Sterling Multi-Channel Selling Solution

Implementation Guide

Release 8.0.1

Sterling Commerce
An IBM Company

THE APACHE SOFTWARE FOUNDATION SOFTWARE

The Sterling Commerce Software is distributed with or on the same storage media as the following software products (or components thereof): Apache Ant v1.6.5, Apache Axis v1.4, avalon-framework-4.0.jar, batik-1.5-fop-0.20-5.jar, Apache Jakarta Commons Collections v2.1, Apache Commons EL v1.0, Apache Commons Logging v1.0.4, Apache FOP v0.20.5, Apache Jakarta Regexp v1.4, Apache log4j v1.2.8, Apache Lucene v2.3, Apache Standard Taglib 1.1, Apache Web Services Invocation Framework (WSIF) v.20, Apache Xalan v2.7.0, Apache Xerces v2.8.0, xml-apis-01.3.03.jar, commons-codec-1.2.jar, commons-httpclient-3.0.1.jar (collectively, "Apache 2.0 Software"). Apache 2.0 Software is free software which is distributed under the terms of the Apache License Version 2.0. A copy of License Version 2.0 is found in the following locations and applies only to the individual pieces of the Apache 2.0 Software found in the directory location(s) specified below for that copy of License Version 2.0:

<installdir>\thirdpartylicenses\Apache_Ant_1.6.5_license_OrderSelling.doc applies to the Apache 2.0 Software located at <installdir>\WEB-INF\lib\ ant-1.6.5.jar;

<installdir>\thirdpartylicenses\Apache_Axis_v1.4_license_OrderSelling.doc applies to the Apache 2.0 Software located at <installdir>\WEB-INF\lib\ axis.jar, <installdir>\WEB-INF\lib\wsdl4j-1.5.1.jar, <installdir>\WEB-INF\lib\saaj.jar, <installdir>\WEB-INF\lib\jaxrpc.jar, <installdir>\WEB-INF\lib\commons-discovery-0.2.jar;

<installdir>\thirdpartylicenses\Apache_Avalon_Framework_4.0_license_OrderSelling.doc applies to the Apache 2.0 Software located at <installdir>\WEB-INF\lib\avalon-framework-4.0.jar;

<installdir>\thirdpartylicenses\Apache_FOP_0.20.5_license_OrderSelling.doc applies to the Apache Software located at <installdir>\WEB-INF\lib\batik-1.5-fop-0.20-5.jar;

<installdir>\WEB-INF\lib\Apache_Commons_Collections_2.1_license_OrderSelling.doc applies to the Apache 2.0 Software located at <installdir>\WEB-INF\lib\commons-collections-2.1.jar

<installdir>\thirdpartylicenses\Apache_Commons_EL_1.0_license_OrderSelling.doc applies to the Apache 2.0 Software located at <installdir>\WEB-INF\lib\commons-el-1.0.jar;

<installdir>\thirdpartylicenses\Apache_Common_ Logging_1.0.4_license_OrderSelling.doc applies to the Apache 2.0 Software located at <installdir>\WEB-INF\lib\commons-logging-1.0.4.jar;

<installdir>\thirdpartylicenses\Apache_FOP_0.20.5_license_OrderSelling.doc applies to the Apache 2.0 Software located at <installdir>\WEB-INF\lib\fop-0.20.5.jar;

<installdir>\thirdpartylicenses\Apache_Jakarta_Regexp_1.4_license_OrderSelling.doc applies to the Apache 2.0 Software located at <installdir>\WEB-INF\lib\jakarta-regexp-1.4.jar;

<installdir>\thirdpartylicenses\Apache_log4j_1.2.8_license_OrderSelling.doc applies to the Apache 2.0 Software located at <installdir>\WEB-INF\lib\log4j-1.2.8.jar;

<installdir>\thirdpartylicenses\Apache_Lucene_2.3_license_OrderSelling.doc applies to the Apache 2.0 Software located at <installdir>\WEB-INF\lib\lucene-core-2.3.0.jar, <installdir>\WEB-INF\lib\lucene-demos-2.3.0.jar;

<installdir>\thirdpartylicenses\Apache_Standard_Taglib_1.1_license_OrderSelling.doc applies to the Apache 2.0 Software located at <installdir>\WEB-INF\lib\standard.jar;

<installdir>\thirdpartylicenses\Apache_WSIF_2.0_license_OrderSelling.doc applies to the Apache 2.0 Software located at <installdir>\WEB-INF\lib\wsif.jar;

<installdir>\thirdpartylicenses\Apache_Xalan_2.7.0_license_OrderSelling.doc applies to the Apache 2.0 Software located at <installdir>\WEB-INF\lib\xalan-2.7.0.jar

<installdir>\thirdpartylicenses\Apache_Xerces_2.8_license_OrderSelling.doc applies to the Apache 2.0 Software located at <installdir>\WEB-INF\lib\xercesImpl-2.8.0.jar;

<installdir>\thirdpartylicenses\Apache_xml_apis_1.3.03_license_OrderSelling.doc applies to the Apache 2.0 Software located at <installdir>\WEB-INF\lib\xml-apis-1.3.03.jar

Unless otherwise stated in a specific directory, the Apache 2.0 Software was not modified. Neither the Sterling Commerce Software, modifications, if any, to Apache 2.0 Software, nor other Third Party Code is a Derivative Work or a Contribution as defined in License Version 2.0. License Version 2.0 applies only to the Apache 2.0 Software located in the specified directory file(s) and does not apply to the Sterling Commerce Software or to any other Third Party Software.

BEANSHELL SOFTWARE

CYBERSOURCE SOFTWARE

EHCACHE SOFTWARE AND JINI SOFTWARE

GETOPT SOFTWARE AND HTTPCLIENT SOFTWARE

JUNIT SOFTWARE

Sterling Commerce, Inc.

4600 Lakehurst Court Dublin, OH 43016-2000 * 614/793-7000

Preface

Welcome to the Sterling Multi-Channel Selling Solution. This *Implementation Guide* and the associated documentation provides all the information required for you to implement the Sterling Multi-Channel Selling Solution at your enterprise.

Purpose

This guide provides step-by-step instructions to install and implement the Sterling Multi-Channel Selling Solution. On completion, you should be able to verify that the system is up and running, and that you can perform the basic administration steps described in the accompanying *Administration Guide*.

Audience

This guide presupposes an advanced level of information systems knowledge, familiarity with basic network and database concepts, and Java for certain implementation steps.

Conventions

Throughout this guide, we will use the following conventions shown in Table 1, "Conventions", on page viii:

TABLE 1. **Conventions**

Type	Convention
File names	**Sample.txt**
Paths and directory names	**/top_level/next_level/next_level/destination_directory/**
Sample code extracts	`public void method(String s)`
Values to be provided	*`<value supplied by developer>`*

Contents

CHAPTER 1

Architecture and Configuration Overview

This chapter presents an overview of the Sterling Multi-Channel Selling Solution. It provides a brief description of the underlying technology and architecture and discusses the parts of the system that you need to customize to meet the needs of your installation. It also presents a description of the configurations for the Sterling Multi-Channel Selling Solution.

Architecture

The Sterling Multi-Channel Selling Solution is designed to conform to the Java 2 Platform, Enterprise Edition (J2EE) architecture as defined in *Java 2 Platform Enterprise Edition Specification, v 1.2* published by Sun Microsystems, Inc. The Sterling Multi-Channel Selling Solution server architecture is illustrated schematically in "Logical Representation of the Sterling Multi-Channel Selling Solution Server Architecture" on page 2.

The Sterling Multi-Channel Selling Solution is deployed as a Web application that comprises a set of Java classes together with accompanying configuration files, HTML templates, and JSP pages. It must be installed into a servlet container that conforms to the J2EE standard. You can use an existing servlet container that conforms to the standard or deploy the Sterling Multi-Channel Selling Solution using the servlet container that we provide as part of the distribution software. See CHAPTER 4, "Installation Requirements" for further information.

The Sterling Multi-Channel Selling Solution is designed to conform to the Model 2 architecture. In this architecture, three functional components referred to as the Model, View, and Controller (MVC) partition the functionality of the server into logically distinct components.

- *Model*: this component manages the data and business objects that are used by the system.
- *View*: this component is responsible for generating the content displayed to the user.
- *Controller*: this component determines the logical flow of the application. It determines what actions are performed on the model and manages the communication between model and view components.

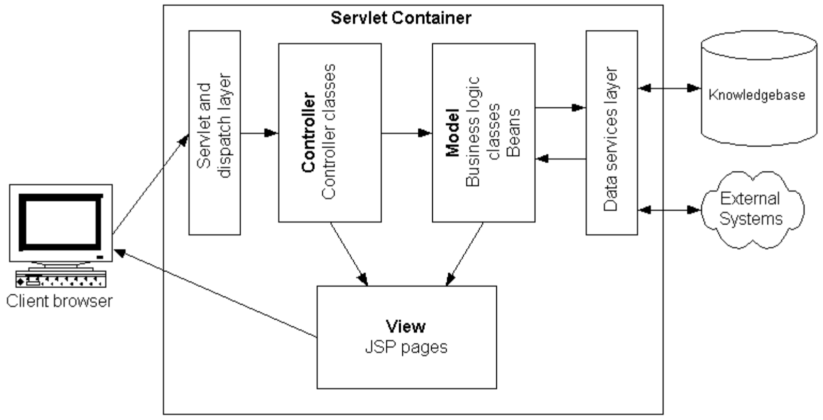


FIGURE 1. Logical Representation of the Sterling Multi-Channel Selling Solution Server Architecture

The Sterling Multi-Channel Selling Solution is designed to be flexible and extensible. You tailor the following components of the Sterling Multi-Channel Selling Solution as part of the implementation of your system.

TABLE 1. Implementation Components

Component	Function
JSP pages	Customize the JSP pages that determine the look and feel of the Web pages for end-users.
XML schema and data objects	Define the data object schema as a set of XML files. These specify the structure of the data objects and the data sources that provide their content.
Business logic and BizAPI classes	These Java classes determine the business logic that processes requests and messages.
Controller classes	These Java classes handle incoming requests from customer browsers and determine how the responses are displayed.
Configuration files	Use the configuration files to determine the properties of the Sterling Multi-Channel Selling Solution and control how incoming requests and messages are processed.

Implementation details are covered in the following chapters.

Configurations

This section provides a description of the possible configurations for the Sterling Multi-Channel Selling Solution.

Platform Edition

This configuration provides:

- Sterling Commerce Manager: the basic server that supports the data services layer, message passing, and system administration.

Enterprise Edition

This configuration comprises the base functionality of the Sterling Multi-Channel Selling Solution. It supports:

- Sterling Product Manager: this component supports the administration interface to manage the product catalog and products.
- Sterling Profile Manager: this component supports the administrative interface to the Knowledgebase to manage partner profiles and users.
- Sterling Quotes: this component supports the e-commerce interface to end-users.

Sterling Multi-Channel Selling Solution Applications

In addition to the Enterprise Edition package, the Sterling Multi-Channel Selling Solution provides a number of different applications. These components can be licensed and installed separately. They are:

- Sterling Advisor: this component provides the administrative interface to the guided selling interface of the Sterling Quotes component. It also provides the customer-facing applications that provide customers with the ability to navigate through your product catalog and to search the catalog.
- Sterling Analyzer: this component provides a set of sophisticated reporting capabilities to analyze the commerce activity on your Sterling Multi-Channel Selling Solution.
- Sterling Configurator: this component provides support for the creation of configurable products and their use as part of the end customer experience.
- Sterling Integrator: this component provides the means to integrate the Sterling Multi-Channel Selling Solution with your existing ERP systems.
- Sterling Promotions: this component provides the administrative interface to manage promotions and their assignment to products.
- Sterling MarketLink: this component provides support for integrating the Sterling Multi-Channel Selling Solution with external systems such as Ariba procurement systems.
- Sterling Orders: this component provides support for the direct commerce capabilities of the Sterling Multi-Channel Selling Solution. It provides the functionality for customers to place orders, modify or cancel orders, and

for customer service representatives to review and modify customer orders.

- Sterling Pricing: this component provides the administrative interface to manage pricing and price lists.
- Sterling Returns: this component provides support for managing customer returns.

Sterling Partner.com Configuration

You can implement the Sterling Multi-Channel Selling Solution to host a number of different partners each of which operate separately from each other. All hosted partners are supported as Partner.com partners in the same Knowledgebase. See the *Sterling Multi-Channel Selling Solution Administration Guide* for more information.

CHAPTER 2

Implementation Overview

This chapter presents an overview of implementing the Sterling Multi-Channel Selling Solution. Its intended audience is system integrators and IT professionals charged with successfully executing an implementation of the Sterling Multi-Channel Selling Solution. It covers both the installation of the system and the steps required to integrate the system with existing e-commerce and ERP systems.

Implementation Tasks

This section describes a suggested methodology for implementing the Sterling Multi-Channel Selling Solution and an outline of the implementation steps.

Implementation Methodology

The Sterling Multi-Channel Selling Solution implementation methodology consists of phases that ensure that implementation can be planned and tracked through to completion. Table 2, "Sterling Multi-Channel Selling Solution Implementation Methodology", on page 8 provides a summary of the phases and the activities to complete in each phase.

A standard set of documents can be used to track each phase. Contact Sterling Commerce for further information.

TABLE 2. **Sterling Multi-Channel Selling Solution Implementation Methodology**

Implementation phase	**Description**
Plan	Plan the implementation: set a timeline, milestones, and identify risks and dependencies
Analyze	Organization and administration, define business rules, user interface, messaging protocols, data sources, e-commerce flow planning, training needs, rollout strategy, environment preparation, operations planning
Design and configure	Installation, configuration, integration, unit testing, and training development
Test and deploy	Testing server configuration, enterprise to partner communication, partner to enterprise communication; cut over to production systems, distributor training, documentation delivery, support
Improve	Ongoing enhancement activities, partner training, and support

Steps to Implementation

The main tasks you perform in implementing Sterling Multi-Channel Selling Solution are:

- *Project analysis:* agree to a schedule for the implementation project that sets a timeline. Identify milestones to measure the progress of the implementation and identify dependencies and risks that might prevent the implementation from completing on time.
- *Configuration analysis*: determine a suitable Sterling Multi-Channel Selling Solution configuration (the number of machines to be used and their location on internal networks in relation to firewalls and proxy servers). See "High Availability and Load Balancing" on page 11 for further details about a clustered implementation.
- *Integration analysis*: identify integration points with existing e-commerce systems.
- *Requirements analysis*: check hardware and software requirements to make sure that the machines are sufficiently powerful to support the

anticipated traffic and response times required. See CHAPTER 4, "Installation Requirements" for more information.

- *Installation of Sterling Multi-Channel Selling Solution*: install the Sterling Multi-Channel Selling Solution on the designated machine(s). See CHAPTER 5, "Installing the Sterling Multi-Channel Selling Solution" for more information.
- *Knowledgebase setup*:

    a. Installation of Knowledgebase: installing the Knowledgebase schema in the designated database server.

    b. Knowledgebase setup: checking connectivity to the Knowledgebase database server and populating it with all your e-commerce-related information. This must include the partner profiles for your partners, your product catalog, and price list information.

    See CHAPTER 6, "Creating and Populating the Knowledgebase" for more information.
- *Sterling Multi-Channel Selling Solution configuration*: modify configuration files to define the system configuration in your production environment. See CHAPTER 9, "Customizing Your Sterling Multi-Channel Selling Solution" for more information.
- *Role and security definition*: define groups and roles and modify configuration files and ACL scripts accordingly. These determine the security privileges for your enterprise server users. See the *Reference Guide* for more information.
- *Schema creation*: create the business object schema to provide data source information. The data layer manages access between the enterprise server and the external systems. See the *Reference Guide* for more information.
- *Customizing BLCs and controllers*: modify business logic and controller classes to support your business logic. In some cases, you need to modify the Java classes in order to implement business processes specific to your organization.
- *Customizing JSP pages*: modify templates to meet your “look-and-feel”, search, and static page requirements. The JSP pages provided by the Sterling Multi-Channel Selling Solution are used to display the browser pages and may be customized to meet the needs of your organization.

- *Product integration*: import product information into the Knowledgebase or provide punch-out integration. If your implementation is to support product ordering from a non-Sterling product, then you need to provide a means of integrating the product data with the Sterling Multi-Channel Selling Solution.
- *Testing server configuration*: before you deploy the Sterling Multi-Channel Selling Solution, thoroughly test the system. We provide a number of scripts to test the chief functional components.
- *Testing enterprise to partner communication*: send test messages from the enterprise server to other enterprise servers.
- *Testing partner to enterprise communication*: send test messages from other enterprise servers to your enterprise server.
- *Assess and enhance*: once the Sterling Multi-Channel Selling Solution is deployed you must plan for an ongoing process of analyzing its usage and performance.

Implementing the Sterling Multi-Channel Selling Solution Integration

The Sterling Multi-Channel Selling Solution is designed to integrate channel partners into an e-commerce network. Organizations in the network act as enterprises and partners. Each organization acting as an enterprise installs their copy of the enterprise server to transfer information to their channel partners seamlessly.

Each reseller or distributor may work with more than one enterprise, and their installation of the enterprise server must be able to receive and respond to messages from different enterprise servers. The following table summarizes the main activities for an implementation of the Sterling Multi-Channel Selling Solution.

