-WS web services on UNIX and Linux (the client generator is not supported on Windows). If you are modifying the XSD for any API or service which has been exposed as a web service, you must rebuild the web services client. Additionally, each time you extend the database through customization or upgrade, you must run the xsdgenerator, and then rebuild the ear and client. See *Sterling Selling and Fulfillment Foundation: Customizing APIs* for more information about using the xsdgenerator, and see *Sterling Selling and Fulfillment Foundation: Extending the Database* for more information about extending the database. See "Building the Enterprise Archive (EAR) Package on JBoss" on page 142 for information about building the ear.

To generate a client against an HTTPS url, include the trustStore and keyStore system properties when you run the script.

Note: If you are using a SUN JDK or HP-UX JDK, create the file *install_dir* /jdk/jre/lib/stax.properties (or equvalent directory if sandbox.cfg variable *JAVA_HOME* points to an external directory.) Put the following line in stax.properties:

```
javax.xml.stream.XMLInputFactory=com.sun.xml.internal.stream.XMLInputFactoryImpl
```

Otherwise, you may see client build failures.

Procedure

To invoke the generator, run the buildjaxclient.sh script from the *install_dir* directory. Include the following system properties when running the script:

Property	Description
BEAN_NAME	Required. Matches the beanname on the server. The service name may differ, depending on the application server.
APPSERVER	Required. Valid values are websphere, weblogic, or jboss.
SERVER_URL	Required. The HTTP URL of the server up to, but not including, the bean name.
trustStore	Full path to your client truststore (required when generating client from the https url).

Property	Description
trustStorePassword	Password for your client truststore.
keyStore	Full path to your client keystore (required to achieve two-way ssl connection when generating a client from the https url).
keyStorePassword	Password for your client keystore.

The following code samples are examples of how the script might be set up in different scenarios.

- Without ssl:

```
bin/buildjaxclient.sh -DBEAN_NAME=ParticipantBeanService
-DAPPSERVER=jboss -DSERVER_URL=http://00.00.00.00:/SIXBeanXapiJaxWS
```

- For one way ssl:

```
bin/buildjaxclient.sh -DBEAN_NAME=ParticipantBeanService
-DAPPSERVER=jboss -DSERVER_URL=http://00.00.00.00:/SIXBeanXapiJaxWS
-DtrustStore=/home/joe_user/resources/ssl/jaxwsclienttrust.jks
-DtrustStorePassword=changeit
```

- For two way ssl:

```
bin/buildjaxclient.sh -DBEAN_NAME=ParticipantBeanService
-DAPPSERVER=jboss -DSERVER_URL=http://00.00.00.00:/SIXBeanXapiJaxWS
-DtrustStore=/home/joe_user/resources/ssl/jaxwsclienttrust.jks
-DtrustStorePassword=changeit -DkeyStore=/home/joe_user/resources/ssl/
jaxwsclientkey.jks -DkeyStorePassword=changeit
```

When run, the JAX-WS client generator creates a new folder, jaxwsclient, that contains the compiled classes and source code. The folder structure will be similar to the following: *install_dir*/jaxwsclient/output/*beanname*

Sample Code for BeanService Classes - JBoss

Sample code is generated in the same directory as the bean and BeanService class. You can use this sample as a reference for invoking the bean methods.

The sample file has the same name as the service class but with "Sample" appended to the end of the name.

For example, SMCFSBeanService would have an accompanying sample called SMCFSBeanServiceSample class.

```
jaxwsclient/output/ParticipantBeanService/com/sterlingcommerce
/jaxws/participant/webservices/ParticipantBeanServiceSample
```

The class generates a method for each API you expose. The content will require editing.

```
public void testlogin( ) throws Exception {
com.sterlingcommerce.documentation.ycp.login.input.Login input
= new com.sterlingcommerce.documentation.ycp.login.input.Login();
        /*
Insert custom code here to set values on the input object.
        */
com.sterlingcommerce.documentation.ycp.login.output.Login
returnValue = b.login(env,input);
        /*
Insert custom code here to retrieve values from the return object.
        */
    }
```

Including Custom Classes on JBoss

About this task

When deploying Sterling Selling and Fulfillment Foundation as Web service on JBoss, if you have developed custom Java classes (user exits, event handlers, and so forth) you need to deploy them in order for them to be available.

To ensure that your custom classes get invoked, do the following:

Procedure

1. Create a JAR file with all your custom classes.
2. Place this JAR file in a folder structure based on the package name. For more information about packaging and deploying jar files, see the *Sterling Selling and Fulfillment Foundation: Customization Basics*.
3. Rebuild the EAR as described in "Creating the EAR on JBoss."

Results

These classes are automatically included in the `smcfs.ear` file.

Creating the EAR on JBoss

About this task

During the ear build, when JAX-WS webservice creation occurs, Sterling Selling and Fulfillment Suite will look for customer extensions and use them if they exist. If they do not exist, default handler XML and handler classes will be built into the ear and used. Sterling Selling and Fulfillment Suite provides one single set of default handlers that will get used against all beans that are deployed and for which customers have not provided extensions. Customer extensions can be provided on a per JAX-WS bean basis.

Enterprise Archives are built using an ANT (buildEAR.xml) that accepts the following targets:

Main Target
: **Description**

create-ear
: Creates `smcfs.ear` - the Sterling Selling and Fulfillment Foundation application EAR file.

Note: To successfully build the application EAR file in 64-bit JBoss, use the following memory parameters:

```
-XX:MaxPermSize=768m -Xmx2048m -Xms2048m
```

You can set these memory parameters during installation or in the sandbox.cfg file after installation. For more information about setting them during installation, see the topics about running interactive and silent installations. For setting them after installation, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.

Procedure

To create an application EAR file, run the following command from the `<INSTALL_DIR>/bin directory:`

```
./buildear.sh (.cmd for Windows) -Dappserver=jboss
-Dwarfiles=smcfs,sma,sbc -Dearfile=smcfs.ear create-ear
```

Note: Sterling Selling and Fulfillment Foundation supports the RPC encoded or document literal style and usage of invocation for Web services. When choosing style and usage for JBoss, pass the following in the ear command line:

```
-Djboss-java2wsdl-style=<rpc|document>
```

Running this command creates the `smcfs.ear` file in the `<INSTALL_DIR>/external_deployments/` directory. It also puts three war files into the `smcfs.ear`:

- `smcfs.war` - the Sterling Selling and Fulfillment Foundation application war file
- `sma.war` - the System Management Administrator application war file
- `sbc.war` - the IBM Sterling Business Center application war file

  Note: You can add the following options to the end of the above buildear commands:
  - `-Dnowebservice=true` if you do not want to use Web services.
  - `-Ddevmode=true` if you want to use the HTTP API Tester in the development environment.
  - `-Dnodocear=true` if you want to skip the documentation build. Use this option when building the ear for a production environment. The doc ear does not contain end-user documentation, such as context-sensitive help files. It contains only development-related documentation, including API Javadocs, ERDs, and XSDs that should not be deployed to a production server.
  - `-Dsupport.multi.war=true` if you want to copy all the UI jars to `<WAR>/WEB-INF/lib`. The UI jars will be copied based on the entry in `DCL.xml`.

Results

For more information about the System Management Administrator (SMA) see the *Sterling Selling and Fulfillment Foundation: System Management and Administration Guide*. For more information about IBM Sterling Business Center, see the *Business Center: Item Administration Guide* and *Business Center: Pricing Administration Guide*.