TABLE 3. **Implementation Tasks**

Implementation phase	Task
Plan	Project analysis
Analyze	Configuration analysis
	Integration analysis
	Requirements analysis

TABLE 3. **Implementation Tasks (Continued)**

Implementation phase	Task
Design and configure	Preparation of servlet container environment
	Installation of Sterling Commerce Manager
	Installation of Knowledgebase
	Knowledgebase setup
	Sterling Multi-Channel Selling Solution configuration
	Role and security definition
	System administrator authentication
	XML schema creation
	Customizing of BizAPIs, BLCs, and controllers
	Customizing JSP pages
Testing and deployment	Product integration
	Testing server configuration
	Testing enterprise to partner communication
	Testing partner to enterprise communication
	Release to production systems
Improve	Assess and enhance

High Availability and Load Balancing

The Sterling Multi-Channel Selling Solution supports the ability to distribute request-handling over a number of machines. An enterprise server uses the load-balancing capabilities of the servlet container used to implement the Sterling Multi-Channel Selling Solution. Consult your servlet container documentation to see what options are available to you.

Integration Security Issues

Take special care to address security issues. Begin implementation only after you have addressed how users of the Sterling Multi-Channel Selling Solution will access data provided by you and your partners.

This discussion should cover:

- authentication questions including the use of LDAP

- the use of encryption in storing data in the Knowledgebase
- the use of encryption schemes across your networks and the Internet
- direct and indirect access to ERP systems
- your existing firewalls and proxy servers

See CHAPTER 12, "General Security Considerations" for more information.

CHAPTER 3

Installation Worksheet

This chapter presents a worksheet to help you gather the information that you need to install and configure the Sterling Multi-Channel Selling Solution.

Attention: If you do not have this information, then you will not be able to install and run the Sterling Multi-Channel Selling Solution.

- Which servlet container are you going to use for the Sterling Multi-Channel Selling Solution? What release is this servlet container?

  ____

- What version of the Java Servlet Specification does the servlet container support?

  ____

- What is the root directory of the servlet container installation? This is referred to as ***container_home*** throughout the documentation.

  ____

- What Java Runtime Environment (JRE) are you using? Where is its JAVA_HOME and JDK_HOME?

  ____

- What is the database server to be used for the Sterling Multi-Channel Selling Solution Knowledgebase?

___

- What JDBC URL will you use to connect to the Sterling Multi-Channel Selling Solution Knowledgebase database server?
  - You must connect to an Oracle Server using an Oracle JDBC driver.
  - You must connect to a Microsoft SQL Server using a Microsoft SQL Server JDBC driver.

___

- What is the username and password to be used to connect to the database server?

___

- What name will you choose for the servlet context to be used for the Sterling Multi-Channel Selling Solution?

___

CHAPTER 4

Installation Requirements

This chapter presents a description of the hardware, software, and network requirements to install the Sterling Multi-Channel Selling Solution. Make sure that your system meets these requirements before you begin installing the Sterling Multi-Channel Selling Solution. See:

- "Hardware Requirements" on page 15
- "Software Requirements" on page 16
- "Network Requirements" on page 19
- "Browser Requirements" on page 19
- "Database Server Requirements" on page 21

Pay special attention to performance requirements and what requirements that will drive in terms of hardware needs. See "Sizing Requirements" on page 24 for more information.

Hardware Requirements

This section provides a description of the minimum hardware requirements of the Sterling Multi-Channel Selling Solution.

Windows 2000

- 512 MB of RAM
- Single or dual Intel processors rated at 400 MHz or faster

UNIX

- 512 MB of RAM
- Single or dual processors rated at 400 MHz or faster

Software Requirements

This section provides a description of the software requirements of the Sterling Multi-Channel Selling Solution.

Operating Systems

Hewlett-Packard UNIX

- HP-UX 11.iv3.

Before deploying the Sterling Multi-Channel Selling Solution on HP-UX, you must apply the JavaOOB bundle provided by Hewlett-Packard. See the following URL for further information:

```
http://www.hp.com/products1/unix/java/java2/outofbox/infolibrary/
release_notes_java_oob.html
```

JavaOOB is a stand-alone bundle that upon installation, installs startup (RC) scripts, modifies kernel parameters, rebuilds the kernel, and reboots the system. During startup, the startup scripts modify system tunables.

For the user used to run the servlet container, you must increase the number of files that the user may have open. Do this by running the ulimit command as follows:

```
>ulimit –Sn 1028
```

IBM AIX

- AIX 5.3 ML4.

Linux

- Red Hat Enterprise Linux 5.0.

Microsoft Windows

- Windows 2003 Server with Service Pack 1 or Windows 2000 Server or Professional with Service Pack 4.

Sun SPARC Solaris

- Sun SPARC Solaris 9 operating environment or subsequent compatible version: see "Patch Information for the Solaris Platform" on page 391 for information relating to required Solaris patches.

Java Development Kit

You must use either JDK 1.5 or the most recent version of JDK 1.4.2. If you use JDK 1.4.2, your version must be at least 1.4.2_06; however, you should use the most current version. Problems have been reported using 1.4.2_03 because it uses incompatible keystores.

SDK

You must use JDK 1.5 to use the SDK to install the Sterling Multi-Channel Selling Solution and to create customizations using the SDK.

Servlet Containers

The Sterling Multi-Channel Selling Solution has been certified to run in the servlet containers listed in the following table. Install your servlet container before installing the Sterling Multi-Channel Selling Solution. Follow and complete the installation instructions for your selected servlet container. We recommend using Tomcat to test your implementation of the Sterling Multi-Channel Selling Solution before deploying it to your production system.

TABLE 4. Servlet Container Support

Servlet Container	**Vendor**	**Release**	**Servlet Specification Support**
Tomcat	Open Source	6.0.14 with JDK 1.5 On Windows installations, you must set the JVM used to the client **jvm.dll**	2.3
WebLogic	BEA Systems	10 with JDK 1.5	2.3
WebSphere	IBM	6.1	2.3

The Sterling Multi-Channel Selling Solution is designed to run in any J2EE-compliant servlet container. Contact Sterling Commerce regarding installing your Sterling Multi-Channel Selling Solution in another servlet container that meets this specification.

Servlet Container Clustering

The Sterling Multi-Channel Selling Solution is designed as a fully J2EE-compliant Web application capable of being deployed in any servlet container that supports the J2EE standard. It can be deployed to all servlet containers that are operating within a cluster or it can be deployed to independent servlet containers that are operating behind a load-balancing solution such as Cisco Local Director.

See "High Availability and Clustering" on page 93 for more information on implementing a clustered solution. See CHAPTER 26, "Installing a Clustered Implementation" for more information about setting up clustered implementations:

- See "Setting Up a WebLogic Cluster" on page 381 for information relating to WebLogic clustering.

Network Requirements

The Sterling Multi-Channel Selling Solution machine(s) must also be able to establish a JDBC or an ODBC connection to the database server that is used in conjunction with the Sterling Multi-Channel Selling Solution. You must ensure that the appropriate database client software is installed on the Sterling Multi-Channel Selling Solution machine. See CHAPTER 5, "Installing the Sterling Multi-Channel Selling Solution" for more information.

Browser Requirements

To access and use the enterprise administration pages, users must run Internet Explorer 6.0 SP2 or subsequent compatible versions, or Firefox 1.5.x or 2.0, or Netscape 7.x and subsequent compatible versions. This requirement includes partner users performing administrative tasks on the Sterling Multi-Channel Selling Solution.

All of the external customer-facing pages support Internet Explorer 5.5 and Netscape Navigator 7.0 and subsequent compatible versions.

Security Settings

You must enable your browser to support scripting.

Firefox

1. Select **Options** from the Tools menu.
2. Click the **Content** tab.
3. If the **Enable Java** and **Enable Javascript** check boxes are not already checked, then check them.
4. Click **OK**.

Internet Explorer

1. Select **Internet Options...** from the Tools menu.
2. Click the **Security** tab.
3. Click **Custom Level...**.
4. Under **Scripting**, make sure that Active scripting is enabled.
5. Click **OK**.

6. Click **OK**.

Mozilla

1. Select **Preferences...** from the Edit menu.
2. Select **Advanced**, then select **Scripts & Plug-ins**.
3. If the **Enable JavaScript for Navigator** check box is not already checked, then check it.
4. Click **OK**.

Netscape Navigator

1. Select **Preferences...** from the Edit menu.
2. Select **Advanced**.
3. If the **Enable Javascript for Navigator** check box is not already checked, then check it.

Character Sets

Bear in mind that browsers used by Sterling Multi-Channel Selling Solution users must support the character sets required to display the data correctly. If your implementation of the Sterling Multi-Channel Selling Solution manages data from non-ASCII character sets, then make sure that the browser is set to support Unicode characters.

In particular, make sure that dialog boxes use fonts that support these characters. On Windows systems, this is set using the Display Properties control panel applet.

1. Select **Start -> Settings -> Control Panel**, and start the Display applet. Alternatively, right click the Desktop background and select **Properties**.
2. Click **Appearance**.
3. Select Message Box from the **Item** drop-down list.
4. Select a Unicode font, for example Arial Unicode MS.
5. Click **Apply**.

Database Server Requirements

This section lists the general set-up requirements for Oracle and SQL Server databases. See CHAPTER 5, "Installing the Sterling Multi-Channel Selling Solution", for details.

The Sterling Multi-Channel Selling Solution requires one of the following database servers to act as the Knowledgebase database:

- Oracle 10g
- Microsoft SQL Server 2005

Attention: We recommend that you run the database server on a separate machine from the Sterling Multi-Channel Selling Solution.

You must ensure that there is a valid userid (username/password pair) set up on the database that acts as the authenticated userid for all Sterling Multi-Channel Selling Solution connections to the database. This userid must have the necessary privileges to create, modify, and execute database objects. Make sure that the database default character set is set to UTF-8 Unicode.

Make sure that you have the appropriate client tools installed on the Sterling Multi-Channel Selling Solution machine(s). In particular, make sure that you have or can obtain the appropriate JDBC library files from Microsoft or Oracle if you plan to deploy against SQL Server or Oracle.

The Sterling Multi-Channel Selling Solution involves collecting data about your users on a transactional database, and transferring that data to a segmentation database for processing and re-use in marketing activities. You configure your Sterling Multi-Channel Selling Solution implementation to provide efficient processing of user data and to enable communication between the transactional and segmentation databases. You configure the transactional and segmentation databases to suit your business requirements, and you can choose not to use the segmentation feature.

Database Owner Requiements

The database owner must have the following privileges:

- CREATE TABLE
- CREATE VIEW
- CREATE SYNONYM

- CREATE DATABASE LINK (on the segmentation database)
- CREATE TRIGGER
- CREATE SESSION
- CREATE PROCEDURE
- CREATE SYNONYM
- UNLIMITED TABLESPACE

The database owner must have the following roles:

- CONNECT
- RESOURCE

Microsoft SQL Server Requirements

This section describes requirements for running the Sterling Multi-Channel Selling Solution with Microsoft SQL Server.

You must have a Microsoft SQL Server Release 2005 or subsequent compatible version running on Windows 2003 Server. You must connect to this SQL server using the JDBC drivers provided by Microsoft. You can download the JDBC drivers from: http://msdn.microsoft.com/data/ref/jdbc/ (note that this URL is subject to change).

Configure the SQL Server to not return UPDATE counts. You can do this by executing the following commands in the SQL Server Management Studio:

```
USE master;
GO
EXEC sp_configure 'disallow results from triggers', '1';
RECONFIGURE WITH OVERRIDE;
```

General Requirements

When you set up the SQL Server, you must specify that the database character set is Unicode. You must also set up the SQL Server client software on the Sterling Multi-Channel Selling Solution machine to use Unicode. On the servlet container machine:

1. Start the SQL Server Client Network Utility.
2. On the DB-Library Options tab, uncheck **Automatic ANSI to OEM conversion**.
3. Click **Apply**, and then **OK**.

If you use SQL Server, then a search that includes the following characters will return zero results: é, ö, ü, ç.

SQL Server 2005 Communication Requirements

To enable communication between the transactional and segmentation databases on two separate servers, follow the steps specified in CHAPTER 5, "Installing the Sterling Multi-Channel Selling Solution", in the section "SQL Server 2005 Setup" on page 39.

Sterling Analyzer

There is a known issue with the use of non-ASCII characters and their display in Sterling Analyzer reports. If you plan to use SQL Server for the Knowledgebase and support multiple locales, contact your Sterling Commerce representative for more information.

Oracle Requirements

This section describes the requirements for running the Sterling Multi-Channel Selling Solution with Oracle 10g using the Oracle 10g JDBC driver.

If you do not have a local installation of Oracle products on the installation machine, download the appropriate JAR file from the Oracle Technology Network Web site. The URL is:

```
http://www.oracle.com/technology/index.html
```

Search for “JDBC driver”.

When setting up the database, select the Custom option in the Database Configuration Assistant in order to set the character set to UTF-8. You can verify that the correct settings are set by invoking a SQL*PLUS session to the database server and entering:

```
SELECT * FROM NLS_DATABASE_PARAMETERS WHERE PARAMETER =
'NLS_CHARACTERSET';
```

You should see:

```
NLS_CHARACTERSET UTF8
```

UNIX

To use the Oracle OCI driver to connect from the Sterling Multi-Channel Selling Solution to the database server, set the following environment variable:

NLS_LANG=AMERICAN_AMERICA.UTF8

Oracle Communication Requirements

To enable communication between the transactional and segmentation databases on two separate servers, follow the steps specified in CHAPTER 5, "Installing the Sterling Multi-Channel Selling Solution", in the section "Oracle Setup" on page 36.

You must also set up a TNSNAMES.ora entry on the transactional database server for the segmentation database server, then set up a TNSNAMES.ora entry on the segmentation database server for the transactional database server.

See CHAPTER 5, "Installing the Sterling Multi-Channel Selling Solution", in the section "Oracle Setup" on page 36, for details.

Sizing Requirements

The Sterling Multi-Channel Selling Solution provides a sizing tool to help select hardware and software that will best meet your implementation needs. This is the *Sterling Multi-Channel Selling Solution Capacity Planning Guide* provided with the SDK.

CHAPTER 5

Installing the Sterling Multi-Channel Selling Solution

This chapter describes how to install the Sterling Multi-Channel Selling Solution and deploy either the reference implementation or the minimal implementation. This chapter does not cover the customization work necessary to meet the needs of your implementation. See CHAPTER 9, "Customizing Your Sterling Multi-Channel Selling Solution", for details about customization.

Most implementations of the Sterling Multi-Channel Selling Solution use the SDK to perform installation, and you must use the SDK to manage the customizations to your implementation. You can also install the reference Web application without using the SDK: this is useful for initial sanity testing and to verify the proposed deployment environment. See "Installing the Reference Sterling Multi-Channel Selling Solution" on page 69.

You can install Release 8.0.1 as a full install, or by upgrading from Release 8.0 to Release 8.0.1. If you are upgrading from a release previous to Release 8.0, you must first upgrade to Release 8.0 before proceeding.

Attention: The Sterling Multi-Channel Selling Solution is a complex product. Follow these instructions carefully.

This chapter covers the following topics:

- "Installation Overview" on page 26
- "Preparing to Install" on page 28

- "Known Issues" on page 31
- "Configuring the Transactional and Segmentation Databases" on page 31
- "Installing the Software Development Kit" on page 41
- "Installing the Sterling Multi-Channel Selling Solution Using the SDK" on page 43
- "Deploying the Sterling Web Application" on page 58
- "Matrix Reference Segments Setup" on page 68
- "Installing the Reference Sterling Multi-Channel Selling Solution" on page 69
- "Default XML Identity Setup" on page 79
- "Database Server Steps" on page 84
- "Managing Database Connections" on page 86
- "Pagination Settings" on page 89
- "Setting the Product Catalog Root" on page 90
- "Setting the Session Timeout" on page 90
- "Modifying the URL for the Web application DTD" on page 91
- "Managing Memory" on page 92
- "Configuring Ehcache" on page 92
- "High Availability and Clustering" on page 93
- "Sharing Directories" on page 93
- "Directory and File Organization" on page 94
- "Cron Job Setup" on page 96
- "Setting Up Apache as a Front-end to Tomcat" on page 97
- "Filtering Static Content" on page 101
- "Compressing Output From the Sterling Multi-Channel Selling Solution" on page 103

Installation Overview

Installing the Sterling Multi-Channel Selling Solution involves these stages:

- Preparing to Install
- Installing the Software Development Kit
- Installing the Sterling Multi-Channel Selling Solution Using the SDK: follow either
  - "To Install the Sterling Multi-Channel Selling Solution as a Full Release" on page 43
  - "To Install the Sterling Multi-Channel Selling Solution by Upgrading from an Earlier Release" on page 50
- Deploying the Sterling Web Application
- Database Server Steps

Once you complete these stages, continue to CHAPTER 6, "Creating and Populating the Knowledgebase".

Preparing to Install

This stage covers how to prepare for an efficient installation: identifying known issues, determining how you will configure your transactional and segmentation databases, identifying the database information you will need, identifying the servlet container root directory and the destination directory for the **Sterling.war** file, and so on. See "Preparing to Install" on page 28.