For more information about JBoss, see the documentation provided by the vendor.

Precompiling the WAR File on JBoss

See the *Sterling Selling and Fulfillment Foundation: Performance Management Guide* for instructions on how to precompile the WAR on JBoss. There are settings that must be configured before you create the EAR file.

Deploying the Enterprise Archive (EAR) on JBoss

Note:

The doc ear does not contain end-user documentation, such as context-sensitive help files. It contains only development-related documentation, including API Javadocs, ERDs, and XSDs and should not be deployed to a production server. Deploy the doc ear with the application ear on test or development environments. Do not deploy the doc ear on a production environment.

Sterling Selling and Fulfillment Foundation provides support for deploying Multiple EARs (Enterprise Archives) on a single application server. On the same application server, you can:

- Deploy different customizations of the same or different versions of the application, or
- Deploy different versions of the same application

Multiple EARs or context roots require additional memory for the application server JVM. Testing has shown that the deployment of a second IBM EAR file requires 2.5 - 3.5 times the memory of a single EAR. Supporting two deployments may require up to 2.5 GB of heap space and 1.2 GB of permanent space.

During installation, you can use JVM-specific arguments to avoid out-of-memory errors. For more information, see the *Sterling Selling and Fulfillment Foundation: Properties Guide* descriptions of ADDITIONAL_ANT_JAVA_TASK_ARGS and ADDITIONAL_ANT_COMPILER_TASK_ARGS.

For information about JVM tuning on your application server, see the general and application server-specific JVM chapters in the *Sterling Selling and Fulfillment Foundation: Performance Management Guide*.

For more information about how to implement multiple EAR files on the same application server, see the *Sterling Selling and Fulfillment Foundation: Customization Basics*.

Deploy your newly created `smcfs.ear` file as described in your application server documentation, using the `deployEARJboss.xml` ant script.

1. Stop the application server.
2. Copy the ear file to the deployment directory on the application server (the JBoss installation directory is <servername>/deploy).
3. Restart the application server.
4. Log in.

To verify the Sterling Selling and Fulfillment Foundation installation:

1. Restart your application server.
2. Start Internet Explorer.
3. To access the Application Console:
   a. Access `http://<hostname>:<port>/smcfs/console/login.jsp`.
   b. When prompted, enter your Login ID and Password.
4. To access IBM Sterling Business Center:
   a. Access the IBM Sterling Business Center login page by setting the enterprise appropriately. For more information about setting the enterprise while logging in to IBM Sterling Business Center, see "Setting an Enterprise for Logging In to IBM Sterling Business Center" on page 159.
   b. When prompted, enter your Login ID and Password.

Configuring DataSource Connection Pooling on WebLogic, WebSphere, and JBoss

You can specify an external datasource and connection pool instead of the default connection pooling supplied with Sterling Selling and Fulfillment Foundation to manage your database connections. The configuration process includes these high-level tasks:

- Create and configure any datasources and connection pools necessary on your application server.
- Define the datasource in Sterling Selling and Fulfillment Foundation using the jdbc datasource parameter.
- Edit the -dvendor option in the application server configuration. This is done in the application server software.

Configuring Data Source Connection Pooling for Single Schema Installations

About this task

To configure a datasource and connection pool for an instance of Sterling Selling and Fulfillment Foundation that was installed to use a single database schema:

Procedure

1. Create the connection pool and datasource on your application server. See your application server documentation for specific configuration and operating instructions.
2. Install Sterling Selling and Fulfillment Foundation.
3. In a text editor, open (or create) the customer_overrides.properties file in the *INSTALL*\properties folder.
4. In the customer_overrides.properties, add the line for your database from the following list:
   - For DB2, add `jdbcService.db2Pool.datasource=`*`datasourceName`*
   - For Oracle, add `jdbcService.oraclePool.datasource=`*`datasourceName`*

   where *datasourceName* is the logical name of the external datasource.
5. Save the customer_overrides.properties file and run setupfiles.sh/cmd.
6. Build and deploy the EAR.
7. Update the application server configuration to use `-Dvendor=`*`appserver`*
   – where the value of *appserver* is weblogic, jboss, or websphere.
8. Start the application server.

Results

For more information about `-Dvendor` and other properties in the application server startup scripts, see “Configuring an Oracle WebLogic Application Server - Overview” on page 109, “Configuring a WebSphere Application Server - Overview” on page 125, or “Configuring a JBoss Application Server” on page 141.

Configuring Data Source Connection Pooling for Multischema Installations

About this task

In a multischema installation, you can have four types of schema: Metadata, Statistics, System Configuration, and Transaction/Master.

- You must configure a datasource on your application server for each schema and colony you have. For example, you could have a datasource named metadata_ds for the Metadata schema, a datasource named config_ds for the System Configuration schema, and so on.
- When multiple schemas are defined, the schemas must all use either datasource or direct JDBC calls: they cannot use both. Sterling Selling and Fulfillment Foundation does not support 2-mode (datasource and direct JDBC calls). For example, you could not have a schema called "DEFAULT_METADATA" use a datasource pool, and a schema called "DEFAULT_BI_STAGING" use a direct JDBC call. They would both have to use the same method.
- Datasources for the Statistics, System Configuration, and Transaction/Master schema can be configured using either the System Management Administrator (SMA) or the manageDBpool API. To configure the datasource for a Metadata schema, you must use the manageDBpool API.
- At least three data sources must be specified for multischema installations.
- If not set, the datasource name defaults to Pool ID.
- For more information about multischema database parameters, see "About Running a Silent Installation in Multischema Mode" on page 52

To configure datasources for a multischema installation:

Procedure

1. Create the connection pools and datasources for each schema on your application server. See your application server documentation for specific configuration and operating instructions.
2. Install the application.
3. In a text editor, open (or create) the `customer_overrides.properties` file in the *INSTALL*\properties folder.
4. In customer_overrides, add the line for your database from the following list:
   - For DB2, add `jdbcService.db2Pool.datasource=`*`datasourceName`*
   - For Oracle, add `jdbcService.oraclePool.datasource=`*`datasourceName`*

   where *`datasourceName`* is the logical name of the external datasource.
5. Save the customer_overrides file and run setupfiles.sh/cmd.
6. Use the System Management Administrator (SMA) or manageDBpool API to add the datasource attribute to the selected pools.

   See the *Sterling Selling and Fulfillment Foundation: Multi-Tenant Enterprise Guide* for more information about the SMA.

   Note: You cannot add the datasource attribute for the metadata pool in the SMA. You must use the manageDBpool API instead.
7. Build and deploy the EAR.
8. Update the application server configuration to use `-Dvendor=<`*`appserver`*`>`

   – where the value of <*appserver*> is weblogic, jboss, or websphere.
9. Start the application server.

Results

Depending on your application server, you can find more information about `-Dvendor` and other properties in the application server startup scripts in "Configuring an Oracle WebLogic Application Server - Overview" on page 109, "Configuring a WebSphere Application Server - Overview" on page 125, or "Configuring a JBoss Application Server" on page 141.

Configuring a Restrictive Cookie Path - Overview

When multiple applications are deployed on the same domain and the restrictive cookie path is not set, the user may be automatically logged off from the application when one application sends information to another application. For example, in IBM Sterling Business Center, you will automatically be logged off in the following scenarios:

- The image server and the IBM Sterling Business Center application are deployed on the same domain, and you click the **Related Tasks** link in the IBM Sterling Business Center application.
- The Sterling Configurator Visual Modeler and the IBM Sterling Business Center application are deployed on the same domain, and you click the **Launch Visual Modeler** link in the IBM Sterling Business Center application.

Configuring a Restrictive Cookie Path on WebLogic

About this task

To set the restrictive cookie path in Oracle WebLogic, complete the following:

Procedure

1. Extract the `weblogic.xml` file from the war package where you want to add the restrictive cookie path.
2. Copy the extracted `weblogic.xml` file to the following location:
   `<INSTALL_DIR>/extensions/<WAR Package>`

   where `<WAR Package>` is the package for the deployed application. For example, this would typically be smcfs, sbc, and sfs for the Sterling Selling and Fulfillment Foundation, IBM Sterling Business Center, and IBM Sterling Field Sales applications, respectively.
3. Add the following to the `weblogic.xml` file:

```
<session-descriptor>
   <session-param>
   <param-name>CookiePath</param-name>
   <param-value>/<context-path></param-value>
   </session-param>
  </session-descriptor>
```

   where *<context-path>* is the context path for the deployed application. For example, this would typically be /smcfs, /sbc, and /sfs for the Sterling Selling and Fulfillment Foundation, IBM Sterling Business Center, and IBM Sterling Field Sales applications, respectively.
4. Rebuild the EAR file.

Configuring a Restrictive Cookie Path on WebSphere

Procedure

1. In the WebSphere Administration Console, navigate to the **Session Manager** > **Cookie** tab.

2. In the **Cookie** tab, set **Cookie Path** to the context path of your application.

   For example, this would typically be /smcfs, /sbc, and /sfs for the Sterling Selling and Fulfillment Foundation, IBM Sterling Business Center, and IBM Sterling Field Sales applications, respectively.

Configuring a Restrictive Cookie Path on JBoss

By default, JBoss sets the restrictive cookie path; therefore, no additional configuration is required.

Setting the Client Character Display

When displaying special characters, such as for various languages, the client computer must be configured to display these characters.

For Unicode characters to display correctly in the Application Console, each Windows client must be configured.

1. To configure a client machine, select Control Panel > Regional and Language Options.

Clearing Browser Caches

About this task

Once Sterling Selling and Fulfillment Foundation is ready for deployment, each user must clear the browser caches on their client machines before launching Sterling Selling and Fulfillment Foundation.

To clear the browser cache:

Procedure

1. From the Windows start menu, select Settings > Control Panel > Internet Options. Choose the General tab, and in the Temporary Internet Files inner panel, choose the Delete Files button. The Delete Files dialog displays.
2. Enable the Delete All Offline Content option. Then click OK, and click OK once more.
3. Close the Internet Properties window.

Clearing the Java Plugin Cache

About this task

Once Sterling Selling and Fulfillment Foundation is ready for deployment, each user must clear the Java Plugin caches on their client machines before launching Sterling Selling and Fulfillment Foundation.

To clear the Java plugin cache:

Procedure

1. From the Windows start menu, select Settings > Control Panel > Java Plugin and choose the Cache tab.
2. Click Clear JAR Cache.
3. Click OK.
4. Close the Java Plugin Control Panel window.

Statistics Monitoring

In order to measure throughput performance, runtime statistics can be gathered. Note that this feature and the data gathered by it in the YFS_STATISTICS_DETAILS table are only for use byIBM personnel, as any metric can change without notice.

In a production environment, you should leave statistics generation enabled to collect statistics data in 10 minute intervals (the default). You should also schedule statistic purges on a regular basis (for example, every two weeks).

Setting an Enterprise for Logging In to IBM Sterling Business Center

IBM Sterling Business Center supports enterprise-specific login. Therefore, when you log in to IBM Sterling Business Center, it is mandatory that along with your User ID and password, you set the enterprise code of the enterprise that you want to administer.

You should set the enterprise code as a request parameter in the login URL. Therefore, to log in to IBM Sterling Business Center, you should use the following URL:

```
http://<server>:<port>/sbc/sbc/launch.jsp?EnterpriseCode=<Enterprise_Code>
```

For example, if the enterprise code of the enterprise you want to administer is XYZ-123, use the following URL to log in to IBM Sterling Business Center:

```
http://<server>:<port>/sbc/sbc/launch.jsp?EnterpriseCode=XYZ-123
```

Chapter 15. Deploying and Updating the Rich Client Platform Applications

Before You Begin RCP Deployment

Before you start deploying a Rich Client Platform application you must have installed Sterling Selling and Fulfillment Foundation. For more information about installing Sterling Selling and Fulfillment Foundation, see Chapter 8, "Installing Sterling Selling and Fulfillment Foundation in UNIX and Linux Environments," on page 57 or Chapter 7, "Installing Sterling Selling and Fulfillment Foundation in a Windows Environment," on page 37.

Go to the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning for information about supported software versions specifically for RCP, including operating systems and browsers.

The components that are shipped as part of Sterling Selling and Fulfillment Foundation (that is, what is available when Sterling Selling and Fulfillment Foundation is installed) are:

- RCP Infrastructure plug-in compressed file
- RCP Foundation plug-in compressed file
- JREs for each of the supported operating systems. For more information about supported JREs and operating systems, go to the IBM Support Portal at http://www.ibm.com/support/entry/portal/Planning.
- Eclipse dependencies for each of the supported operating systems

After you install Sterling Selling and Fulfillment Foundation, you can view the directory structure, which contains:

- The <INSTALL_DIR>/platformrcp/<PLATFORM_VERSION> folder—This contains the Rich Client Platform files, plug-ins, or JREs.
- The <INSTALL_DIR>/platformrcp/<PLATFORM_VERSION>/rcpclient folder—This contains the Rich Client Platform plug-in and tools plug-in compressed files.
- The <INSTALL_DIR>/platformrcp/<PLATFORM_VERSION>/rcpdependencies folder—This contains the Rich Client Platform dependency directories for the supported operating systems. For example, linux-gtk, windows, and so forth. Each of these directories contains the supported JREs and Eclipse plug-ins, features, or files. Also, each of these directories contain the osversion.properties text file which provides information about the supported versions of the operating system.
- The <INSTALL_DIR>/repository/xapi/template/merged/api folder—This contains the API XML templates used by the Rich Client Platform.
- The <INSTALL_DIR>/properties folder—This is the location of the customer_overrides.properties file, which is used when enabling auto updates for the individual PCA. For more information about enabling auto updates, see "Deploying RCP - Applying Updates" on page 165

What is available when a Rich Client Platform-based PCA is installed?

When you install a Rich Client Platform-based PCA client, a compressed file that contains the Rich Client Platform application plug-ins or features is provided. For example, when you install the IBM Sterling Call Center application, the <INSTALL_DIR>/rcp/COM/rcpclient directory is automatically created. The com.zip file is stored in this directory, which contains the IBM Sterling Call Center-specific plug-ins or features.

Note: If deploying an application, such as IBM Sterling Call Center or IBM Sterling Store Inventory Management, also see the application Deployment Guide for information specific to that application for configuring the Rich Client Platform.

Deploying RCP - Creating the RCP_EXTN_FOLDER Folder

About this task

To maintain all SSL certificates, you must create a <RCP_EXTN_FOLDER> folder in which to store any new plug-ins and resource files that you create while extending Rich Client Platform-based PCA client application. The environment variable for this folder is RCP_EXTN_FOLDER.