Installing the Software Development Kit

This stage covers how to install the Sterling Multi-Channel Selling Solution Software Development Kit (SDK). After you install the SDK, you can manage the installation of the Sterling Multi-Channel Selling Solution using targets provided by the SDK. See "Installing the Software Development Kit" on page 41.

Installing the Sterling Multi-Channel Selling Solution Using the SDK

This stage covers how to use the SDK to install the Sterling Multi-Channel Selling Solution Web application. After you complete this stage, you deploy the **Sterling.war** to a servlet container as a web application. See "Installing the Sterling Multi-Channel Selling Solution Using the SDK" on page 43.

Deploying the Sterling Web Application

This stage covers how to deploy the **Sterling.war** file as a Web application. How you deploy depends on which servlet container you are using. See "Deploying the Sterling Web Application" on page 58.

Database Server Steps

This stage covers the preliminary configuration steps you perform to prepare for the steps covered in CHAPTER 6, "Creating and Populating the Knowledgebase". The preparation steps require some knowledge of the database server that you plan to use for the Knowledgebase. See "Database Server Steps" on page 84.

Preparing to Install

1. Determine whether you are performing a full install of Release 8.0.1 or upgrading from Release 8.0.
2. Read the **Readme.txt** file supplied on the Sterling Multi-Channel Selling Solution CD-ROM for any final instructions not included in this guide.
3. Determine that your servlet container supports the Java Servlet Specification 2.3.
4. Determine the transactional and segmentation schema configurations. The system determines the configuration that you are using by examining properties that you set in the project properties file.

   The out-of-the-box Sterling Multi-Channel Selling Solution supports segmentation using Oracle and SQL Server database servers.

   See "Configuring the Transactional and Segmentation Databases" on page 31 for detailed information before proceeding.
5. Ensure that your databases and database servers are tuned appropriately for your implementation. Default settings may not always work, for example, you may have to increase settings such as the DEFAULT CURSORS setting. Consult your DBA to ensure proper database and database server tuning is complete before setting up the Sterling Multi-Channel Selling Solution application.
6. Determine the database connection information used to connect the Sterling Multi-Channel Selling Solution to the Knowledgebase database server:

   a. For an Oracle database server:

   - Determine the configuration of your transactional and segmentation databases.
   - Set up access to the transactional database and to the segmentation database from the SDK machine: set up a TNS alias for the transactional

database and, if you are using a separate server for the segmentation database, set up another TNS alias for the segmentation database.

- To run the transactional and segmentation databases on separate servers, set up the link between the two database servers. You specify the name of the link as part of specifying database connection information.

b. For a SQL Server 2005 database server using JDBC to access the database:

- Ensure that your SQL Server JDBC driver is at least version 1.1 or higher.
- To run the transactional and segmentation databases as separate databases on separate servers, set up the link between the two database servers. The transactional database must have permission to access the segmentation database, and you must specify the transactional database server's login to the segmentation database server. For this configuration, you must also set up data replication between the two SQL Servers. See "Configuring the Transactional and Segmentation Databases" on page 31 for details.
- To run the transactional and segmentation databases as separate databases on the same SQL Server, the transactional database must have permission to access the segmentation database, including login permission.

7. If you plan to add custom (or auxiliary) price types to your implementation, see "Defining Auxiliary Price Types" on page 352 for detailed information before proceeding.
8. Identify any known issues. See "Known Issues" on page 31.

9. Identify the location of the servlet container root directory, ***container_home***.

   In a typical installation on Windows 2003, the location is as follows:

   TABLE 5. **Servlet Container Homes on Windows**

Servlet Container	Home Location
WebSphere	**C:\WebSphere\AppServer\**
Tomcat	**C:\Program Files\Apache Software Foundation\Tomcat 6**
WebLogic	**C:\bea\weblogic92**

   In a typical installation on UNIX, the location is as follows:

   TABLE 6. **Servlet Container Homes on UNIX**

Servlet Container	Home Location
WebSphere	**/usr/WebSphere/AppServer/**
Tomcat	**/usr/local/tomcat/**
WebLogic	**/apps/bea/weblogic92/**

10. Identify the destination directory location, ***debs_home***, of the Sterling Multi-Channel Selling Solution. This is usually a sub-directory of ***container_home***, but its precise location can vary from one servlet container to another.

    In a typical installation on Windows 2003, the location is as follows:

Tomcat	**C:\Program Files\Apache Software Foundation\Tomcat 6\webapps\**
WebSphere	**C:\WebSphere\AppServer\hosts\default_host\**
WebLogic	**C:\bea\weblogic92\user_projects\domains\*mydomain*\applications**

11. Remove any existing deployment of the Sterling Multi-Channel Selling Solution Web application from the ***debs_home*** directory before starting the installation procedure. This requires using the servlet container's administrative console to remove the Web application, and then physically deleting the directories and files.

12. If you plan to implement Sterling Configurator, create an environment variable to specify the location of your JDK on the servlet container machine. For Windows systems, at the command line, enter:

    ```
    set JDK_HOME=<path_to_JDK>
    ```

    For example:

```
set JDK_HOME=c:\jdk1.5
```

For UNIX systems, enter:

```
setenv JDK_HOME <path_to_jdk>
```

For example:

```
setenv JDK_HOME /usr/java/jdk1.5
```

You must also set a JAVA_HOME environment variable on the machine used to run the SDK. If you use the same machine for both functions, then JAVA_HOME and JDK_HOME must have the same value.

The PATH environment variable must include the **JDK_HOME/bin/** directory.

Known Issues

There are no known issues for this release.

Configuring the Transactional and Segmentation Databases

The segmentation feature of the Sterling Multi-Channel Selling Solution involves collecting data about your users on your transactional database, and transferring that data to a segmentation database for processing and re-use in marketing activities. You configure your Sterling Multi-Channel Selling Solution implementation to provide efficient processing of user data and to enable communication between the transactional and segmentation databases. You must choose a configuration whether you are installing for the first time or upgrading from a previous release.

Segmentation is supported on Oracle and SQL Server 2005.

You may wish to re-use your database instances for particular purposes. The following table lists the acceptable and unacceptable database re-use combinations of transactional and segmentation databases. OK indicates an allowed combination. Not OK indicates a disallowed combination. Disallowed combinations avoid damage to your environment and allow supporting different deployments. Using a disallowed combination results in errors.

TABLE 7. **Acceptable and Unacceptable Database Combinations**

Previous DB/Current DB	**Transactional**	**Segment**	**Both**
Transactional	OK	Not OK	OK
Segment	Not OK	OK	Not OK
Both	OK	Not OK	OK

In addition, in a SQL Server 2005 two-database or two-server deployment, the transactional database server can be used only for one replication. Once this is deployed, the same transactional database server should not be used again for another two-database or two-server deployment. For best performance, we recommend one replication per server..

Attention: The segmentation database must be a clean database which has not previously been used as a transactional database.

See the *Sterling Multi-Channel Selling Solution Administration Guide* for more information about segmentation. See "Customer Segmentation" on page 127 for information about creating custom segments.

Database Configurations

The possible database configurations are:

- Oracle:
  - Transactional and segmentation databases within the same schema on the same server. This configuration is useful for development and testing purposes.
  - Transactional and segmentation databases in two separate schemas on the same server. This configuration is useful if you plan only limited use of the segmentation capabilities of the Sterling Multi-Channel Selling Solution.
  - Transactional and segmentation databases on two separate servers. This configuration is recommended for implementations that plan heavy use of the segmentation capabilities of the Sterling Multi-Channel Selling Solution. This configuration has the least performance impact on the transactional database.

- SQL Server 2005:
  - Transactional and segmentation tables in the same database. This configuration is useful for development and testing purposes.
  - Transactional and segmentation tables in two separate databases on the same server. This configuration requires that you set up data replication between the two databases. This configuration is useful if you plan only limited use of the segmentation capabilities of the Sterling Multi-Channel Selling Solution
  - Transactional and segmentation databases on two separate servers. This configuration requires that you set up data replication between the two databases. This configuration is recommended for implementations that plan heavy use of the segmentation capabilities of the Sterling Multi-Channel Selling Solution. This configuration has the least performance impact on the transactional database.

Determining Configuration Type

The system determines the configuration by examining the properties you define in the ***project*****_dev.properties** file during the installation process. The following sections describe how the system determines your implementation's configuration type for each of the supported database servers.

Determining the Oracle Configuration

- If the URL's and usernames for the transactional and segmentation databases are the same, then the transactional and segmentation databases share a schema on the same server.
- If the URL's for the transactional and segmentation databases are the same and the usernames are different, then the transactional and segmentation databases reside in separate schema on the same database server.
- If the URL's for the transactional and segmentation databases are different, the transactional and segmentation databases reside on entirely separate database servers.

Determining the SQL Server 2005 Configuration

- If the URL's and database names of the transactional and segmentation databases are the same, the transactional and segmentation databases share a database on the same server.

- If the URL’s for the transactional and segmentation databases are the same and the database names are different, then the transactional and segmentation databases are separate databases that share a database server.
- If the URL’s for the transactional and segmentation databases are different, then the transactional and segmentation databases reside on entirely separate database servers.

The following sections describe the ***project*_dev.properties** file database properties for Oracle and for SQL Server 2005.

Oracle and SQL Server Properties

The following are the Oracle and SQL Server 2005 database properties:

- ***DBTYPE*_URL:** the location of the transactional database.
- ***DBTYPE*_USERNAME**: the username for logging into the transactional database. This user is the owner of the database.
- ***DBTYPE*_PASSWORD:** the password for logging into the transactional database.
- ***DBTYPE*_SEGMENT_URL**: the location of the segmentation database.
- ***DBTYPE*_SEGMENT_USERNAME**: the username for logging into the segmentation database. This user is the owner of the database.
- ***DBTYPE*_SEGMENT PASSWORD:** the password for logging into the segmentation database.

SQL Server 2005 Replication Properties

Set the following replication properties in the ***project*_dev.properties** file. These properties are in addition to the properties listed in "Oracle and SQL Server Properties" on page 34.

The following properties must be set up on the transactional database server:

- MSSQLJDBC_SA_USERNAME: the SQL Server system administrator user of the transactional database server for setting up replication. This is needed for setup only: you can delete this property after setup is complete.
- MSSQLJDBC_SA_PASSWORD: the SQL Server system administrator password of the transactional database server for setting up replication. This is needed for setup only: you can delete this property after setup is complete.

- MSSQLJDBC_SA_DISTRIBUTOR_DATABASE: a unique name for the SQL Server distribution database. Ensure that this name is unique and that this database does not already exist on the transactional database server.
- MSSQLJDBC_SA_DATA_FOLDER: the full pathname of the folder to contain replication data. This folder must already exist and reside on the transactional database server machine.

  For Windows systems: The SDK considers the backslash character, "\", to indicate an Escape character. To specify a pathname such as C:\Replication, you must enter two backslashes. For example, C:\\Replication.
- MSSQLJDBC_START_DATE: the date on which to start data replication. The format is YYYYMMDD.

You set up replication as the SQL Server system administrator user.

Note that the replication properties are required for all SQL Server 2005 transactional/segmentation database configurations. However, replication actually occurs only for the separate databases/same server configuration or the separate databases/separate servers configuration.

SQL Server 2005 Migration Properties

The following properties must be set only when migrating data from an earlier release. Otherwise they can be left blank.

- MSSQLJDBC_SEGMENT_LINK: Name of the linked transactional server. This property needs to be set *only* when using a configuration of *two separate servers.*
- MSSQLJDBC_SCHEMA_NAME: Name of the schema used by the transactional database. This property needs to be set *only* when using a configuration of *two databases on a single server.*

When migrating data from an earlier release and using a configuration of two separate servers, the transactional server must be set up as a linked server on the segmentation server, see "SQL Server 2005 Setup - Migration with Two Servers" on page 40 for details.

Database Communication Setup

The following types of transactional and segmentation communication occurs:

- The transactional database pushes data to the segmentation database, which processes the data and resolves segment criteria into segment membership lists.

- The Sterling Multi-Channel Selling Solution Web application pulls data from the segmentation database for marketing activity purposes.
- The Sterling Multi-Channel Selling Solution Web application uploads third-party data to the segmentation database. The third-party data generally consists of files containing list of users and related information for segmentation and marketing purposes.

This section explains how to set up communication between the transactional and segmentation databases for configurations in which the transactional and segmentation databases reside on two separate servers (Oracle and SQL Server 2005).

If the transactional and segmentation databases reside in the same schema or two schemas within the same server (Oracle) or the transactional and segmentation tables reside in the same database or two databases within the same server (SQL Server), no communication links are required, so no manual setup is necessary.

You enable two databases to communicate with each other by creating a link between them. How you do that depends upon the database server you are using.

Oracle Setup

To enable communication between the transactional and segmentation databases on two separate servers, perform the following tasks:

1. Log on to the segmentation database as DBA and grant the following permissions to the owner of the segmentation database.

   ```
   GRANT CREATE MATERIALIZED VIEW TO SEGMENTATION_DB_OWNER
   GRANT CREATE PROCEDURE TO SEGMENTATION_DB_OWNER
   GRANT CREATE SEQUENCE TO SEGMENTATION_DB_OWNER
   GRANT CREATE SESSION TO SEGMENTATION_DB_OWNER
   GRANT CREATE TABLE TO SEGMENTATION_DB_OWNER
   GRANT CREATE TRIGGER TO SEGMENTATION_DB_OWNER
   GRANT CREATE VIEW TO SEGMENTATION_DB_OWNER
   GRANT CREATE DATABASE LINK TO SEGMENTATION_DB_OWNER
   GRANT UNLIMITED TABLESPACE TO SEGMENTATION_DB_OWNER
   GRANT "CONNECT" TO SEGMENTATION_DB_OWNER
   GRANT "RESOURCE" TO SEGMENTATION_DB_OWNER
   GRANT CREATE SYNONYM TO SEGMENTATION_DB_OWNER
   ```

2. Log on to the transactional database as DBA and grant the following permissions to the owner of the transactional database.

   ```
   GRANT CREATE MATERIALIZED VIEW TO TRANSACTIONAL_DB_OWNER
   GRANT CREATE PROCEDURE TO TRANSACTIONAL_DB_OWNER
   GRANT CREATE SEQUENCE TO TRANSACTIONAL_DB_OWNER
   GRANT CREATE SESSION TO TRANSACTIONAL_DB_OWNER
   ```

```
GRANT CREATE TABLE TO TRANSACTIONAL_DB_OWNER
GRANT CREATE TRIGGER TO TRANSACTIONAL_DB_OWNER
GRANT CREATE VIEW TO TRANSACTIONAL_DB_OWNER
GRANT UNLIMITED TABLESPACE TO TRANSACTIONAL_DB_OWNER
GRANT "CONNECT" TO TRANSACTIONAL_DB_OWNER
GRANT "RESOURCE" TO TRANSACTIONAL_DB_OWNER
```

3. To allow communication between the transactional database server and the segmentation database server:

   a. Set up a TNSNAMES.ora entry on the transactional database server for the segmentation database server, then

   b. Set up a TNSNAMES.ora entry on the segmentation database server for the transactional database server.

   For example, suppose that your transactional database server is TXServer.matrix.corp and your segmentation database server is SEGServer.matrix.corp. On TXServer, the TNSNAMES.ora entry for SEGServer looks like this:

```
SEGServer =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = segserver.matrix.corp)(PORT
= 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SID = DEBS10G)
    )
  )
```

   On SEGServer, the TNSNAMES.ora entry for TXSERVER looks like this:

```
TXServer =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = txserver.matrix.corp)(PORT
= 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SID = DEBS10G)
    )
  )
```

4. Create the link between the transactional and segmentation databases. You set up the link on the segmentation database to allow access to the transactional database.

   Log in to the segmentation database as the segmentation database owner and execute the following SQL commands:

   ```
   CREATE DATABASE LINK SEGMENTLINK CONNECT TO transactional_db_owner
   IDENTIFIED BY transactional_db_owner_password
   USING ’tnsname of transactional database from segment database’
   ```

   Note that database link names must be no more than 12 characters long.

   The TNSNAME must be enclosed in single quotes.

   For example, suppose that your transactional database owner name is txowner with password #pa$sw4d:

   ```
   CREATE DATABASE LINK SEGMENTLINK CONNECT TO txowner IDENTIFIED BY
   #pa$sw4d
   USING 'TXServer'
   ```

5. Set up the ***project*_dev.properties** file properties. Note that database link names must be no more than 12 characters long.

   ```
   # Oracle Transactional DB Properties
   ORACLE_URL=jdbc:oracle:thin:@<machine>:<port>:<sid>
   ORACLE_USERNAME=<username>
   ORACLE_PASSWORD=<password>
   ORACLE_DATABASE=<tnsalias>
   ORACLE_INDEX_TABLESPACE=<TABLESPACE_NAME>

   # Oracle Segmentation DB properties
   ORACLE_SEGMENT_URL=jdbc:oracle:thin:@<machine>:<port>:<sid>
   ORACLE_SEGMENT_USERNAME=<username>
   ORACLE_SEGMENT_PASSWORD=<password>
   ORACLE_SEGMENT_DATABASE=<tnsalias>
   ORACLE_SEGMENT_INDEX_TABLESPACE=<TABLESPACE_NAME>
   ORACLE_SEGMENT_LINK=<segmentlink>
   ```

SQL Server 2005 Setup

To enable communication between the transactional and segmentation databases on two separate servers, perform the following tasks as the database owner.