The <RCP_EXTN_FOLDER> folder structure is better explained with an example as follows:

Procedure

1. Create an appropriate <RCP_EXTN_FOLDER> folder for storing the RCP extensions that you create when extending the Rich Client Platform-based PCA client application. For example, rcpextnworkarea folder.

   Note: You can create the <RCP_EXTN_FOLDER> folder in any directory outside the <INSTALL_DIR> directory.
2. Under the <RCP_EXTN_FOLDER> folder, create the following directories:
   - libs
   - plugins
   - resources
   - truststore
3. In the <RCP_EXTN_FOLDER>/plugins directory, store all new plug-ins that you created for extending the screens.
4. In the <RCP_EXTN_FOLDER>/resources directory, store the locations.ycfg file, rcpsecureapis.xml file (if necessary), localized bundle and theme files, and localized icons. The ant script creates the resources.jar file and copies the contents of the resources folder into this jar file. After copying the contents, the resources.jar file is copied into the Rich Client Platform plug-in.
5. In the <RCP_EXTN_FOLDER>/truststore directory, store the SSL trust certificates that needs to be used when the client application is communicating with the server in secure mode. The SSL certificates are automatically copied by the ant script to the correct folder in the Rich Client Platform plug-in.
6. Create the jasper folder within the <RCP_EXTN_FOLDER>/libs directory.
7. Copy the following jasper libs needed for JasperReports to the <RCP_EXTN_FOLDER>/libs/jasper folder:
   - barbecue-1.5-beta1.jar

- commons-beanutils-1.8.0.jar
- commons-collections-2.1.1.jar
- commons-digester-1.7.jar
- commons-logging-1.0.4.jar
- iReport.jar
- itext-2.1.0.jar
- jasperreports-3.6.2.jar

  These are third party files, which must be downloaded from a third party site. They are not provided with Sterling Selling and Fulfillment Foundation. For instructions on downloading jasper libs, see the <INSTALL_DIR>/xapidocs/code_examples/jasperreports/alert_readme.html file for instructions.

Deploying RCP- Caching Data Types Locally

About this task

To improve the system performance when logging into the Rich Client Platform-based PCA application, you must cache data locally in the client.

To cache data types locally:

Procedure

1. Copy the datatypes.xml file from the <INSTALL_DIR>/repository/datatypes folder, and yfsdatatypemap.xml files from the <INSTALL_DIR>/repository/xapi/template/merged/resource directory to the <RCP_EXTN_FOLDER>/resources directory.
2. Create the extn directory under the <RCP_EXTN_FOLDER>/resources directory.
3. Copy the extended datatypes.xml and yfsdatatypemap.xml from the extensions directory to the <RCP_EXTN_FOLDER>/resources/extn directory.

Deploying RCP - Configuring Locations

A location is synonymous to a geographic location. For example, store location, call center location, and so forth. Each location has an identifier associated with it, which uniquely identifies the appropriate geographical location.

To configure locations, you must define locations in the locations.ycfg file. For Sterling Selling and Fulfillment Foundation, the locations.ycfg.sample file is located in the <INSTALL_DIR>/platformrcp/<PLATFORM_VERSION>/rcpclient/com.yantra.yfc.rcp_<version> folder.

To configure locations, copy the locations.ycfg.sample file to create a new locations.ycfg file.

Note: If deploying an application, such as Sterling Call Center, Sterling Store, or Sterling Store Inventory Management, see the application Deployment Guide for information specific to that application.

Deploying RCP - Creating and Configuring a New locations.ycfg XML File

About this task

To configure a new locations.ycfg file:

Procedure

1. Copy the sample locations.ycfg file from the <INSTALL_DIR>/platformrcp/<PLATFORM_VERSION>/rcpclient/com.yantra.yfc.rcp_<version> folder.and store it in the <RCP_EXTN_FOLDER>/resource directory.

   where <RCP_EXTN_FOLDER> refers to the folder that you created for storing Rich Client Platform-based PCA client application extensions. For more information about creating the <RCP_EXTN_FOLDER> folder, see “Deploying RCP - Creating the RCP_EXTN_FOLDER Folder” on page 162.

2. Define new locations in the locations.ycfg file by using the information provided in the locations.ycfg.sample file, which contains proxy server and application server URL settings for various geographical locations.

   Following is sample configuration data from the locations.ycfg.sample file:

```
<?xml version="1.0" encoding="UTF-8"?>
 <Locations>
   <Location id = "DEFAULT"
             proxyServer="yourproxyserver.com"
             proxyPort="8080"
             updateType ="pull">
             <Config Name = "DEFAULT"
                     Protocol = "http"
                     BaseUrl = "localhost"
                     PortNumber = "7001"
                     ApiUrl ="/<WEB_APP_NAME>smcfs/RcpServlet"
                     CompressionEnabled = "N"
             </Config>
   </Location>
   <Location id = "REMOTE"
             proxyServer="yourproxyserver.com"
             proxyPort="8080"
             updateType ="client">
             <Config Name = "IMAGE"
                     Protocol = "http"
                     BaseUrl = "localhost"
                     PortNumber = "7001"
                     ApiUrl ="/icons/rcp/$param1$.gif"
                     CompressionEnabled = "N"
             </Config>
   </Location>
 </Locations>
```

3. Define the Locations root element.
4. Define the Location element under the Locations root element with id such as DEFAULT, LOCAL, REMOTE, and so forth. You can configure the proxy server and application server URL settings for each location.

   Note: You must have one Location element with the id attribute value set as "DEFAULT". This Location element must have a Config element whose `Name` attribute must have the value set as "DEFAULT".

   Note: When you log in to a Rich Client Platform application using a particular location, the system checks whether or not the loaded location has a "DEFAULT" Config element defined for it. If the selected location has the "DEFAULT" Config element, the system loads the "DEFAULT" configuration. Otherwise, the system loads the "DEFAULT" configuration defined in the "DEFAULT" location.

   For more information about location configuration settings, see “Deploying RCP - Configuring Location Settings” on page 168.

Deploying RCP - Modifying the locations.ycfg.sample XML File

About this task

To modify the locations.ycfg.sample file:

Procedure

1. Copy the locations.ycfg.sample file from the <INSTALL_DIR>/platformrcp/<PLATFORM_VERSION>/rcpclient/com.yantra.yfc.rcp_<version> folder and store it in the <RCP_EXTN_FOLDER>/resource directory.

   where <RCP_EXTN_FOLDER> refers to the folder that you created for storing Rich Client Platform-based PCA client application extensions. For more information about creating the <RCP_EXTN_FOLDER> folder, see "Deploying RCP - Creating the RCP_EXTN_FOLDER Folder" on page 162.
2. Rename the locations.ycfg.sample file to locations.ycfg file.
3. Modify the location configurations settings as needed. For information about location configuration settings, see "Deploying RCP - Configuring Location Settings" on page 168.

Deploying RCP - Localizing Bundle and Theme Files

You can localize the Rich Client Platform application's locale-specific files based on the user's locale. The Rich Client Platform supports the bundle and theme locale-specific files. All the Rich Client Platform application plug-ins contain the `<Plug-in_id>_<name>.properties` bundle file and `<Plug-in_id>_<theme_name>.ythm` theme file. For more information about localizing bundle and theme files, see the *Sterling Selling and Fulfillment Foundation: Localization Guide*.

Deploying RCP - Enabling HTTPS

If you are using the HTTPS connection to communicate with the application server, copy all SSL (Secure Socket Layer) certificates in the `truststore` directory. For more information about the `truststore` directory, see "Deploying RCP - Creating the RCP_EXTN_FOLDER Folder" on page 162.