Note	If you are *migrating* from an earlier release and using a t*wo server configuration*, additional steps are required to establish communication between the two servers. See "SQL Server 2005 Setup - Migration with Two Servers" on page 40 for details

1. Set up the ***project*_dev.properties** file properties:

```
# SQL Server 2005 Transactional DB properties
MSSQLJDBC_URL=jdbc:sqlserver:<full network address to SQL Server
instance>;DatabaseName=<dbname>
MSSQLJDBC_USERNAME=<username>
MSSQLJDBC_PASSWORD=<password>
MSSQLJDBC_DATABASE=<dbname>
MSSQLJDBC_SERVERNAME=<machine name only: not the network address>

# SQL Server 2005 Segmentation DB properties file
# Tokenized files ODBCDataSources.xml/MSSQLSegCreateSchema.bat
MSSQLJDBC_SEGMENT_URL=jdbc:sqlserver:<full network address to SQL
Server instance>;DatabaseName=<dbname>
MSSQLJDBC_SEGMENT_USERNAME=<username>
MSSQLJDBC_SEGMENT_PASSWORD=<password>
MSSQLJDBC_SEGMENT_DATABASE=<dbname>
MSSQLJDBC_SEGMENT_SERVERNAME=<machine name only: not the network
address>

# SQL Server 2005 Replication properties
MSSQLJDBC_SA_USERNAME=<system_admin_name>
MSSQLJDBC_SA_PASSWORD=<system_admin_password>
MSSQLJDBC_DISTRIBUTOR_DATABASE=<distributor_dbname>
MSSQLJDBC_DISTRIBUTOR_DATA_FOLDER=<replication_folder_path>
MSSQLJDBC_START_DATE=<YYYYMMDD>

# SQL Server 2005 Migration properties
MSSQLJDBC_SEGMENT_LINK=<transactional_server_name>
MSSQLJDBC_SCHEMA_NAME=<schemaname>
```

The MSSQLJDBC_SEGMENT_LINK and MSSQLJDBC_SCHEMA_NAME properties need to be set *only* when migrating data from an earlier release. Otherwise MSSQLJDBC_SEGMENT_LINK can be left blank and the default value for MSSQLJDBC_SCHEMA_NAME can be left in place. See "SQL Server 2005 Migration Properties" on page 35 for further information.

The MSSQLJDBC_SERVERNAME property is used to specify the machine on which the SQL Server 2005 instance is running as well as to specify the name of the database server. In most cases the machine running the SQL Server 2005 instance and the database server are the same, but they can be different. In your SDK environment, you must ensure that the machine name by itself is enough to enable the SQL client application sqlcmd to access the SQL Server 2005 instance, and that this is the name of the SQL server instance.

SQL Server 2005 Setup - Migration with Two Servers

If you are migrating from an earlier release and using a configuration of two separate servers, the transactional server must be set up as a linked server on the segmentation server. Perform the following tasks as the system administrator (sa) to establish communication between the servers:

1. On the transactional database, perform the following as sa:

   ```
   USE [master]
   GO
   GRANT ALTER ANY LINKED SERVER TO [TRANSACTIONAL_DATABASE_NAME]
   GO
   ```

2. Create a link on the segment server to transactional server:

   ```
   USE [master]
   GO
   EXEC sp_addlinkedserver 'TRANSACTIONAL_SERVER_NAME',N'SQL Server'
   GO
   ```

3. On the segmentation server, establish the following login mapping:

   ```
   USE [master]
   GO
   EXEC master.dbo.sp_addlinkedsrvlogin @rmtsrvname =
   N'TRANSACTIONAL_SERVER_NAME', @locallogin = N'SEGMENT_USER', @use-
   self = N'False', @rmtuser = N'TRANSACTIONAL_USER', @rmtpassword =
   N'TRANSACTIONAL_PASSWORD'
   GO
   ```

4. Set up the ***project*_dev.properties** file properties:

   ```
   # SQL Server 2005 Transactional DB properties
   MSSQLJDBC_URL=jdbc:sqlserver:<full network address to SQL Server
   instance>;DatabaseName=<dbname>
   MSSQLJDBC_USERNAME=<username>
   MSSQLJDBC_PASSWORD=<password>
   MSSQLJDBC_DATABASE=<dbname>
   MSSQLJDBC_SERVERNAME=<machine name only: not the network address>

   # SQL Server 2005 Segmentation DB properties file
   # Tokenized files ODBCDataSources.xml/MSSQLSegCreateSchema.bat
   MSSQLJDBC_SEGMENT_URL=jdbc:sqlserver:<full network address to SQL
   ```

```
Server instance>;DatabaseName=<dbname>
MSSQLJDBC_SEGMENT_USERNAME=<username>
MSSQLJDBC_SEGMENT_PASSWORD=<password>
MSSQLJDBC_SEGMENT_DATABASE=<dbname>
MSSQLJDBC_SEGMENT_SERVERNAME=<machine name only: not the network
address>

# SQL Server 2005 Replication properties
MSSQLJDBC_SA_USERNAME=<system_admin_name>
MSSQLJDBC_SA_PASSWORD=<system_admin_password>
MSSQLJDBC_DISTRIBUTOR_DATABASE=<distributor_dbname>
MSSQLJDBC_DISTRIBUTOR_DATA_FOLDER=<replication_folder_path>
MSSQLJDBC_START_DATE=<YYYYMMDD>

# SQL Server 2005 Migration properties
MSSQLJDBC_SEGMENT_LINK=<transactional_server_name>
MSSQLJDBC_SCHEMA_NAME=<schemaname>
```

The value of the MSSQLJDBC_SEGMENT_LINK property will be the same as the value for the *TRANSACTIONAL_SERVER_NAME* established in the previous steps. MSSQLJDBC_SCHEMA_NAME is the name of the schema used by the transactional database.

The MSSQLJDBC_SERVERNAME property is used to specify the machine on which the SQL Server 2005 instance is running as well as to specify the name of the database server. In most cases the machine running the SQL Server 2005 instance and the database server are the same, but they can be different. In your SDK environment, you must ensure that the machine name by itself is enough to enable the SQL client application sqlcmd to access the SQL Server 2005 instance, and that this is the name of the SQL server instance.

Installing the Software Development Kit

You can use the Software Development Kit (SDK) to install the Sterling Multi-Channel Selling Solution and to manage customizations to your implementation. You must use JDK 1.5 and Version 3.5.1 of the SDK to install the Sterling Multi-Channel Selling Solution.

This chapter covers only those SDK functions used to install the Sterling Multi-Channel Selling Solution. The *Sterling Multi-Channel Selling Solution Developer Guide* provides a comprehensive guide to using the SDK to customize your Sterling Multi-Channel Selling Solution. The *Sterling Multi-Channel Selling Solurtion SDK Guide* provides a comprehensive guide to the SDK.

To install the SDK, identify or create a directory on your machine to use as the development directory, ***sdk_home***.

1. If the JAVA_HOME environment variable is not already set, set it to the location of your Java Development Kit. For example:

   ```
   set JAVA_HOME=<path_to_JDK>
   ```

   For example:

   ```
   set JAVA_HOME=c:\jdk1.5
   ```

   For UNIX systems, enter:

   ```
   setenv JAVA_HOME <path_to_jdk>
   ```

   For example:

   ```
   setenv JAVA_HOME /usr/java/jdk1.5
   ```

   For the Bourne shell, enter:

   ```
   export JAVA_HOME=<path_to_jdk>
   ```

   For example:

   ```
   export JAVA_HOME=/usr/java/jdk1.5
   ```

2. Version 3.5.1 of the SDK is delivered as a JAR file, **sdk-framework.jar**. Unjar the JAR file in the ***sdk_home*** directory.
3. At the command line, navigate to the ***sdk_home*** directory, and enter:

   ```
   sdk setup
   ```

4. For Windows systems: if your system directory is not **c:/winnt/system32**, edit the ***sdk_home*/my_sdk.properties** file and update the windows.system.dir property.
5. After you install Release 8.0.1, set the version number on the SDK Index HTML page by running the `generateIndexFile` target.

To continue with the installation, proceed to "Installing the Sterling Multi-Channel Selling Solution Using the SDK" on page 43.

Installing the Sterling Multi-Channel Selling Solution Using the SDK

This section describes how to use the SDK to install the Sterling Multi-Channel Selling Solution on either a Windows 2000/2003 or a UNIX operating system. Successful installation creates a **Sterling.war** file on your SDK machine.

Attention:	Please install your servlet container before attempting to install the Sterling Multi-Channel Selling Solution.

Follow the steps described in one of these two tasks:

- "To Install the Sterling Multi-Channel Selling Solution as a Full Release" on page 43
- "To Install the Sterling Multi-Channel Selling Solution by Upgrading from an Earlier Release" on page 50

Attention:	If your release number is not Release 8.0.1 (for example, it is Release Release 8.0.1.2), then substitute the string for the release (for example, “8.0.2”) wherever these instructions refer to “Release 8.0.1”.

To Install the Sterling Multi-Channel Selling Solution as a Full Release

You must perform this task as a user with local machine administration privileges.

1. Locate the release **Sterling.jar** file for this release (called something like **Sterling-def-8.0.1-RC*x*.jar**) and copy it to a temporary location on your system.
2. Locate the content JAR file, **doc-8.0.1-RC*x*.jar**, and copy it to a temporary location on your system. This file contains the HTML pages for the Javadoc and SDK index.
3. At the command line, navigate to the ***sdk_home*/** directory.
4. Edit the ***sdk_home*/my_sdk.properties** file to specify the value of the container.home and app.name properties. The SDK uses these properties to determine the values of other properties as follows:
    - deploy.home is set to container.home/apps.dir. The deploy.home property is used to specify the servlet container deployment directory.
    - The app.name is the name of the Web application. The app.name is usually the same as the Web application’s directory under the deployment

directory. In this guide, we assume that the value of this property is "Sterling". If you want to change the name of the Web application (for example, to change the name of the generated WAR file), modify the app.name entry in the **my_sdk.properties** file.

- debs.home takes the value container.home/apps.dir/app.name.

The project.name property defined in the **sdk-settings.properties** file is used to specify the name of the project directory in the SDK. In general, this is *not* the same as the app.name.

Note: If you are running the SDK on UNIX, you may have to add execute permission to the ***sdk_home*/sdk.sh** file:

```
chmod +x sdk_home/sdk.sh
```

After you run the merge target, you may have to modify the permissions on the ***sdk_home*/workspaces/*project*/OracleCreateSchema.sh** file.

5. Run the install target twice, specifying the location of the **Sterling.jar** file set in Step 1, and the location of the **doc.jar** file set in Step 2 on Page 43. For example:

   ```
   sdk install /tmp/Sterling-8.0.1-def-RC-4.jar
   sdk install /tmp/doc-8.0.1-def-RC-4.jar
   ```

   Note that you can use quotes if the paths to the files have spaces in them. These targets can take a few minutes to run.

   The product documentation is installed in the following directories:

   - ***sdk_home/*docs/debs-*release/*index.html:** entry point to the SDK documentation, API Reference, Developer Guide, Data Migration Guide, and entity relationship diagrams.
   - ***sdk_home/*docs/debs-*release/*proddocs:** product documentation in PDF format.

6. Run the newproject target, specifying a project name for this installation. For example:

   ```
   sdk newproject matrix
   ```

   This target can take a few minutes to run.

7. The newproject target creates properties files for the new project in the ***sdk_home*/projects/*project*/** directory. The default name for the properties file is ***project*_dev.properties**. Note that the values of properties (such as container.home) set in the properties file override the values set in the

local-sdk.properties and **my_sdk.properties** files. See "Project Property File Settings" on page 55 for details about the appropriate values for your system.

Edit the project properties file to set the database connection information for both the transactional database and the segmentation database. Note that the transactional and segmentation databases must be of the same type.

You can also set other properties such as the logging level. Values you set here are automatically merged into the **prefs.xml** configuration files under the ***sdk_home*/builds/*project*/** directory. See "Email Addresses" on page 56 for information about the email addresses set in the properties files.

8. Database targets:

   a. If you plan to run the Knowledgebase on Oracle, run the installOracle target, specifying the location of the Oracle JDBC JAR file, to copy the Oracle JDBC drivers JAR file to the project files. The name and location of the JAR file will vary from installation to installation. For example:

      C:\oracle\product\10.2.0\client_1\jdbc\lib\ojdbc14.jar

      or
      /opt/oracle/product/10.2.0/client_1/jdbc/lib/ojdbc14.zip

      For example:
      ```
      sdk installOracle /tmp/Oracle_jdbc.jar
      ```

      This copies the JAR file to the **WEB-INF/lib/** directory in the release directory. It also renames the file to **oraclejdbc.jar**.

   b. If you plan to run the Knowledgebase on SQL Server 2005, run the installMSSQLJDBC target, specifying the location of the SQL Server JDBC JAR file. For example:

      ```
      sdk installlMSSQLJDBC "C:\Program Files\Microsoft SQL Server
      2005 JDBC Driver\sqljdbc_1.1\enu\sqljdbc.jar"
      ```

Note	For SQL Server 2005, your JDBC driver must be version 1.1 or higher.

9. Run the env.setDBType target to set the appropriate database type:

   ```
   sdk env.setDBType Oracle
   ```

   or
   ```
   sdk env.setDBType MSSQLJDBC
   ```

10. To set the database password to be encrypted:

    a. Set the Encrypted flag to true:

```
sdk setVal DataServices.DataSource.ENTERPRISE.Encrypted true
```

b. Encrypt the database password string:

```
sdk encryptVal <password>
```

The result is an encrypted version of the password.

c. Edit the encrypted password string into the appropriate properties file as the relevant password property. For example:

```
ORACLE_PASSWORD=<encrypted_password>
```

The encrypted form of the password is entered into the schema creation scripts that are used by the createDB target, so you must run the createDB target before you encrypt the password.

11. If you plan to support locales other than the default U.S. English (en_US) locale, perform the localization installation and customization steps. See CHAPTER 11, "Localization" for instructions.

12. Run the merge target to create your first build in the **builds/** directory.

```
sdk merge
```

This target can take a few minutes to run. The sdk merge target copies the Web application files from the releases directory and merges in the files and properties currently in your project directory. If the target fails with a message relating to the JDBC driver, check that you have run the database install targets appropriately:

- If you are creating an Oracle-based project, then check that you have run the installOracle target, and ensure that the **oraclejdbc.jar** file is now in the ***sdk_home*****/releases/debs-8.0.1/overlay/WEB-INF/lib/** directory.
- If you are creating a SQL Server 2005-based project, then check that you have run the installMSSQLJDBC target, and ensure that the **mssqljdbc.jar file** is now in the ***sdk_home*****/releases/debs-8.0.1/overlay/WEB-INF/lib/** directory**.**

13. Run the distWar target to create the WAR file that you will deploy. For example:

```
sdk distWar
```

This target can take a few minutes to run. The generated WAR file is in ***sdk_home*****/dist/**. Its name is determined by the app.name and deploy.environment properties and a timestamp. You can rename the WAR file to **Sterling.war**.

14. Alternatively, you can run the dist target. This creates a JAR file that contains the WAR file along with JAR files that provide the SQL scripts and XML data files. You can install the Sterling Multi-Channel Selling Solution from this JAR file by following the instructions provided in "Installing the Reference Sterling Multi-Channel Selling Solution" on page 69. Note that you can skip the steps to set the database properties information because your **prefs.xml** file already has this information.

15. In the ***sdk_home*/dist/*time_stamp*/** directory, rename the generated **prefs_*env*.xml** file to **prefs.xml** file. This is the basic configuration file that must be copied under the home directory of the user running the servlet container.

16. Add any custom (auxiliary) price types to your implementation. The following lists the general steps; see "Defining Auxiliary Price Types" on page 352 for more detailed information.

Note:	All customizing is done in the ***sdk_home***/projects/***project_name*** directory structure.

a. Retrieve the **LightWeightLookupList** file for customizing:

```
sdk customize WEB-INF/xmldata/Minimal/LightWeightLookupList
```

This places the **LightWeightLookupList** file in the directory ***sdk_home*/projects/*project-name*/WEB-INF/xmldata/Minimal**.

b. Open the LightWeightLookupList file with a text editor and add your new price types to the file. The format is as follows:

```
<LightWeightLookup state="INSERTED">
  <LookupType state="INSERTED">PriceType</LookupType>
  <LookupCode state="INSERTED">lookup_code</LookupCode>
  <Locale state="INSERTED">en_US</Locale>
  <Description state="INSERTED">price_type_name</Description>
</LightWeightLookup>
```

Where ***lookup_code*** is the unique numeric code associated with the price type, such as 1000, 2000, or 3000, and ***price_type_name*** is the name of the price type, such as Monthly, Cancellation, or Overage.

Note that if you plan to load the Matrix reference data, the auxiliary price type lookup codes 1000, 2000, and 3000 are used as part of the reference data; choose other lookup codes for your custom auxiliary price types.

c. Create a new file in the directory ***sdk_home*/projects/*project-name*/ WEB-INF/xmldata** called **PriceTypeList** (no file extension) to contain your auxiliary price types. The contents must be plain text: do not use special characters, or characters such as smart quotes.