For more information about configuring connection settings for HTTPS connection, see "Deploying RCP - Configuring Connection Settings for HTTPS Connection" on page 172.

Deploying RCP - Applying Updates

About this task

The Rich Client Platform's update process is based on the timestamp of the files. In the `<INSTALL_DIR>/properties/yfs.properties.in` file, the `yfs.rcp.pca.updates.dir` property points to the directory where updates for the PCAs are located. The `yfs.rcp.pca.updates.cache.dir` property points to the local directory on the application server where updates for the PCAs can be cached.

To deploy updates for the Rich Client Platform application on a client:

Procedure

1. Modify the `<INSTALL_DIR>/properties/customer_overrides.properties` file to configure the following properties:
   - Configure the `yfs.rcp.pca.updates.dir` property by specifying the path of the directory where updates for the PCAs are located. The directory that you

specify can also be a shared directory on the network. For example, yfs.yfs.rcp.pca.updates.dir = <INSTALL_DIR>/<PCA_UPDATES_DIR>

where yfs.rcp.pca.updates.dir is the property, <INSTALL_DIR> is the directory where you have installed Sterling Selling and Fulfillment Foundation, and <PCA_UPDATES_DIR> is the directory which contains individual updates folder for each Sterling Selling and Fulfillment Foundation PCA.

For example, if the root folder for PCA updates is maintained in the <INSTALL_DIR>/<PCA_UPDATES_DIR> directory, and for Sterling Store Inventory Management, if the application identifier is YFSSYS00011, PCA code is com20, and operating system configuration is win32.win32.x86, the client searches for updates based on the application ID, PCA code, and operating system configuration. The Sterling Store Inventory Management updates are maintained in the <INSTALL_DIR>/<PCA_UPDATES_DIR>/YFSSYS00011/com20/win32.win32.x86 directory.

You can find the following resources in this directory:

- Rich Client Platform client plug-in
- Sterling Store Inventory Management and related plug-ins
- Eclipse related plug-ins

Note: The JRE files are not updated.

- Configure the yfs.yfs.rcp.pca.updates.cache.dir property by specifying the path of the local directory on the application server where updates for PCAs need to be cached. For example, yfs.yfs.rcp.pca.updates.cache.dir = <INSTALL_DIR>/<PCA_UPDATES_DIR>/<UPDATES_CACHE_DIR>

Note: Make sure that the directory specified in the yfs.rcp.pca.updates.dir property is different from the directory specified in the yfs.rcp.pca.updates.cache.dir property.

For additional information about overriding properties using the customer_overrides.properties file, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.

2. Modify the locations.ycfg file to define the type of update you want to deploy on the client in the updateType attribute of the Location element. The Rich Client Platform supports two methods of deploying updates on the client: Client Pull and Push Updates. For more information about the different types of updates that the Rich Client Platform supports, see "Deploying RCP - Types of Updates."

Deploying RCP - Types of Updates

The Rich Client Platform's update process is based on the timestamp of files. The Rich Client Platform supports two methods of deploying updates for a Rich Client application on the client:

- Client Pull or Automatic Update—Client Pull is the automatic way of deploying updates on the client. In this type of update, when a user logs in to a Rich Client Platform application, based on the location configuration settings, the client application automatically starts searching for updates in a background thread and installs them. Once all updates are downloaded successfully and installed, the user is authorized to restart the application.
- Push Updates or Manual Update—Push Updates is the manual way of deploying updates on the client. If you want to use push updates option, copy the contents of the update directory based on the client application which you want to update to the client machine. For example, if you have specified the

update directory as: <INSTALL_DIR>/<PCA_UPDATES_DIR>, copy the contents from the following directory to the client machine.

<INSTALL_DIR>/<PCA_UPDATES_DIR>/<PCA_APPLICATION_ID>/
<PCA_APPLICATION_VERSION>/<OS_CONFIG>

where <INSTALL_DIR> refers to the directory where you have installed the Sterling Selling and Fulfillment Foundation.

<PCA_UPDATES_DIR> is the directory where updates are located,
<PCA_APPLICATION_ID> is the identifier of the client application for which you want to deploy updates using the Push Update method,
<PCA_APPLICATION_VERSION> is the version number of the client application, and
<OS_CONFIG> refers to the <Windowing_System>.<OS>.<OS_ARCH> operating system.

Deploying RCP - Running the Ant Script

Run the application-specific ant script with the appropriate ant target as needed. The ant script is provided by the appropriate Rich Client Platform application. For example, if you want to deploy Sterling Call Center, run the buildcomapplication.xml file.

The ant file contains multiple ant targets to generate the deployable folder for all unique combinations of the Operating System and Application such as buildCOMForWindows, buildCOMForLinuxGTK, buildSOPForWindows, and so forth.

For example, if you want to deploy Sterling Call Center on Windows, run the following ant script from the <INSTALL_DIR> directory with an ant target:

For Windows:

```
sci_ant.cmd -f bin/buildcomapplication.xml buildCOMForWindows
```

For UNIX:

```
sci_ant.sh -f bin/buildcomapplication.xml buildCOMForWindows
```

Note: For this ant script to run, the following arguments or variables need to be exported:

- <INSTALL_DIR>—name of the folder where Sterling Selling and Fulfillment Foundation is installed.
- <RCP_EXTN_FOLDER>—specify the name of the <RCP_EXTN_FOLDER> folder that you created for storing Rich Client Platform-based PCA client application extensions. For more information about creating <RCP_EXTN_FOLDER> folder. see “Deploying RCP - Creating the RCP_EXTN_FOLDER Folder” on page 162.

After you run this ant script runtime the resources or directory structure are created or generated for the call center application.

- The rcpdrop folder is created within the <INSTALL_DIR> directory.

  where <INSTALL_DIR> refers to the directory where you have installed Sterling Selling and Fulfillment Foundation.
- Based on the ant target that you specified, when you run the ant script, a folder for the operating system is created. For example, the windows folder is created if you specify buildCOMForWindows as the ant target to deploy Sterling Call Center on the Windows operating system.
- Under the windows folder, the <application> folder is created. For example, com.

  The com folder contains the required files and resources for the application that are to be built. These resources are accumulated from the following folders:

- <INSTALL_DIR>/rcpclient/
- <INSTALL_DIR>/rcpdependencies/windows
- <INSTALL_DIR>/rcp/COM/rcpclient
- <RCP_EXTN_FOLDER>

  where <INSTALL_DIR> refers to the directory where you have installed Sterling Selling and Fulfillment Foundation.

  <RCP_EXTN_FOLDER> refers to the folder that you created for the storing Rich Client Platform-based PCA client application extensions. For more information about creating the <RCP_EXTN_FOLDER> folder, see "Deploying RCP - Creating the RCP_EXTN_FOLDER Folder" on page 162.

- Also the updatets.xml file is created which is used by the Rich Client Platform to check for auto updates.

  The updatets.xml file contains a list of files that are present in the application. It also includes the timestamp for these files.

  Note: The updatets.xml file is automatically generated by the ant script provided with Sterling Selling and Fulfillment Foundation for building a PCA Application.

Deploying RCP - Deploying Clients through a Remote Terminal

RCP clients can be deployed and accessed on a terminal server through a remote login from a client machine, by using Windows Terminal Server. Terminal Server is the server component of Terminal services. It authenticates clients, provides access to remote clients and also controls the level of access for each client. This service uses the Remote Desktop Protocol (RDP), which enables a user to connect to the remote server (running Microsoft Terminal Services). Any client that supports RDP can be used as terminal client to connect to the server.

To run RCP clients through a remote login, add the Terminal Server and route your LAN through it.

Note: Only one user per client can log in to the terminal server.

Deploying RCP - Configuring Location Settings

About this task

Location configurations are defined in the locations.ycfg file. You can set different preferences for each location.

To define a new location configuration:

Procedure

1. Set the attributes of the Location element. For Location element attributes and their descriptions, see Table 9.

Table 9. Location Element Attribute List

Attribute	Description
id	Specify a unique identifier for the geographical location. For example, DEFAULT, REMOTE, LOCAL, and so forth.
proxyServer	Specify the unique proxy server used to connect to the internet, if applicable.

Table 9. Location Element Attribute List (continued)

Attribute	Description
proxyPort	Specify the port number of the proxy server.
updateType	Set this attribute only when you are updating a Rich Client Platform application. Specify the mode of update you want to perform, if applicable. Valid update modes are: pull and push. For more information about update modes, see "Deploying RCP - Applying Updates" on page 165.

2. Define a Config element under the Location element to configure the connection settings. Each location has multiple Config elements. For example, DEFAULT, IMAGE_SMALL, IMAGE_BIG, and so forth. Using the Config element, define the various configuration settings. Set all attributes of the Config element to specify the application server URL you want to use. For more information about configuring connection settings, see "Deploying RCP - Connection Settings" on page 170.

   Note: You must have one Location element with `id` attribute value as "DEFAULT" and this Location element must have a Config element whose `Name` attribute should have the value as "DEFAULT".

   When you log into a Rich Client Platform application using a particular location, the system checks whether or not loaded location has a "DEFAULT" Config element defined for it. If the selected location has "DEFAULT" Config element, the system loads the "DEFAULT" configuration. Otherwise the system loads the "DEFAULT" configuration defined in the "DEFAULT" location.

   A sample configuration data used to define a location configuration is as follows:

```
<Location id = "DEFAULT"
          proxyServer="proxy.sterling.com"
          proxyPort="8080">
          <Config Name = "DEFAULT"
                  Protocol = "http"
                  BaseUrl = "localhost"
                  PortNumber = "7001"
                  ApiUrl = "/<WEB_APP_NAME>smcfs/RcpServlet"
                  CompressionEnabled = "N">
          </Config>
</Location>
```

   When you start the Rich Client Platform application, the system reads the `locations.ycfg` file and loads the location information available in this file.

   When you start the application for the first time, the Location Preferences window displays.

3. Select a location from the drop-down list.
4. Configure the proxy server settings, if applicable.

Results

Based on the location preferences, you are logged into the application.

Deploying RCP - Connection Settings

To connect to the application server, you must configure the Rich Client Platform application. In the `locations.ycfg` file, set the protocol, base URL, port number, API URL, and other attributes of the Config element. For Config element attributes, see Table 10. You can configure the connection settings for fetching images from the server or connecting to HTTPS.

Table 10. Config Element Attribute List

Attribute	Description
Name	Specify a unique name for the server configuration. For example, LOCAL, DEFAULT, and so forth.
Protocol	Specify the name of the protocol to use to communicate with the application server. For example, http or https. For more information about configuring connection settings for HTTPS protocol, see “Deploying RCP - Connection Settings for Fetching Images from the Server” on page 171.
BaseUrl	Specify the base URL path of the application server. For example localhost or 10.11.25.80 or www.myserver.com.
PortNumber	Specify the port number based on the protocol you specified. For example, 7001 or 7002.
ApiUrl	Specify the URL path of the application server where all APIs are stored. For example, `/<WEB_APP_NAME>smcfs/RCPServlet`. If you want to display images from the server, the URL path must contain `$param1$` parameter. For more information about configuring connection settings for fetching images from the server, see “Deploying RCP - Connection Settings for Fetching Images from the Server” on page 171.
CompressionEnabled	If the data received from the server is in the compressed format, set the `CompressionEnabled` attribute to "Y". The Rich Client Platform supports only Gzip compression format. For more information about the supported compression format, see “Deploying RCP - Compression in the Rich Client Platform” on page 173.

A sample configuration data used to configure a server is as follows:

```
<Config Name = "DEFAULT"
             Protocol = "http"
             BaseUrl = "localhost"
             PortNumber = "7001"
             ApiUrl = "/<WEB_APP_NAME>smcfs/RcpServlet"
             CompressionEnabled = "Y">
</Config>
<Config Name = "LOCAL"
        Protocol = "http"
        BaseUrl = "localhost"
        PortNumber = "7001"
           HttpsPortNumber = "7002
        ApiUrl = "/<WEB_APP_NAME>smcfs/RcpServlet"
        CompressionEnabled = "N">
</Config>
```

Note: You must have one location element with `id` attribute value as "DEFAULT". This location element must have a Config element with `Name` attribute value as "DEFAULT", which defines the DEFAULT URL for the connecting to the application server.

The Rich Client Platform application is initially launched by connecting to the server specified in the "DEFAULT" URL. You can define the URL at each command level, if applicable. If the command element in the `<Plug-in_id>_commands.ycml` file is not associated with the URL, the system considers the "DEFAULT" URL for that command.

Deploying RCP - Connection Settings for Fetching Images from the Server

About this task

You can configure the connection settings to fetch images from the server by setting the protocol, base URL, port number, API URL, and other attributes of the Config element in the `locations.ycfg` file. For Config element attributes, see Table 11. You can create more than one configurations to display different types of images.

Note: This configuration for displaying item images is relevant only for Sterling Store Inventory Management PCA. To configure item images in Sterling Call Center and Sterling Store PCA, refer the "Displaying Item Images" section of the "Searching and Viewing the Details of an Item Solution" topic in *Sterling Store: Implementation Guide* and *Sterling Call Center: Implementation Guide*.

Table 11. Config Element Attribute List

Attribute	Description
Name	Specify a unique name for the server configuration.
Protocol	Specify the name of the protocol to use to communicate with the application server. For example, http or https.
BaseUrl	Specify the base URL path of the server. For example localhost, 10.11.25.80, or www.myserver.com.
PortNumber	Specify the port number based on the protocol that you have specified. For example, 80.
ApiUrl	Specify the URL path of the server where all the images are stored. The URL path must contain `$param1$` parameter. For example, `/icons/rcp/$param1$.gif`.
DefaultApiUrl	Specify the URL path of the image that displays if the image specified in the ApiUrl is not found, if applicable. For example. `/icons/rcp/404.jpeg.`

Note: You can create the following server configurations to fetch images of different types such as GIF, JPEG, PNG, and so forth:

- IMAGE
- IMAGE_SMALL
- IMAGE_MEDIUM
- IMAGE_BIG

Each location must have a server configuration named "IMAGE" which defines the URL for fetching images from the server. You can configure the "IMAGE" URL to get images of type GIF, JPEG, PNG, and so forth. All other server configurations are optional.