The format is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<PriceTypeListData>
<PriceTypeList state="INSERTED" type="BusinessObject">

<PriceType state="INSERTED">
  <PriceTypeCode state="INSERTED">price_type_code</PriceTypeCode>
  <Locale state="INSERTED">en_US</Locale>
  <PriceTypeGroupCode state="INSERTED">
    group_code</PriceTypeGroupCode>
  <PriceTypePropertyName state="INSERTED">
    PRICE: <property_name></PriceTypePropertyName>
    <UpdatedBy state="INSERTED">1</UpdatedBy>
    <CreatedBy state="INSERTED">1</CreatedBy>
  <ActiveFlag state="INSERTED">Y</ActiveFlag>
</PriceType>

</PriceTypeList>
</PriceTypeListData>
```

Where ***price_type_code*** is the unique numeric code associated with the price type and which matches the lookup code, such as 1000, 2000, or 3000; ***group_code*** is the price type group code with which this price type is associated, such as 20 for one-time prices; ***property_name*** is the name of the price type property, is uppercase, and always begins with PRICE:, such as PRICE: ACTIVATION; and the ***UpdateDate*** and ***CreateDate*** dates are timestamps.

d. Retrieve the **MinimalData.lst** file for customization:

```
sdk customize WEB-INF/scripts/MinimalData.lst
```

This places the MinimalData.lst file in the directory ***sdk_home/project-name*/WEB-INF/scripts.**

e. Open the **MinimalData.lst** file with a text editor and add the following line:

```
WEB-INF/xmldata/PriceTypeList
```

f. Merge the customized files into the project:

```
sdk merge
```

17. Run the createDB target to create the Knowledgebase schema.

a. If you are running the Knowledgebase on either Oracle or SQL Server, run:

   ```
   sdk createDB
   ```

   If you are running the Knowledgebase on SQL Server 2005 and want to use the JDBC connection to connect to it at runtime, then you must still use the ODBC scripts to create the database schema. Consequently, you must set the ODBC properties as well as the MSSQLJDBC properties before running this target.

b. Check the results of the createDB target by looking at log files in the ***sdk_home*/logs/projects/*project_name*** directory.

18. Run either the loadDB target (to load the minimal data set) or the loadMatrixDB target (to load the full Matrix reference data set) into the Knowledgebase.

    ```
    sdk loadDB
    ```

    or

    ```
    sdk loadMatrixDB
    ```

    Check the results of the loadDB or loadMatrixDB targets by looking at the ***sdk_home*/workspaces/*project_name*/debs.log** file.

    Check for an error similar to the following:

    ```
    File: WEB-INF/xmldata/Minimal/Partner:
    [CMGT_LOOKUP_CACHE_ENTRY_DOESNT_EXIST] error: "No Lookup Cache
    entry for locale en_ Lookup Category PartnerStatus Lookup Code
    10."
    com.comergent.api.dataservices.NoLookEntryExistForSourceExcep-
    tion: [CMGT_LOOKUP_CACHE_ENTRY_DOESNT_EXIST] error: " No Lookup
    Cache entry for locale en_ Lookup Category PartnerStatus Lookup
    Code(or Lookup Description) 10."
    ```

    This error is caused by an invalid locale specification in your system environment. On UNIX, check whether the LANG environment variable has a country setting. It should look like this:

    ```
    LANG=en_US.UTF-8
    ```

19. Create the segmentation database:

    ```
    sdk createSegDB
    ```

    Check the results of the createSegDB target by looking at the ***sdk_home*/logs/projects/*project_name*** directory.

20. Run either the loadSegDB target (to load the minimal data set) or the loadSegMatrixDB target (to load the full Matrix reference data set) into the segmentation database:

    ```
    sdk loadSegDB
    ```

    or

    ```
    sdk loadSegMatrixDB
    ```

    Check the results of the loadSegDB target by looking at the ***sdk_home*/logs/*project_name/*** directory.

Next, deploy the Web application into the servlet container. Follow the steps for your servlet container provided in "Deploying the Sterling Web Application" on page 58.

To Install the Sterling Multi-Channel Selling Solution by Upgrading from an Earlier Release

Part of the upgrade process is to upgrade your Knowledgebase database schema. This is an “in place” upgrade so be sure to back up your current database before starting, and make sure that you can restore the backup database if required. Read "Encrypted Data" on page 54 for information about handling fields that store encrypted data.

These upgrade instructions are written for the upgrade from Release 8.0 to Release 8.0.1.

You must perform the upgrade as a user with local machine administration privileges.

Upgrade Considerations for Segmentation

Releases starting with Release 8.0 support segmentation. When you migrate to Release 8.0.1 from Release 8.0, you must use the same segmentation settings and property values as you did for Release 8.0.

Upgrade and Migration Process

1. Locate the **Sterling.jar** file for this release (called something like **Sterling-amber-8.0.1-RC*x*.jar**) and copy it to a temporary location on your system.
2. Locate the content JAR file, **doc-8.0.1-RCx.jar**, and copy it to a temporary location on your system.
3. At the command line, navigate to your ***sdk_home*/** directory.

4. Run the install target twice, specifying the location of the **Sterling.jar** file set in Step 1, and the location of the content JAR file set in Step 2 on Page 43. For example:

   ```
   sdk install /tmp/Sterling-amber-8.0.1-RCx.jar
   sdk install /tmp/doc-8.0.1-RCx.jar
   ```

   Note that you can use quotes if the paths to the files have spaces in them. These targets can take a few minutes to run.

5. Run the switchdebs target:

   ```
   sdk switchdebs debs-8.0.1
   ```

6. Switch to the Release 8.0 project:

   ```
   sdk switchproject <project name>
   ```

7. Enter:

   ```
   sdk touchproject
   ```

8. If you already ran merge on your project using Release 8.0, then delete the generated bean source Java files from the ***sdk_home*/tmp/projects/<*Name of project*>/src/com/comergent/bean/simple/** directory.

9. If you want to perform automated merges, then you should install a three-way diff tool. You can download the diff3 tool from http://www.gnu.org or use a commercially available tool such as SiberMerge. Edit the **my_sdk.properties** file to specify the three-way diff tool as the mergetool.name property. In the corresponding ***sdk_home*/plugins/mergetools/** sub-directory, edit the **handler.xml** file to specify the executable that must be run to execute the merge. Make sure that this executable can be found on the path of the machine. See the SDK documentation for more information about setting up your three-way diff tool.

10. Enter:

    ```
    sdk project.premigrate debs-8.0.1
    ```

    The project.premigrate target examines the **.java** and **.jsp** files in the project and prepares a list of files to be upgraded, files to be deleted, and files that have conflicts. The files included in the deleted and conflicts lists will not be migrated automatically. If there are conflicts or other issues, then you must manually resolve them.

11. Examine the following generated log files in the ***sdk_home*/projects/<*project name*>/** directory:

- **conflictingfiles.log**: this file lists all the files that for some reason have been identified as needing manual attention as part of your upgrade. These files will not be merged when you run the project.migratefiles target.
- **filestobeupgraded.log**: this file lists all the files that have changed in both your project and in the upgrade from Release 8.0 to Release 8.0.1. You must manually edit these files to merge changes from your project into your customization of Release 8.0.1 or merge changes using a three-way diff tool.
- **filestobedeleted.log**: if it exists, this file lists all the files that were deleted between Release 8.0 and Release 8.0.1, but which are also customized in your project. This file is not generated if no files were deleted between Release 8.0 and Release 8.0.1. You must examine how these files are used in your project and manually resolve how they will need to be re-implemented for Release 8.0.1.
- The old project ***.properties** files are saved as ***.properties.bak**. You must edit the new ***.properties** files so that the relevant values, such as database connection information, are transferred to the new files.

12. You have two options for merging your project files with Release 8.0.1.
    - If you installed a three-way diff tool, then perform an automated merge of your project files as follows:

      ```
      sdk project.migratefiles
      ```

      This merges changes from the project and Release 8.0.1 files, resolving conflicts between merged files as it runs.
    - Manually merge the files by comparing the changes made in the project and Release 8.0.1 copies of the files.
13. Run the appropriate install target for your database server: installMSSQLJDBC, or installOracle.
14. Run the env.setDBType target for your database server:

    ```
    sdk env.setDBType <db_type>
    ```
15. Once you have performed any necessary modifications to your project files, run the merge and distWar targets to re-create the WAR file for your deployment as follows.
    a. Enter:

```
sdk merge -clean
```

If this target fails with compilation errors, you must resolve the errors manually.

b. Enter:

```
sdk distWar
```

This completes the upgrade of the Web application from Release 8.0 to Release 8.0.1. Now you must upgrade the database schema and data to Release 8.0.1.

Attention:	Before starting a database migration, back up your current database, and make sure that you can recover the database from the backup. The following steps are not reversible.

16. Enter:

    ```
    sdk migrateDB DEBS_8_0_to_DEBS_8_0_1
    ```

    Check the log file ***sdk_home*/logs/projects/*project*/mig_*db_type*.log** for errors. If the migration has been successful, then you proceed to the next step.

17. Enter:

    ```
    sdk addMigrateData DEBS_8_0_to_DEBS_8_0_1
    ```

    Check the log file ***sdk_home*/logs/projects/*project*/*db_type*_addMigrateData.log** for errors.

18. Enter:

    ```
    sdk migrateSegDB DEBS_8_0_to_DEBS_8_0_1
    ```

    Check the log file ***sdk_home*/logs/projects/*project*/mig_*db_type*.log** for errors. If the migration has been successful, then you proceed to the next step.

19. Enter:

    ```
    sdk addMigrateSegData DEBS_8_0_to_DEBS_8_0_1
    ```

    Check the log file ***sdk_home*/logs/projects/*project*/*db_type*_addMigrateData.log** for errors.

20. Manually install the **Sterling.war** file, following the servlet container-specific procedure for doing this. After you successfully install the Sterling Multi-Channel Selling Solution, continue with the implementation by following the steps in "Deploying the Sterling Web Application" on page 58.

Encrypted Data

This section describes how to manage encrypted data during the upgrade process. Some fields in your source Knowledgebase database are encrypted, and in Release 7.0.2 and higher some new fields are encrypted that were previously not encrypted. There are two scenarios to manage:

- A field is encrypted in your implementation. If the field is marked as unencrypted in Release 7.0.2 and higher, then you must ensure that your **DsDataElements.xml** configuration file continues to mark the field as encrypted. After you have run the data migration scripts as described above, you should make sure that your encryption scheme is declared in the **CryptograpyServices.xml** configuration file.
- A field is unencrypted in your implementation, but the field is now encrypted in Release 7.0.2 and higher. After you have run the data migration scripts as described above, you must decided what to do about this field:
  - We suggest that you mark the field as encrypted. If you leave the unencrypted values in place, then when data objects that reference the field are restored, a null value will be set in this field. If a new value is set in this field, then this will be encrypted when the data object is persisted. Contact Sterling Commerce for help with encrypting the current values: a standalone tool can be provided for this purpose.
  - If you choose to leave the field unencrypted, then you must ensure that your **DsDataElements.xml** configuration file continues to mark the field as unencrypted.

  The following field is encrypted in Release 7.0.1 and higher:

  - UserContact.PaymentNumber

  AuthUserContact objects contain user authentication data, including the user’s encrypted password.

If your data has been encrypted using the LegacySymmetricEncrypter and LegacyDigester classes, then note that these classes have been removed from Release 7.1 and higher. A library has been created to provide these classes to support upgrades: please contact Sterling Commerce, Inc. for further information.

Project Property File Settings

This section provides a brief description of the database connection properties. Before running the merge target, you must ensure that the values of the database connection properties are set correctly in your ***project_env*.properties** file.

Oracle

The Oracle section of the properties file looks like this:

```
ORACLE_URL=jdbc:oracle:thin:@<machine>:<port>:<sid>
ORACLE_USERNAME=<username>
ORACLE_PASSWORD=<password>
ORACLE_DATABASE=<tnsalias>
ORACLE_INDEX_TABLESPACE=<TABLESPACE_NAME>
```

The Oracle segmentation database section of the properties file looks like this:

```
ORACLE_SEGMENT_URL=jdbc:oracle:thin:@<machine>:<port>:<sid>
ORACLE_SEGMENT_USERNAME=<username>
ORACLE_SEGMENT_PASSWORD=<password>
ORACLE_SEGMENT_DATABASE=<tnsalias>
ORACLE_SEGMENT_INDEX_TABLESPACE=<TABLESPACE_NAME>
ORACLE_SEGMENT_LINK=<segmentlink>
```

The ORACLE_URL property is used to populate the ConnectionString attribute of the DataSource element set in the **prefs.xml** file. The ORACLE_DATABASE property should be set to the TNS alias of the Oracle server as seen from the SDK machine. It is used in the schema creation scripts, but is not used in the running system. The ORACLE_INDEX_TABLESPACE property specifies the location of your Oracle database server's index files.

SQL Server 2005

Access to a SQL Server 2005 database server is supported through a JDBC driver. The MSSQLJDBC section of the properties file looks like this:

```
# Tokenized files
MSSQLJDBC_URL=jdbc:sqlserver:<sqldb_name>;DatabaseName=<dbname>
MSSQLJDBC_USERNAME=<username>
MSSQLJDBC_PASSWORD=<password>
MSSQLJDBC_DATABASE=<dbname>
MSSQLJDBC_SERVERNAME=<machinename>
```

The SQL Server 2005 segmentation database section of the properties file looks like this:

```
# Tokenized files ODBCDataSources.xml/MSSQLSegCreateSchema.bat
MSSQLJDBC_SEGMENT_URL=jdbc:sqlserver:<sqldb_name>;Database-
Name=<dbname>
MSSQLJDBC_SEGMENT_USERNAME=<username>
MSSQLJDBC_SEGMENT_PASSWORD=<password>
MSSQLJDBC_SEGMENT_DATABASE=<dbname>
MSSQLJDBC_SEGMENT_SERVERNAME=<machinename>
```

The SQL Server 2005 replication section of the properties file looks like this:

```
# SQL Server 2005 Replication properties
MSSQLJDBC_SA_USERNAME=<system_admin_name>
MSSQLJDBC_SA_PASSWORD=<system_admin_password>
MSSQLJDBC_DISTRIBUTOR_DATABASE=<distributor_dbname>
MSSQLJDBC_DISTRIBUTOR_DATA_FOLDER=<replication_folder_path>
MSSQLJDBC_START_DATE=<YYYYMMDD>
```

The SQL Server 2005 migration section of the properties file looks like this:

```
# SQL Server 2005 Migration properties
MSSQLJDBC_SEGMENT_LINK=<sgementlink>
MSSQLJDBC_SCHEMA_NAME=<schemaname>
```

The MSSQLJDBC_URL property is used to populate the ConnectionString attribute of the DataSource element for the transactional database set in the **prefs.xml** file. The MSSQLJDBC_SEGMENT_URL property is used to populate the ConnectionString attribute of the DataSource element for the segmentation database set in the **prefs.xml** file. The MSSQLJDBC_USERNAME, MSSQLJDBC_PASSWORD, MSSQLJDBC_DATABASE properties are used to populate values in the **prefs.xml** file. The MSSQLJDBC_SERVERNAME property specifies an alias for the machine on which the database server runs, as specified in the MSSQLJDBC_URL property.

Email Addresses

As part of implementing the Sterling Multi-Channel Selling Solution, set up the email addresses used by the system. They reside in one of the following locations:

- Configuration Files
- Minimal Data

Configuration Files

Email addresses set in the configuration files are used by applications when sending email from the system. You set the values for these addresses in the ***.properties** files used by your SDK. When you run the SDK merge target, these values are merged into the configuration files. The following email addresses must be set:

- SMTP_SENDER: used as the From address when email is sent from the Sterling Multi-Channel Selling Solution.
- INVOICE_EMAIL_ADDRESS: the email address of an enterprise user to whom emails are sent relating to invoices. The user must have the AccountReceivable role.
- RFQ_EMAIL_ADDRESS: the email address of an enterprise user to whom emails are sent relating to RFQs: the user must have the CustomerServiceRepresentative role.
- ENTERPRISE_EMAIL_ADDRESS: set to the same value as SMTP_SENDER.
- SMTP_RECIPIENT: no longer used.

Minimal Data

When you load the minimal data, you create some partners and some users. The email addresses associated with these are currently set to changeme@changeme.com. You should change these values to more suitable values before loading the data.

The Partner Profile email addresses for the Enterprise, AnonymousUserPartner, and RegisteredUserPartner should all be set to a system administrator email account.

The email addresses for the users (admin, ERPAdmin, and AnonymousUser) should be set to the email address of a system administrator at your implementation. They can be changed through the Sterling Multi-Channel Selling Solution user interface.

ObjectMap Settings

Sterling Configurator uses rules to determine some of the behavior of models used to configure products. These rules are compiled into Java classes when the rule is first fired. The rules may be compiled using either an external compiler (that is, using javac) or an internal compiler (that is, the com.sun.tools.javac.Main class). An element of the **ObjectMap.xml** configuration file is used to specify the one you want to use. Bear in mind that there are differences between using these two compilers as described here:

- external compiler: this is spawned as a new process. On UNIX, this can cause the creation of a copy of the current process running the Sterling Multi-Channel Selling Solution and so may require a large memory allocation, and fail if this is not available.

- internal compiler: this is generally faster, but may be prone to memory leaks.

If you are running Sterling Configurator and must use the external form of the Java compiler (for example if you are running Tomcat 6 on Windows), you must specify that the Sterling Multi-Channel Selling Solution uses the external rule compiler. You do this by editing the **WEB-INF/properties/ObjectMap.xml** configuration file to change:

```
<Object ID="com.comergent.apps.configurator.util.ConfigCompiler">
<ClassName>
    com.comergent.apps.configurator.util.InMemoryRuleCompiler
</ClassName>
</Object>
```

to:

```
<Object ID="com.comergent.apps.configurator.util.ConfigCompiler">
<ClassName>
    com.comergent.apps.configurator.util.RuleCompiler
</ClassName>
</Object>
```

We recommend using the external form of the compiler for production systems: a memory leak can result if you use the internal compiler. To use the internal compiler, copy the **tools.jar** file from your JDK to the appropriate **lib/** directory in your servlet container deployment. For example, in a typical Tomcat 6 deployment, you can copy **tools.jar** to your ***tomcat_home*/shared/lib/** directory.

Deploying the Sterling Web Application

Once you successfully install the **Sterling.war** file, you must deploy it as a Web application. This process varies from one servlet container to another. Check your servlet container documentation for further details. This section provides the specific deployment steps for each of the supported servlet containers.

You must identify the operating system user that runs the servlet container. You must copy the **prefs.xml** configuration file created in Step 13 on Page 46 to a sub-directory of the home directory of this user. The sub-directory is called ***user_home*/cmgt/debs/conf/**.

The ***user_home*** home directory location can vary from one operating system to another. The following table provides some typical locations.

TABLE 8. **Operating System Home Directories**

Operating System	Standard Home Directory
Windows 2003	C:\Documents and Settings*username*
Solaris 9.0	/export/home/*username*
Linux 2.6	/home/*username*

Note that you can put this file in an alternate location which is specified as the comergent.preferences.store system property. If you do so, then you must specify its location so that it can be read when the servlet container starts the Sterling Multi-Channel Selling Solution Web application. You can do this in the following ways:

- Set its location as a system variable. For example, add the following to the command that starts the servlet container:

  ```
  -Dcomergent.preferences.store=/home/scowner/tomcat6014/prefs.xml
  ```

- Set its location in the **WEB-INF/web.xml** using the following element:

  ```
  <init-param>
      <param-name>comergent.preferences.store</param-name>
      <!--BEGIN:com.comergent.tools.ant.taskdefs.SetFileContents
          (do not modify this tag) -->
      <param-value>/home/scowner/tomcat6014/prefs.xml</param-value>
      <!--END:com.comergent.tools.ant.taskdefs.SetFileContents
          (do not modify this tag) -->
      <description>Location of Comergent's preferences store
      </description>
  </init-param>
  ```

We provide deployment steps for the following servlet containers:

- "To Deploy the Sterling Web Application on Apache Tomcat" on page 60
- "To Deploy the Sterling Web Application on IBM WebSphere" on page 62
- "To Deploy the Sterling Web Application on WebLogic 9.2" on page 65

XML Parser Settings

To ensure that the correct Java classes are used for the XML processing performed by the Sterling Multi-Channel Selling Solution, you must ensure that the Java Virtual Machine settings specify the correct classes. In general, you can either set the classes as additional parameters in the command line that starts the servlet container or you can specify them as parameters for the Web application.

The following sections describe the steps necessary to set the command line parameters for each of the supported servlet containers. To set the parameters for the Web application, add the following to the **web.xml** file located in the ***debs_home*/Sterling/WEB-INF/** directory:

```
<context-param>
    <param-name>Comergent.xml.SAXParserFactory</param-name>
    <param-value>
        org.apache.xerces.jaxp.SAXParserFactoryImpl
    </param-value>
    <description>SAX Parser factory configuration</description>
</context-param>
<context-param>
    <param-name>Comergent.xml.DocumentBuilderFactory</param-name>
    <param-value>
        org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
    </param-value>
    <description>DOM Parser factory configuration</description>
</context-param>
```

Note that these settings are overridden by values set at the command line.

Tomcat Releases

To Deploy the Sterling Web Application on Apache Tomcat

If you have installed the **Sterling.war** file into the default Web applications directory, ***container_home*/webapps/**, then Tomcat can automatically detect it, and it is deployed automatically when you start Tomcat. Make sure that there is no pre-existing **Sterling/** directory already in ***container_home*/webapps/**.

1. Ensure that your JDK **bin** directory is defined in the system PATH environment variable. For example:

   ```
   C:\Program Files\Java\jdk1.5.0_08\bin
   ```

2. On Windows installations of Tomcat, you must use the client version of the Java VM DLL. Open **Start -> All programs -> Apache Tomcat 6.0 -> Configure Tomcat**, and on the Java tab, set the Java Virtual Machine to the location of the client JVM DLL (for example: C:\Program Files\Java\jdk1.5.0_07\jre\bin\client\jvm.dll).

3. Modify the Tomcat startup parameters to set the following Java parameters:

   - -Xms128m

- -Xmx*<75% of physical memory>*
- -XX:MaxPermSize=128M

Set the Java parameter -Xmx to 75% of physical memory. For example, on a machine with 1 gigabyte of RAM, set -Xmx768m. On a machine with 2 gigabytes of RAM, set -Xmx1793m.

Do not use the -Xss option, which sets the Java thread stack size.

a. If you are running Tomcat on a UNIX or Linux system, then set the Java parameters as follows:

```
set JAVA_OPTS=-Xms128m -Xmx<75% of physical memory>
-XX:MaxPermSize=128M
```

Note that, while you can also use the **set** command to set the Java parameters on Windows, it is best to define them in the JAVA_OPTS or CATALINA_OPTS environment variables.

b. If you are running Tomcat as a service, set the Java parameters as part of the Tomcat service configuration. Open **Start -> All programs -> Apache Tomcat 6.0 -> Configure Tomcat**, then click the **Java** tab. The fields map as follows:

   - Initial memory pool: -Xms
   - Maximum memory pool: -Xmx

4. On Windows systems with Tomcat installed as a service, you must set the login information through the service. To ensure that the Sterling Multi-Channel Selling Solution works correctly and locates the correct **prefs.xml** file, set the login to the user that owns **prefs.xml**.

   For example, if the **prefs.xml** file is located in:
   `C:\Documents and Settings\adminuser.DOMAIN\cmgt\debs\conf`

   Then the Tomcat login information should be similar to the following:

   - Login: DOMAIN\adminuser
   - Password: adminuser’s password

Notes on Using Apache Tomcat

Note that Apache Tomcat does not automatically re-compile JSP pages that are included in other JSP pages: if you make a change to an included JSP page, then remove the corresponding compiled servlet classes from the ***container_home*/ work/** directory to force the JSP page to be re-compiled.

If you use the shutdown command to stop Tomcat gracefully, then persistent session information is saved to a file:

- ***container_home*/work/Standalone/localhost/Sterling/SESSIONS.ser** on Tomcat 6

When you restart the servlet container, the servlet container will attempt to reload this session data and throw exceptions. You should remove this file before re-starting the servlet container.

You can force Tomcat to not save session information by setting the saveOnRestart attribute of the Context element to “false”. To do this within the SDK, modify the **tomcat-context.xml** file to the following:

```
<Context path="/@app.name@" docBase="@project.base@" >
<Manager className="org.apache.catalina.session.PersistentManager"
    debug="0"
    saveOnRestart="false"
    maxActiveSessions="-1"
    minIdleSwap="-1"
    maxIdleSwap="-1"
    maxIdleBackup="-1">
    <Store className="org.apache.catalina.session.FileStore"/>
    </Manager>
</Context>
```

If you run the fastdeploy target, then this XML file is used to declare the Web application in the ***container_home*/webapps/** directory.

Certain class files in JAR files are not loaded from ***container_home*/webapps/Sterling/WEB-INF/lib/**. If you place the JAR files in ***container_home*/lib/** or ***container_home*/common/lib/**, then they will be loaded, but note that this may affect the running of other Web applications in the same servlet container.

Continue with the steps described in "Database Server Steps" on page 84.

WebSphere Releases

To Deploy the Sterling Web Application on IBM WebSphere

Follow these steps to deploy the Sterling Multi-Channel Selling Solution into your installation of IBM WebSphere Application Server. These steps are written to deploy into WebSphere 6.1. In this section, we refer to ***WAS_HOME***: it is the IBM terminology for the ***container_home*** directory. On a UNIX system, its default location is **/usr/WebSphere/AppServer/**. In the WebSphere Administrator’s

Console, the node on which the servlet container is installed is displayed as the machine name: we refer to this as <*server*>.

Notes:	Start and stop the WebSphere application servers using the administration console (Integrated Solutions Console) or choose Start and Stop from the Windows Start menu's IBM WebSphere -> Application Server -> Profiles -> <server> menu. Stopping the server manually can corrupt the configuration data. Do not use the "_" character in WebSphere server names: WebSphere regards URLs with this character as invalid.

1. Log in to administration console at:

   ```
   https://<server>:9043/ibm/console/logon.jsp
   ```

   using the WebSphere administrator account. You can also start the administration console from the Windows Start menu.

2. Browse to **Security->Secure administration, applications, and infrastructure**, and uncheck the following check boxes:

   - **Enable application security** to disable WebSphere security
   - **Java 2 Security** to disable Java policy security

3. Click **Apply**.

4. Browse to **Applications -> Install New Application**.

5. In the **Preparing for the application installation** panel, check the **Local file system** radio button, and click **Browse** to browse to the location of the WAR file.

6. Enter the context name (for example, "Sterling") for your Web application in the **Context root** text field, then click **Next**.

7. In the **Install New Application** panel, **Step 1 Select installation options**, set the value of Application Name to your preferred one, for example "Sterling", then click **Next**.

8. In **Step 2: Map modules to servers,** Check **Sterling Product Suite** check box, then click **Next**.

9. In **Step 3: Map virtual hosts for Web modules**:

   a. Check the **Sterling Product Suite** check box.

   b. Select default_host in the Virtual Host drop-down list.

   Click **Next**.

10. In **Step 4: Summary**, click **Finish**.
11. When WebSphere completes installing the Sterling Multi-Channel Selling Solution Web application, click the **Save directly to the master configuration** link.

    The Install New Application page displays when deployment completes.
12. Click the **Enterprise Applications** link. On the Enterprise Applications page, select the Sterling application, then click **Start**.
13. Verify that the Sterling Multi-Channel Selling Solution is successfully deployed by pointing your browser to:

    ```
    http://<server>:9080/Sterling/en/US/enterpriseMgr/matrix
    ```

Solaris Optional Step

On Solaris installations of WebSphere, the following may help with JSP page compilation problems.

14. Add a JVM setting as follows:
    a. Using the Integrated Solutions Console, navigate to:

       Servers -> Application Servers -> <*server*> ->Java and Process Management ->Process Definition -> Java Virtual Machine.

    b. In the Maximum Heap Size, enter 128.

    c. Click **Apply,** then click **Save**.
15. Restart the WebSphere Application Server.

Continue with the steps described in "Database Server Steps" on page 84.

Pre-Compiling JSP Pages

As an optional step, you can pre-compile the JSP pages in the Sterling Multi-Channel Selling Solution as follows:

1. At the command line, navigate to ***WAS_HOME*/bin/**.
2. Enter:

   ```
   ./JspBatchCompiler.sh -enterpriseApp Sterling -webModule "Sterling
   Product Suite" -verbose true [-nameServerPort <port>]
   ```

   You only need to specify the nameServerPort if your servlet container is listening on a non-standard port.

WebLogic Releases

Deployment Considerations For WebLogic 10

WebLogic 10 requires additional JVM settings to allow credit card authorization to CyberSource. To ensure that credit card authorization to CyberSource works properly, start the Weblogic 10 instance with the following JVM setting:

```
-Dweblogic.security.SSL.allowSmallRSAExponent=true
```

If you run WebLogic 10 with proxy enabled, disable the SSL hostname verification as follows:

```
-Dweblogic.security.SSL.ignoreHostnameVerification=true
```

At the time of publishing, more information is available on the BEA site at the following URL:

```
http://e-docs.bea.com/wls/docs90/ConsoleHelp/taskhelp/security/
DisableHostNameVerification.html
```

To Deploy the Sterling Web Application on WebLogic 9.2

Deployment of the Sterling Multi-Channel Selling Solution into WebLogic Release 9.2 has been simplified from earlier releases of WebLogic. However, because the Sterling Multi-Channel Selling Solution must run as an “expanded” Web application (as opposed to as a WAR file), these instructions ensure that the Web application WAR file is expanded as part of the deployment process.

1. In your WebLogic installation, identify the ***domain_home*** directory that you plan to use for your deployment of the Sterling Multi-Channel Selling Solution. The default location is ***container_home*/user_projects/domains/*mydomain*/**.
2. In the ***container_home*/user_projects/domains/*mydomain*/** directory, create a directory in which to deploy your applications called **applications**. In the **applications** directory, create the directory in which to deploy your Sterling Multi-Channel Selling Solution application, **Sterling**.
3. Expand the Sterling.war file into the ***container_home*/user_projects/domains/*mydomain*/applications/Sterling** directory using a tool such as WinZip. This process can take a few minutes. Verify that the Sterling directory structure is in place as you expand the Sterling.war file.
4. Log in to the WebLogic server as the user used to install WebLogic and start the BEA WebLogic Administration Console.
5. Install the Sterling Multi-Channel Selling Solution as a WebLogic application:

   a. In the Domain Configurations section, click Deployments, then click the Lock & Edit button in the left-hand panel.

   b. In Deployments, click the Install button. Using the Install Applications Assistant, locate the ***container_home*/user_projects/domains/*mydomain*/applications** directory, select Sterling, and click Next.

   c. Use the Install Applications Assistant to configure your Sterling application, or accept the defaults and click Finish.

   d. In the left-hand panel, click the Activate Changes button. The Sterling application appears in the Deployments list with a State of Prepared.

6. Click Start to start the Sterling application and choose Servicing All Requests. The application's State will update to Start/Running.

7. Your Sterling application is now installed. Click the Release Configuration button.

8. Verify that you can log into the Sterling Multi-Channel Selling Solution by pointing your browser to the standard URL:

   ```
   http://<server>:<port>/Sterling/en/US/enterpriseMgr/matrix
   ```

   For WebLogic servers, the default port number is 7001.

Pre-Compiling JSP Pages

As an optional step, we suggest that you pre-compile the JSP pages before going live to improve performance. You can follow the instructions provided by the *WebLogic JSP Reference* (consult the document currently at this URL: http://edocs.bea.com/wls/docs81/jsp/reference.html) to pre-compile JSP pages, or contact your Sterling Commerce representative for further information.

WebLogic cleans up the directories of compiled JSP pages when the server is stopped and restarted. It is possible to use the **weblogic.xml** file to ensure that compiled JSP pages are preserved by specifying that the keepgenerated parameter is set to true, and specifying a working directory as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE weblogic-web-app
    PUBLIC "-//BEA Systems, Inc.//DTD Web Application 7.0//EN"
    "http://www.bea.com/servers/wls700/dtd/weblogic700-web-jar.dtd" >
<weblogic-web-app>
    <jsp-descriptor>
        <jsp-param>
            <param-name>
                keepgenerated
            </param-name>
```

```
            <param-value>
                true
            </param-value>
        </jsp-param>
        <jsp-param>
            <param-name>
                workingDir
            </param-name>
            <param-value>
                Comergent_jsp
            </param-value>
        </jsp-param>
    </jsp-descriptor>
</weblogic-web-app>
```

XML Parsing

If an error message displays, review CHAPTER 8, "Troubleshooting and Backing Up the Sterling Multi-Channel Selling Solution". If you suspect problems with the XML parser settings, then you can set up an XML Registry for the WebLogic server as follows. Make the following changes to the ***container_home*****/config/mydomain/config.xml** configuration file.

1. Add the following:

```
<XMLRegistry Name="Comergent XML Registry"
    DocumentBuilderFactory=
        "org.apache.xerces.jaxp.DocumentBuilderFactoryImpl"
    SAXParserFactory=
        "org.apache.xerces.jaxp.SAXParserFactoryImpl"
    TransformerFactory=
        "org.apache.xalan.processor.TransformerFactoryImpl" />
```

2. Add the following attribute to the Server element: XMLRegistry="Comergent XML Registry". For example:

```
<Server ListenPort="7001" Name="server" NativeIOEnabled="true"
    StdoutDebugEnabled="true" StdoutSeverityLevel="64"
    TransactionLogFilePrefix="config/ICC/logs/"
    XMLRegistry="Comergent XML Registry">
```

Continue with the steps described in "Database Server Steps" on page 84.

Solaris and Oracle OCI Driver

Note that if you are using an Oracle database server for the Knowledgebase, then you can use the Oracle OCI JDBC driver to connect from the Sterling Multi-Channel Selling Solution to the Oracle database server. See "Support for Oracle Server" on page 84 for further information.

Further Deployment Steps

Once you have deployed the application, you should review the following topics:

- "Database Server Steps" on page 84
- "Setting the Product Catalog Root" on page 90
- "Setting the Session Timeout" on page 90
- "Modifying the URL for the Web application DTD" on page 91
- "Managing Memory" on page 92
- "High Availability and Clustering" on page 93

Matrix Reference Segments Setup

The Sterling Multi-Channel Selling Solution provides out-of-the-box segments that use the Matrix reference data. If you load the Matrix reference data, you can use these segments as examples for creating custom segments for your implementation. These segments are associated with activities such as promotions and pricing rules. You must run the segment calculations and publish the results for these segments to associate them with users (members).