The sample configuration data that is used to configure server for displaying images is given below:

```
<Config-List>
     <Config Name = "IMAGE"
             Protocol = "http"
             BaseUrl = "localhost"
             PortNumber = "80"
             ApiUrl = "/icons/imgservlet/?file=$param1$"
       </Config>
       <Config Name = "IMAGE_SMALL"
             Protocol = "http"
             BaseUrl = "localhost"
             PortNumber = "80"
             ApiUrl = "/icons/rcp/$param1$.gif"
                 DefaultApiUrl = "/icons/rcp/404.gif"
     </Config>
     <Config Name = "IMAGE_BIG"
             Protocol = "http"
             BaseUrl = "localhost"
             PortNumber = "80"
             ApiUrl = "/icons/rcp/$param1$.jpeg"
                 DefaultApiUrl = "/icons/rcp/404.gif"
     </Config>
</Config-List>
```

For example, to get an image from the server using the `http://localhost:80/icons/imgservlet/?file=Y001` URL, define a Config element named `IMAGE` as shown in the sample code (above). To fetch an image from the server using the `http://localhost:80/icons/rcp/Y001.gif` URL, define a Config element named `IMAGE_SMALL` as shown in the sample code (above). In both the cases, the `$param1$` variable is replaced by the image's name.

Note: You can modify the Config element for the IMAGE URL. But ensure that you do not delete it.

For example, if you want to get an image for an OrderNo label:

Procedure

1. Set the source binding for the label as:

   ```
   lblOrderNo.setSourceBinding("ServerImageList:Images/Icons/RCP/Image1/@OrderNo");
   ```

   where `lblOrderNo` is the label name and ServerImageList is the namespace for the model.
2. Set the server image configuration for the label to display the image from the server as shown:

   ```
   lblBindingData.setServerImageConfiguration(YRCConstants.IMAGE_SMALL);
   ```

   where `lblBindingData` is the binding object and `IMAGE_SMALL` is the value of the `Name` attribute of the Config element, which is defined in the configuration file.

Results

When getting the image for the `lblOrderNo` label, the `$param1$` parameter is replaced by the value of the `OrderNo` attribute. If the value of the `OrderNo` attribute is "Y001", the image `Y001.gif` displays for the lblOrderNo label.

Deploying RCP - Configuring Connection Settings for HTTPS Connection

About this task

To configure the connection settings to communicate with application servers:

Procedure

1. In the `locations.ycfg` file when defining the connection settings, set the value of `Protocol` attribute of the Config element as "https". Also, specify the port number for the HTTPS protocol in the `PortNumber` attribute of the Config element. For more information about configuring the connection settings, see "Deploying RCP - Connection Settings" on page 170.
2. By default, during handshaking, if there is a mismatch between the URL's hostname and the server's identification hostname, the Rich Client Platform allows the HTTPS connection.
3. Copy all SSL or public key certificates required for configuring an HTTPS connection in the `truststore` directory under the `extensions` folder that you created as shown:

   `<RCP_EXTN_FOLDER>/truststore.`

   where `<RCP_EXTN_FOLDER>` refers to the folder that you created for storing the Rich Client Platform-based PCA client application extensions. For more information about creating <RCP_EXTN_FOLDER> folder, see "Deploying RCP - Creating the RCP_EXTN_FOLDER Folder" on page 162.

   A trusted Certificate Authority (CA) like VeriSign issues these security certificates. For more information about SSL or security certificates, see "Deploying RCP - Security Certificates."

Deploying RCP - Security Certificates

An SSL certificate or public key certificate is a certificate that uses a digital signature to bind a public key with an identity information such as the name of the person or an organization, address, and so forth. An SSL certificate has information about the owner of the certificate, the usage of the certificate, validity details, resource location or web site address, e-mail address and the certificate ID of the person who certifies (signs) this information. SSL certificates are used for secure communication over the HTTPS protocol.

Whenever a client needs to verify the authenticity of an SSL server, the SSL certificate used by the server needs to be signed by the Certificate Authority that is already trusted by the client. The well-known certificate authorities such as Thawte and VeriSign serve as an authoritative, trusted third party for authentication. They sign the SSL certificates that are used when dealing with sensitive information or services. If these SSL certificates are signed by a trusted authority, it is possible to verify the identity of a server by supplying the SSL certificate.

Deploying RCP - Compression in the Rich Client Platform

The Rich Client Platform enables you to send and receive compressed data to and from the application server. When you enable compression, the Rich Client Platform enables bidirectional compression.

Benefits

- The bidirectional compression helps in reducing the traffic in both directions as only the XML data is passed to an API or service. For example, input XMLs and output templates passed to an API or service.
- The compression is most useful for applications that rely more on multiple API calls because it avoids multiple trips to and from the application server.
- There is minimal overhead in performing compression. For example, when an XML file size is large, we can reduce the size of the data by about 90%.

Note: The Rich Client Platform supports the Gzip compression format.

Note: The Rich Client Platform does not support compression of images or compressed files when fetching images or extracting updates from the server.

To enable compression, in the `locations.ycfg` file, you must set the value of the `CompressionEnabled` attribute of the Config element to "Y". These settings are done when you are configuring the connection settings for a Rich Client Platform application. For more information about configuring connection settings, see "Deploying RCP - Connection Settings" on page 170.

Deploying RCP - Creating Server-Side Commands Without Running the Application

If the RCP PCA application is run with the `yfs.rcp.devmode` property set to "FALSE" in the `yfs.properties` file, ensure that the following command file is present in the server:

```
( <runtime>\templates\com.yantra.yfc.rcp\commands\<APP_ID>\ )
```

The name of the command file can be either `commands.ycml` or `commands_<COMMANDS_VERSION>.ycml`. The `<COMMANDS_VERSION>` value is read from the `client.properties` file. If this value is not available, then the `commands.ycml` file will be used.

To create the merged command file, the following utility class is provided:

```
com.yantra.yfc.rcp.internal.YRCCommandsMergeUtils in
Platform/rcpclient/com.yantra.yfc.rcp_1.0.0/yrcui.jar
```

Invoke this utility class using the following arguments:

- `rcpClientDir=<RCP_CLIENT_DIR>`
- `rcpCommonDir=<RCP_COMMON_DIR>`
- `destDir=<DEST_DIR>`
- `commandsDirs=<PCA_COMMANDS_DIR >`
- `extnCommandsDir=<EXTN_COMMANDS_DIR>`
- `applicationId=<APP_ID>`

Here,

- `<RCP_CLIENT_DIR>` refers to the rcpClient directory.
- `<RCP_COMMON_DIR>` refers to the rcpCommon directory.
- `<DEST_DIR>` refers to the directory for the merged files.
- `<PCA_COMMANDS_DIR>` refers to the directory containing the comma-separated values of all the PCA commands.
- `<EXTN_COMMANDS_DIR>` refers to the directory containing the directories for each of the extension plugins that contain all the command (.ycml) files. The directory name of each plugin should be the plugin ID of that plugin.
- `<APP_ID>` refers to the application ID.

The following files must be present in the class path when running the Java class:

- yrcui.jar
- eclipse Plugins jars
- resources.jar

Index

A

access control 112, 127
agent criteria
  override 104
authentication
  using JAAS 4
  using LDAP 4

B

backing up
  databases 31
backupScriptGen.xml 31
barcode printing
  WebLogic 112
  WebSphere 127
browser
  clearing cache 158

C

capacity planning 17
clearing browser 158
client character 158
client characters
  displaying special characters 158
  displaying Unicode character 158
COM+ (extended component object model) 8
component object model. 8
custom classes deployment 121, 137, 152

D

database schema creation (single schema) 96
database security 10
  credit card encryption 10
database sizing 17
  capacity planning 17
  disk estimation
    DOM module 17
  future disk estimation 19
database user privileges 22, 34
  oracle
    administrative user 22, 34
    application user 23, 34
  Oracle 22, 34
database verification (single schema) 96
databases
  backup and restore 31
  configuring 31
  Oracle 24, 33
  Oracle (Windows) 41
DatabaseSecurity 10
DB2 database
  sizing, See Also database sizing 21, 26, 32
deploying
  custom classes (WebLogic) 121, 137, 152
deploying Selling and Fulfillment Foundation 109
deployment architecture
  analyzing
    authentication mechanism 4
    current security infrastructure 4
    data encryption 4
    network topology 4
disk estimation
  DOM module 17
    estimation methodology 17
    steps involved 18
  future requirements 19
DOM module
  disk space estimation 18
DOM module disk estimation 17

E

EFrame_indexadds.sql 99
EFrame_indexdrops.sql 99
EFrame_sequence.sql 98
EFrame_tablechanges.sql 98
EFrame_TableChanges.sql 98
EFrame_tabledrops.sql 98
EFrame_TextIndexAdds.sql 99
EFrame_TextIndexDrops.sql 99
EFrame_TextIndexModify.sql 99
EFrame_TextIndexUpdates.sql 99
EFrame_UpdateQueries.sql 99
EJB (Enterprise JavaBeans) 6
ejbdeploy.sh 126
enterprise JavaBeans. See EJB 6
estimates of database 17

F

factory defaults 94
  installer restart file 95

G

generating EJB stubs 126
graph display on WebLogic 112
graph display on WebSphere 127

H

history database 18
HTTP
  in-memory session replication 112
HTTP (hypertext transfer protocol) 6
HTTPS (hypertext transfer protocol secure) 7
hypertext transfer protocol secure. See HTTPS 7
hypertext transfer protocol. See HTTP 6

I

installation summary 1
installing
  Oracle 32
  Sterling Sensitive Data Capture Server 79
  UNIX 69
  Windows 42, 47
installing (from command line) 44
  Linux 64, 66
  UNIX 64, 66
  Windows 44
installing (from silent install file)
  Linux 68, 76
installing (from Windows) 47
installing application server 13
installing database schema (single schema) 96
installing database software
  Oracle. See Oracle 32
installing database software (Windows) 21, 31
installing jasperreports 92
installing Oracle 32
installing print server 91
installing Sterling Supply Chain Applications language pack 81
Internet Explorer security
  adding trusted web site 10
invoking
  custom classes 121, 137, 152

J

JAAS (Java Authentication and Authorization Service) 4
jasperreports
  installation 92
Java Authentication and Authorization Service. See JAAS 4
Java messaging service. See JMS 6
Java naming and directory interface. See JNDI 7
java plugin
  clearing cache 158
Java protocol
  disabling 6
    EJB 6
    HTTP 6
    JMS 6
Java protocols
  securing 7
    COM+ 8
    EJB 7
    HTTP API tester 8
JavaProtocolSecurityMeasures 5
JavaServer pages. See JSP 7

JBoss
creating EAR
steps involved 152
using Web services 143
jcert.jar 111
JMS (Java messaging service) 6
JNDI (Java naming and directory interface) 7
jnet.jar 111
JSP (JavaServer Pages) 7
jsse.jar 111
JVM settings
for WebSphere 126

L

LAN (local area network) 5
language pack
creating ear 84
creating EAR 84
deploying EAR 84
installing 81
loading factory defaults 81, 82
loading language pack translations 82, 83
switching base language 84
languages, supported 84, 85
LDAP (lightweight directory access protocol) 4
lightweight directory access protocol. See LDAP 4
Linux
installing (from command line) 64, 66
installing (from silent install file) 68, 76
silent installation file parameters 70
UNIX accounts 62
LLM (loftware label manager) 91
loading factory defaults,database factory defaults 94
local area network. See LAN 5
locales 84, 85
localization 84, 85
loftware label manager 91
bar code label 91
defining printers 92
designing label 91
for UNIX systems 92
supported modes 91
log files
best practices, security 9
LPM (loftware print server) 91

N

network topology
deployment over internet 5
deployment over LAN 5
deployment over VPN 5
non-English language 84, 85

O

operating system permissions 9
Oracle
copying JDBC driver 22
installation 21, 33
running create instance 21, 33
Oracle databases 24, 33
Windows 41

P

post installation recommendations 9, 54, 76
pre-compiling JSP files 109
Preparing the Database Server 21, 31
print server
configuration utility 91
installing loftware components 91
print server installation 91
procuring storage 18

R

recommended data encryption
SSL 128-bit 4
VPN 3DES 4
remote method invocation. See RMI 7
restoring
databases 31
retention policy 18
RMI (remote method invocation) 7
running
Oracle scripts 28

S

security
database security 10
deployment architecture 3
Java protocol See Java protocol 5
web security See web security 8
security plan
creating 3
Selling and Fulfillment Foundation
deploying 109
session replication 112
including session descriptors 112
setting up
dbverify utility (single schema) 96
Oracle scripts 24
Web services 113, 128, 143
WebLogic scripts 110
silent installation file
parameters for Linux 70
parameters for UNIX 70
parameters for Windows 48
SIM installation
defining printers 92
sma.war 113, 128, 142
smcfs.war 113, 128, 142
smcfsejb.jar 113, 128
smcfswsbe.jar 113, 128, 143
stack traces
eliminating 111
Sterling Sensitive Data Capture Server 79
System Console 127

T

time-triggered transactions
override agent criteria 104
transaction data
truncating 103
transactional database 18
truncating transaction data 103
types of order 18

U

UNIX
accounts 62
installing 69
installing (from command line) 64, 66
installing (from silent install file) 68, 76
silent installation file parameters 70
UNIX accounts
Linux 62
UNIX 62
Utilities
development 102
transaction data truncation tool 103
runtime 103
agent server 103
agent trigger 104
Integration server 103
setting up 105

V

verifying database
extra objects 99
verifying database (single schema) 96
virtual private network. See VPN 5
VPN (virtual private network) 5

W

web security 8
routing 9
session security 9
web server executables 9
writing log files 9
Web services 113, 128, 143
WebLogic
creating EAR
file descriptors 122
steps involved 122
eliminating stack traces 111
setting client character display 158
setting HTTP in-memory session replication 112
using Web services 113
WebLogic application server
setting up 109, 110, 125
weblogic.xml 112
WebSphere
avoiding FileNotFound exceptions 125
avoiding warning messages 125
configuring JVM settings 126

WebSphere *(continued)*
creating EAR
steps involved 137
invoking EJBs 126
making EJB calls 126
specifying memory parameters 125
using Web services 128
Windows
installing 42, 47
installing (from command line) 44
silent installation file parameters 48

Y

yantraejb.jar 126
yantrawebservices.war 113, 128

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing

Legal and Intellectual Property Law

IBM Japan Ltd.

19-21, Nihonbashi-Hakozakicho, Chuo-ku

Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be

incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

J46A/G4

555 Bailey Avenue

San Jose, CA 95141-1003

U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© IBM 2013. Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 2013.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at http://www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium and the Ultrium Logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Connect Control Center®, Connect:Direct®, Connect:Enterprise®, Gentran®, Gentran®:Basic®, Gentran:Control®, Gentran:Director®, Gentran:Plus®, Gentran:Realtime®, Gentran:Server®, Gentran:Viewpoint®, Sterling Commerce™, Sterling Information Broker®, and Sterling Integrator® are trademarks or registered trademarks of Sterling Commerce®, Inc., an IBM Company.

Other company, product, and service names may be trademarks or service marks of others.

Printed in USA