To enable the Matrix reference segments, you can start a cron job to perform all segment calculations and publish all results.

1. Log in to the Sterling Multi-Channel Selling Solution as an admin user with access to the job scheduler.
2. Click **Job Scheduler** in the System Administration panel on the Sterling Multi-Channel Selling Solution home page.
3. Click the Nightly Segments Build cron job.

   The details of the cron job display.
4. Start the Nightly Segments Build cron job.

The Nightly Segments Build cron job processes and transfers information such as tracked events, performs the segmentation calculations, and publishes the results.

See the *Sterling Multi-Channel Selling Solution Administration Guide*'s Chapter 41, Job Scheduling Tasks, for more information.

Installing the Reference Sterling Multi-Channel Selling Solution

This section describes the steps to install Sterling Multi-Channel Selling Solution Release 8.0.1 without using the SDK. This provides you with a relatively quick verification of the basic deployment of our reference Web application. Before you start the process of customizing the Sterling Multi-Channel Selling Solution, install the Sterling Multi-Channel Selling Solution into the SDK and work in that environment to create your customized deployment.

Note that to provide support for locales other than U.S. English, you must install a language localization pack using the SDK.

These instructions use three logically distinct machines: the developer machine, the servlet container machine, and the database server machine. Depending on your situation, these machines may actually all be the same physical machine or different ones: we identify what steps are performed on which logical machine.

Make sure that you have the Java Development Kit (JDK) 1.5 installed on all three machines. The servlet container machine and the database server machine should have the database client tools installed to connect to the database server. You will need to know the database connection information required to connect from the servlet container machine to the database server. In addition, you will need to know how to configure your database setup to take advantage of the segmentation feature. See "Configuring the Transactional and Segmentation Databases" on page 31 for more information before proceeding with the reference installation.

1. Identify the location on your development machine in which you will unpack the installation files: we refer to this as ***cmgt_home***.
2. Copy the release JAR file into ***cmgt_home***.
3. Unjar the release JAR file by navigating to ***cmgt_home*** at the command line, and executing:

   ```
   jar -xvf Sterling-8.0.1.jar
   ```

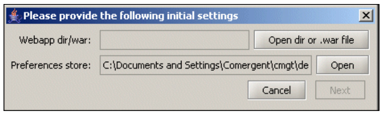
   Note that the name of the JAR file may be slightly different from the name given here.

   This unpacks the release JAR file and creates several sub-directories under ***cmgt_home***.
4. You must now set the values of system properties that the Web application will need, such as the location of the database. At the command line, execute:

```
java -jar install/cmgt-preferences-tools.jar
```

If you are running on Linux and see an error message, then you may have to install the **xorg-x11-deprecated-libs.rpm** RPM package appropriate for your Linux version.

5. The initial settings dialog box displays.

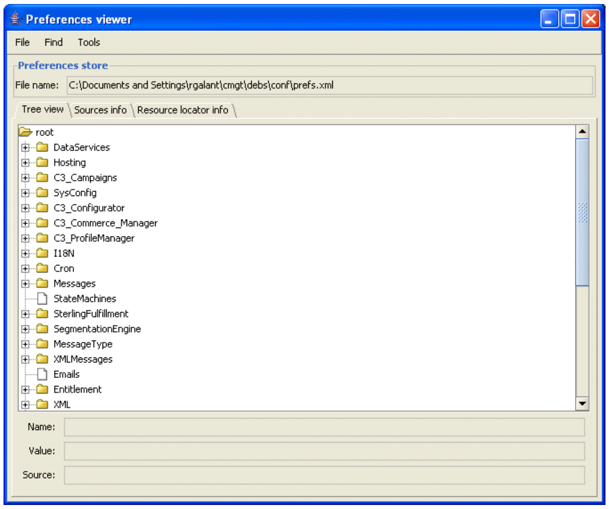


Webapp dir/war:
Preferences store: C:\Documents and Settings\Comergent\cmgt\de

FIGURE 2. **Initial Settings Dialog Box**

6. Click the Open dir or .war file button to navigate to the release WAR file, then select the file named Sterling-8.0.1-def-RC-1.war (or something very similar to this). The release WAR file name appears in the File Name field. Click Open. The Preferences Store Open File dialog box should open up in the correct directory and you should be able to leave the value of the Preferences store with its current value.

7. Click **Next**. The main Preferences Viewer window displays.



File name: C:\Documents and Settings\rgalant\cmgt\debs\conf\prefs.xml

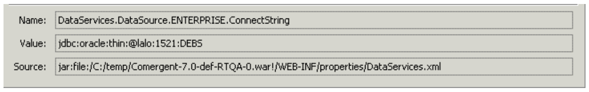
root
DataServices
Hosting
C3_Campaigns
SysConfig
C3_Configurator
C3_Commerce_Manager
C3_ProfileManager
I18N
Cron
Messages
StateMachines
SterlingFulfillment
SegmentationEngine
MessageType
XMLMessages
Emails
Entitlement
XML

Name:
Value:
Source:

FIGURE 3. **Preferences Viewer Window**

8. Using this window, set the values for the following properties as follows:

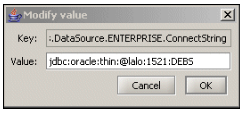
   a. Navigate to the DataServices.DataSource.ENTERPRISE.ConnectString property. You should see the lower property panel display the name of the property, its current value, and where the value is stored:



Name: DataServices.DataSource.ENTERPRISE.ConnectString
Value: jdbc:oracle:thin:@lalo:1521:DEBS
Source: jar:file:/C:/temp/Comergent-7.0-def-RTQA-0.war!/WEB-INF/properties/DataServices.xml

FIGURE 4. Property Panel

   b. In the Tree View, right-click the Connect String property and select Modify Value.



Modify value
Key: :.DataSource.ENTERPRISE.ConnectString
Value: jdbc:oracle:thin:@lalo:1521:DEBS

FIGURE 5. Modify Value Dialog Box

   c. In the Modify Value dialog box, enter the correct value for the connection string property: this is the URL used by the data services layer to connect to the Knowledgebase database server. The form of this URL depends on the database type as follows:

      - For Oracle: “jdbc:oracle:thin:@<*machine*>:1521:<*sid*>”
      - For SQL Server: “jdbc:sqlserver://<*sqldb_name*>; DatabaseName=<*dbname*>”

   d. Repeat these steps for the following properties:

   - DataServices.DataSource.ENTERPRISE.DataService:
      - For Oracle: “JdbcService”
      - For SQL Server: “JdbcService”
   - DataServices.DataSource.ENTERPRISE.SrvcSubType:
      - For Oracle: “ORACLE”
      - For SQL Server: “MS”
   - DataServices.DataSource.ENTERPRISE.UserId

- DataServices.DataSource.ENTERPRISE.Password
- DataServices.DataSource.SEGMENT.ConnectString:
    - For Oracle: “jdbc:oracle:thin:@<*machine*>:1521:<*sid*>”
    - For SQL Server: “jdbc:sqlserver://<*sqldb_name*>; DatabaseName=<*dbname*>”
- DataServices.DataSource.SEGMENT.Dataservice:
    - For Oracle: “JdbcService”
    - For SQL Server: “JdbcService”
- DataServices.DataSource.SEGMENT.SrvcSubType:
    - For Oracle: “ORACLE”
    - For SQL Server: “MS”
- DataServices.DataSource.SEGMENT.UserId
- DataServices.DataSource.SEGMENT.Password
- DataServices.General.JdbcDriver1:
    - For Oracle: “oracle.jdbc.driver.OracleDriver”
    - For SQL Server: “com.microsoft.sqlserver.jdbc.SQLServerDriver”
- DataServices.General.DsKeyGenerators: set to the database-specific value:
    - For Oracle: “OracleKeyGenerators.xml”
    - For SQL Server: “MSSQLJDBCKeyGenerators.xml”
- You can set any other properties that you wish using the Preferences Editor, but these should be enough to get your reference deployment up and running.

9. When you have finished setting property values, click **File -> Save**.
10. Click **File -> Exit**.

    The new property values will be saved to the file: **cmgt/debs/conf/prefs.xml** in your home directory, referred to as ***user_home*** (for example: **C:\Documents and Settings\*username*\** or **/export/home/*username*/**).
11. If the database server is inaccessible from your current network location, then transfer the appropriate sql-data file (depending on your DB_TYPE) to a

temporary location, ***cmgt_data_home***, on a machine that can access the database server machine:

- ***cmgt_home*****/sql/MSSQLJDBCsql-data-8.0.1-def-RC-1.jar**

  or

- ***cmgt_home*****/sql/Oraclesql-data-8.0.1-def-RC-1.jar**

12. On this machine, at the command line navigate to ***cmgt_data_home***.
13. Unpack the JAR file by executing:

```
jar -xvf DB_TYPEsql-data-8.0.1-def-RC-1.jar
```

14. Set up your database users, privileges, and, if you are using an Oracle database server, the Oracle database link. See "Configuring the Transactional and Segmentation Databases" on page 31 for details.
15. If you are using an Oracle database server, edit the following files:

- ***cmgt_data_home*****/WEB-INF/sql/Oracle/setup/oracle_indexes.sql** file: change the value of @ORACLE_INDEX_TABLESPACE@ name, if necessary, to TABLESPACE <*tablespace_name*>. The TABLESPACE name specifies the location of your Oracle database server's index files.
- ***cmgt_data_home/WEB-INF/sql/Oracle/setup/Oracle_privileges.sql*** file: replace @ORACLE_USERNAME@ with the name of the transactional database user name and @ORACLE_SEGMENT_USERNAME@ with the segmentation database user name. If your configuration is transactional and segmentation databases within the same schema on the same server, you can leave @ORACLE_SEGMENT_USERNAME@ blank.
- ***cmgt_data_home/WEB-INF/sql/Oracle/segment/oracle_mviews1.sql*** file: replace @ORACLE_USERNAME@ with the transactional database user name and @ORACLE_SEGMENT_USERNAME@ with the segmentation database user name. If your configuration is transactional and segmentation databases within the same schema on the same server, you can leave @ORACLE_SEGMENT_USERNAME@ blank.
- ***cmgt_data_home/WEB-INF/sql/Oracle/segment/oracle_mviews.sql*** file: If your configuration is transactional and segmentation schema within the same server (either in the same schema or separate schema), you can skip this step. Otherwise, replace @ORACLE_SEGMENT_LINK@ with the name of the Oracle database link that was created on the segmentation database to access the transactional database.

16. If you are using a SQL Server 2005 database server, edit the following files:

- ***cmgt_data_home/WEB-INF\sql\MSSql\setup/ mssql_create_replication.sql*** file: replace the following variables with appropriate values. See "SQL Server 2005 Setup" on page 39 for details.
    - @MSSQLJDBC_SERVERNAME@: This is the actual name of the server machine, not the network name
    - @MSSQLJDBC_SA_PASSWORD@
    - @MSSQLJDBC_DISTRIBUTOR_DATABASE@
    - @MSSQLJDBC_DISTRIBUTOR_DATA_FOLDER@
    - @MSSQLJDBC_DATABASE@
    - @MSSQLJDBC_SEGMENT_SERVERNAME@: This is the actual name of the server machine, not the network name.
    - @MSSQLJDBC_SEGMENT_DATABASE@
    - @MSSQLJDBC_SEGMENT_USERNAME@
    - @MSSQLJDBC_SEGMENT_PASSWORD@
    - @MSSQLJDBC_START_DATE@
- ***cmgt_data_home/WEB-INF\sql\MSSql\setup/ mssql_drop_replication.sql*** file: replace the following variables with appropriate values. See "SQL Server 2005 Setup" on page 39 for details.
    - @MSSQLJDBC_SA_PASSWORD@
    - @MSSQLJDBC_DISTRIBUTOR_DATABASE@
    - @MSSQLJDBC_DISTRIBUTOR_DATA_FOLDER@
    - @MSSQLJDBC_DATABASE@
    - @MSSQLJDBC_SEGMENT_SERVERNAME@: This is the actual name of the server machine, not the network name.
    - @MSSQLJDBC_SEGMENT_DATABASE@
    - @MSSQLJDBC_SEGMENT_USERNAME@
    - @MSSQLJDBC_SEGMENT_PASSWORD@

17. Edit the following batch scripts to add your connection information. The scripts reside in ***cmgt_data_home***.

- For Oracle database servers:
    - **OracleCreateSchema.bat** or **OracleCreateSchema.sh**
    - **OracleCreateSegmentSchema.bat** or **OracleCreateSegmentSchema.sh**
    - If the transactional and segmentation databases are on separate Oracle servers (the database URL's have different values), edit **OracleCreateSegmentSchema1.bat** or **OracleCreateSegmentSchema1.sh**
    - If the transaction and segmentation databases are on the same Oracle server (the database URL's have the same values) and their user names are different, edit **OracleCreateSegmentSchema2.bat** or **OracleCreateSegmentSchema2.sh**
    - **OracleCreateSegmentSchema3.bat** or **OracleCreateSegmentSchema3.sh**
- For SQL Server 2005 database servers:
    - **MSSQLJDBCCreateSchema.bat, MSSQLJDBCCreateSegmentSchema.bat**
    - If the transactional and segmentation databases are on separate SQL Servers or different databases on the same server (the database URL's have different values), edit **MSSQLJDBCCreateSegmentSchema1.bat**.
    - **MSSQLJDBCCreateSegmentSchema3.bat**

18. Run the following scripts in the order specified:
    - Oracle:

    a. **OracleCreateSchema.bat** or **OracleCreateSchema.sh**

    b. **OracleCreateSegmentSchema.bat** or **OracleCreateSegmentSchema.sh**

       You must use the TNS alias name of the Oracle database server as seen from the current machine.

    - SQL Server 2005:

    a. **MSSQLJDBCCreateSchema.bat**

    b. **MSSQLJDBCCreateSegmentSchema.bat**

       You must use the ODBC source name of the SQL Server database as seen from the current machine.

19. Run the appropriate scripts for your transactional/segmentation database configuration:
    - For Oracle:
        - If the transactional and segmentation databases are on separate Oracle servers (the database URL’s have different values), run **OracleCreateSegmentSchema1.bat** or **OracleCreateSegmentSchema1.sh**
        - If the transaction and segmentation databases are on the same Oracle server (the database URL’s have the same values) and their user names are different, run **OracleCreateSegmentSchema2.bat** or **OracleCreateSegmentSchema2.sh**
        - Always run **OracleCreateSegmentSchema3.bat** or **OracleCreateSegmentSchema3.sh**
    - For SQL Server 2005:
        - If the transactional and segmentation databases are on separate SQL Servers (the database URL’s have different values), or the transactional and segmentation databases are on the same SQL Server but in two different databases, run **MSSQLJDBCCreateSegmentSchema1.bat**.
        - Always run **MSSQLJDBCCreateSegmentSchema3.bat** or **MSSQLJDBCCreateSegmentSchema3.sh.**
20. Copy the following files to a temporary location, ***cmgt_app_home***, on the servlet container machine:
    - ***cmgt_home*/Sterling-8.0.1-def-RC-1.war**
    - ***user_home*/cmgt/debs/conf/prefs.xml**
    - ***cmgt_home*/data/Sterling-xmldata-8.0.1-def-RC-1.jar**
    - ***cmgt_home*/data/cmgt-xmlloader-tool.jar**
    - ***cmgt_home*/install/cmgt-cryptography-tool.jar**
    - ***cmgt_home*/install/cmgt-jspResourcer.jar**
    - ***cmgt_home*/install/xmlClient-tool.jar**
21. On the servlet container machine, copy the ***cmgt_app_home***/**prefs.xml** file to the following directory (which you may have to create): ***user_home*/cmgt/**

debs/conf/prefs.xml. Note that this should be the ***user_home*** for the user that is used to run the servlet container.

22. If you are running against Oracle or SQL Server 2005, then install the appropriate JDBC JAR file into the servlet container so that it can be used by the Sterling Multi-Channel Selling Solution web application when deployed. In the Tomcat Application Server, install these JAR files by placing them in the ***container_home*/common/endorsed/** directory. In WebLogic application servers, this step is probably unnecessary, since WebLogic servers are deployed with an extensive collection of JDBC clients.
23. Make sure that you are logged in as the user who is running the application server, and at the command line, navigate to ***cmgt_app_home***.
24. Unpack the **Sterling-xmldata-8.0.1-def-RC-1.jar** file by executing:

    ```
    jar -xvf Sterling-xmldata-8.0.1-def-RC-1.jar
    ```

25. Add any custom (auxiliary) price types to your implementation. The following lists the general steps; see "Defining Auxiliary Price Types" on page 352 for more detailed information.

    a. Open the **Web-INF/xmldata/Minimal/LightWeightLookupList** file in a text editor and add your new price types to the file. The format is as follows:

       Where ***lookup_code*** is the unique numeric code associated with the price type, such as 1000, 2000, or 3000, and ***price_type_name*** is the name of the price type, such as Monthly, Cancellation, or Overage.

    b. Create a new file in the directory **WEB-INF/xmldata** called **PriceTypeList** to contain your auxiliary price types. The format is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>

<PriceTypeListData>
<PriceTypeList state="INSERTED" type="BusinessObject">

<PriceType state="INSERTED">
  <PriceTypeCode state="INSERTED">price_type_code</PriceTypeCode>
  <Locale state="INSERTED">en_US</Locale>
  <PriceTypeGroupCode state="INSERTED">
    group_code</PriceTypeGroupCode>
  <PriceTypePropertyName state="INSERTED">
    PRICE: property_name</PriceTypePropertyName>
  <UpdateDate state="INSERTED">YYYY-MM-DD HH:MM:SS.MS</UpdateDate>
  <UpdatedBy state="INSERTED">1</UpdatedBy>
```

```
  <CreateDate state="INSERTED">YYYY-MM-DD HH:MM:SS.MS</CreateDate>
  <CreatedBy state="INSERTED">1</CreatedBy>
  <ActiveFlag state="INSERTED">Y</ActiveFlag>
</PriceType>

</PriceTypeList>
</PriceTypeListData>
```

Where ***price_type_code*** is the unique numeric code associated with the price type and which matches the lookup code, such as 1000, 2000, or 3000; ***group_code*** is the price type group code with which this price type is associated, such as 20 for one-time prices; ***property_name*** is the name of the price type property, is uppercase, and always begins with PRICE:, such as PRICE: ACTIVATION; and the ***UpdateDate*** and ***CreateDate*** dates are timestamps.

c. Edit the **WEB-INF/scripts/MinimalData.lst** file with a text editor and add the following line:

```
WEB-INF/xmldata/PriceTypeList
```

26. Load the data.

a. On UNIX, run:

```
loadDBFromXML.sh full <jdbc_jar_file.jar>
```

where <*jdbc_jar_file.jar*> is the full path to your JDBC driver. If you get a permissions error, then modify the permissions on the script to give yourself execution privileges.

Attention:	On Linux, if you see errors reporting that a network connection cannot be established to the database server, then check that you do not have Secure Linux enabled. If need be, navigate to **/etc/sysconfig/selinux** and make sure that the following is set: SELINUX=disabled

b. For Windows configurations using ODBC to connect to the Knowledgebase database server, run:

```
loadDBFromXML full
```

c. For Windows configurations with JDBC, run:

```
loadDBFromXML full <jdbc_jar_file.jar>
```

where <*jdbc_jar_file.jar*> is the full path to your JDBC driver.

Note that you can load just the minimal data by specifying “minimal” rather than “full” when you run this command.

27. Load the full segmentation data:

    a. On UNIX, run:

    ```
    loadSegDBFromXML.sh full <jdbc_jar_file.jar>
    ```

    where *<jdbc_jar_file.jar>* is the full path to your JDBC driver. If you get a permissions error, then modify the permissions on the script to give yourself execution privileges.

Attention:	On Linux, if you see errors reporting that a network connection cannot be established to the database server, then check that you do not have Secure Linux enabled. If need be, navigate to **/etc/sysconfig/selinux** and make sure that the following is set: SELINUX=disabled

    b. For Windows configurations using ODBC to connect to the Knowledgebase database server, run:

    ```
    loadSegDBFromXML full
    ```

    c. For Windows configurations with JDBC, run:

    ```
    loadSegDBFromXML full <jdbc_jar_file.jar>
    ```

    where *<jdbc_jar_file.jar>* is the full path to your JDBC driver.

28. Check the results by examining the log files in the directory in which you ran the data load command.
29. Rename the ***cmgt_app_home*/Sterling-8.0.1-def-RC-1.war** file to **Sterling.war** and deploy it to the servlet container using the steps described in "Deploying the Sterling Web Application" on page 27.
30. Restart the application server.
31. You should now be able to point your browser to the standard Sterling Multi-Channel Selling Solution URL and log in as the enterprise administrator user admin/admin.

Default XML Identity Setup

This section describes how to set up a default XML user and configure request IP-based filtering. To set up the filtering, you need to know the subnet masks of the client hosts that will send XML requests.

Requests coming into a storefront do not necessarily contain user information, especially if the requests' originating system and processing system are behind the same firewall or if the service provided by the Sterling Multi-Channel Selling Solution (using XML message relay) is a subset of services that belong in the DMZ.

Incoming XML messages posted to a storefront URL that contain the keyword **amsg** do not necessarily contain user information and are processed differently from URL's that contain the "standard" keyword **msg**. The **amsg** URL has the following form:

```
http://<server>:<port>/Sterling/amsg/<Storefront name>
```

You can set up a default XML user for the storefront to allow the system to create appropriate user credentials and process the incoming requests.

To prevent exposing sensitive data and to prevent modifications to existing data, the default XML user has only limited access to the system based on the entitlements configured in **Entitlements.xml**.The system creates appropriate credentials based on storefront-specific system properties and configurations. The default XML user requires only a user name and must be assigned to the Default XML Identity user type: no password is required.

The general steps are:

1. Configure the default XML user
2. Configure the trusted hosts' IP addresses

You set up the default XML identity from the Sterling Multi-Channel Selling Solution UI as the storefront administrator. You edit the **web.xml** configuration file to configure the trusted IP addresses.

Default XML Identity Configuration

To configure the Default XML Identity, perform the following steps:

1. Log in to your storefront as a storefront administrator.
2. Create a new storefront user of type Default XML Identity.
3. From the System Services page, set the following XML Messages properties:
   - Set *Should the system enable a default xml user identity?* to true. This sets the system property XMLMessages.EnableDefaultXMLIdentity to true. The default value is false.

- Enter a user name in the *Username of default identity for XML messages* field. This sets the default XML identity user name in the system property XMLMessages.DefaultXMLIdentity_UserName. The default value is null.

4. Log out and restart the server. This allows the system to recognize the new default XML identity and user name.

When Default XML Identity is enabled, the system uses the configured default XML user to build user credentials, even if the request specifies a user identity.

The user type of the default XML user must be EnterpriseDefaultXMLIdentity. If it is not, the system throws an exception during creation of the associated user credential.

The minimal data contains a default XML user: XMLGuest.

See the Sterling Multi-Channel Selling Solution Administration Guide for more information about enabling Default XML Identity and creating the Default XML User.

Trusted IP Address Configuration

You set up IP address filtering using the RequestIPFilter class, and use filter mapping in the **WEB-INF\web.xml** configuration file so that the client IP address is matched against entries from the list of trusted hosts. You edit the **web.xml** file to specify allowable IP addresses.

The **web.xml** file contains an entry that invokes the filter AnonXMLIdentityIPFilter when the request URL contains the keyword **amsg**. The default entry is similar to the following:

```
<filter>
 <filter-name>AnonXMLIdentityIPFilter</filter-name>
   <filter-class>com.comergent.dcm.core.filters.RequestIPFilter</fil-
ter-class>
 <init-param>
    <!-- specify comma separated list of IP addresses. Wildcard
    character '*' can be used. -->
    <param-name>AllowableHosts</param-name>
    <param-value></param-value>
 </init-param>
</filter>
```

Specify the trusted IP addresses separated by commas. You can use the wild card character ('*'). For example:

```
<filter>
```

```
 <filter-name>AnonXMLIdentityIPFilter</filter-name>
   <filter-class>com.comergent.dcm.core.filters.RequestIPFilter</fil-
ter-class>
 <init-param>
    <!-- specify comma separated list of IP addresses. Wildcard
    character '*' can be used. -->
    <param-name>AllowableHosts</param-name>
    <param-value>127.0.0.1, 10.62.*.*, 10.64.52.*, 10.68.55.5</param-
value>
 </init-param>
</filter>
```

The filter implementation gets the IP address of the sender and compares that IP address with the allowable IP address list specified in **web.xml**. If the IP address of the sender matches an IP address in the allowable IP address list, the request is processed. If there is no match, the request is filtered and no further processing is done.

Depending on your Sterling Multi-Channel Selling Solution implementation, there may be configuration issues to consider for IP address filtering. For example, if your Sterling Multi-Channel Selling Solution is deployed in a clustered mode, ensure that the load balancer is configured to propagate the client system IP address so that the filter can use this data to determine whether to serve the request or not. If the load balancer cannot be configured this way, seriously consider using a secondary load balancer for your deployment, to be used only by the internal network. For example, you would set up a secondary load balancer so that the fulfillment system posts messages to the secondary load balancer and the filter accepts only those messages coming from the secondary load balancer, ignoring others.

Default XML Identity Request Authentication

The request URL keyword **amsg** cues the MessagingServlet to authenticate requests differently from requests with the **msg** keyword. The MessagingServlet uses a combination of the **amsg** keyword and the system property settings for enabling and configuring the default XML user identity to determine how to authenticate an incoming request.

The following diagram illustrates the authentication sequence.

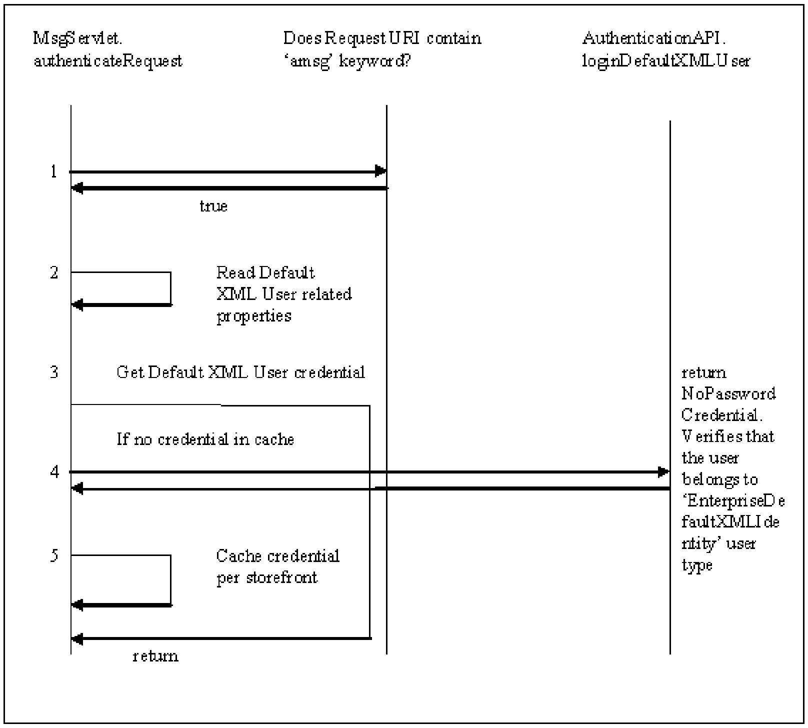

MsgServlet.
authenticateRequest
Does Request URI contain 'amsg' keyword?
AuthenticationAPI.
loginDefaultXMLUser
1
true
2
Read Default XML User related properties
3
Get Default XML User credential
If no credential in cache
4
return NoPassword Credential. Verifies that the user belongs to 'EnterpriseDefaultXMLIdentity' user type
5
Cache credential per storefront
return

FIGURE 6. Authentication Sequence

The authentication sequence is as follows:

1. The MessagingServlet determines how to authenticate a request. If the URL keyword is **amsg** and Default XML Identity is enabled, the MessagingServlet authenticates using the Default XML User credential and ignores any user information contained in the request.
2. The MessagingServlet gets the Default XML User-related system properties.
3. For first-time creation of the Default XML user credential, the AuthenticationAPI.loginDefaultXMLUser API is invoked. The implementation of this API checks whether the user for whom the credential is to be created belongs to the user type EnterpriseDefaultXMLIdentity. Only users belonging to this user type can be configured as a default XML user.
4. The Default XML user credential is cached for every storefront that has a valid configuration for Default XML user.

AuthenticationAPI

This section describes the authentication API. AuthenticationAPI helps log in a Default XML Identity user.

Methods:

- Credential loginDefaultXMLUser(Object authority, Credential authorityCredential, Object bearer, String namespace, String username)
  - authority: authority class object
  - authorityCredential: authority Credential object
  - bearer: bearer class
  - namespace: storefront id
  - username: name of the user for which the login should be attempted
  - return value: the Credential object that is created if the login succeeds.

Note:	A new Credential class, NoPasswordCredential, helps login a user based only on the username: a password is not required.

Database Server Steps

Depending on which database server you use with the Sterling Multi-Channel Selling Solution, perform the required steps in this section. See "Managing Database Connections" on page 86 for information about connection pooling, and "Sorting in Locales" on page 179 for information about data services configuration settings.

Support for Oracle Server

If you plan to use Oracle Server for your database server, then you must run the installOracle target as part of the installation of the Sterling Multi-Channel Selling Solution. This ensures that the Oracle JDBC drivers are in the deployed Web application.

If you plan to use the OCI driver to connect from the Sterling Multi-Channel Selling Solution machine to the Oracle Server, then you must make sure that the OCI driver is set up correctly. On a UNIX system:

1. Make sure that the OCI **liboci*jdbc.so** file is installed on the Sterling Multi-Channel Selling Solution machine. The “*” in the name of the file is the version number of the OCI library.

2. Make sure that the servlet container scripts ensure that the LD_LIBRARY_PATH includes the location of the OCI **liboci*jdbc.so** file and that ORACLE_HOME is set to point to the location of the Oracle client tools.
3. Make sure that the Oracle JAR file matches the version of the OCI library file:

TABLE 9. **OCI library and JDBC Driver Files**

OCI Library	JDBC JAR File
oci804jdbc.so	**Oracle.jar**
ocijdbc8.so	**Classes111.zip**

Support for SQL Server

If you are running the Sterling Multi-Channel Selling Solution on a Windows system and plan to use SQL Server 2003 for your database server, then you must perform the following steps.

1. Copy the appropriate DLL file from ***debs_home*****/Sterling/WEB-INF/lib/winnt/** to the **C:\WINNT\system32\** directory. Note that the installMsSql target will do this on a machine running the SDK, but if your deployment machine is different from the SDK machine, then you must do this step manually.
2. If you plan to support data in locales other than en_US, then you must consider how basic searches are performed. If you do not want searches to differentiate between upper and lower cases instances of the same character, then you must provide values for the elements LowerCase and UpperCase of the Microsoft element of the **DataServices.xml** file, contained in the **WEB_INF\lib\cmgt-dataservices.jar** file.

   Set LowerCase to "LOWER(" and UpperCase to "UPPER("; for example:
   ```
   <UpperCase controlType="text" runtimeDisplayed="true"
   ChangeOnlyAtBootTime="true" visible="true" boxsize="45"
   displayQuestion="UpperCase SQL Function" defaultChoice=""
   help="Enter the SQL function that converts strings to uppercase
   for the selected database.">UPPER(</UpperCase>
   ```
   Note that the use of the UPPER and LOWER functions in searches means that indexes on the tables are not used and this can result in reduced performance.
3. If more than one deployment of the Sterling Multi-Channel Selling Solution is accessing the same Knowledgebase on SQL Server, then you must set a two-digit server ID for each deployment.

a. If the machines are not clustered, then set the ServerId element in the **prefs.xml** file so that each has a unique integer value: 01, 02, and so on.

b. If the machines are clustered, then you must modify the servlet container command or script that starts the servlet container on each machine so that a Java system property is set: Comergent.DataServices.General.ServerId. This should be set on each machine so that each has a unique value: 01, 02, and so on.

   For example, in a Tomcat installation, you can modify the batch file to include:

```
set JAVA_OPTS=-DComergent.DataServices.General.ServerId=02
```

Managing Database Connections

This section describes how to manage the connections between the Sterling Multi-Channel Selling Solution and the Knowledgebase.

Configuration Files

The following files manage the configuration of the data services layer:

- **DataServices.xml**: this file is contained in the **WEB_INF\lib\cmgt-dataservices.jar** file that ships with the Sterling Multi-Channel Selling Solution. This file specifies values for all the data services properties unless they are overridden by the **prefs.xml** configuration file.
- **prefs.xml**: this file contains the properties and their values created during the installation process. In addition, if you make changes to the system properties through the Sterling Multi-Channel Selling Solution administration UI, then the changes are persisted to this file.

Connection Pooling

This section answers some common questions about connection pooling.

- What is the purpose of connection pooling?
- How does Sterling’s connection pooling work?
- Why are there separate query and update connection cools?
- How do I validate connections prior to reuse?
- How can I limit the number of connections used?
- How can I free up connections when demand drops?

- What happens when connection limits are reached?
- Why are the connection limits on the data source?

Common Problems

- My database requests fail with a "connection reset by peer" message
- My database connections are not being released when traffic drops
- I share a database with other applications. I cannot allow the Sterling Multi-Channel Selling Solution to use more than n connections

What is the purpose of connection pooling?

Establishing a database connection is typically processor-intensive. The use of connection pools allows us to maintain a set of open connections for use by database requests. These connections can then be allocated to short duration SQL requests and then immediately returned to the pool for re-use.

How does Sterling's connection pooling work?

The Sterling Multi-Channel Selling Solution maintains these logical pools of connections:

- Pool of message-based connections. This pool is simply a HashMap that mates a message version to an appropriate Message-based DataService. One DataService instance is shared by all requests requiring that message version.
- Pool of database connections. This pool is used for all database requests. When a data bean *persist()* or *restore()* method is invoked, we retrieve a connection from the pool; process the operation; then return the connection to the pool for reuse.

  In past releases the Sterling Multi-Channel Selling Solution supported sharing a database connection across multiple concurrent requests. Not all databases are capable of supporting this functionality. In addition, performance testing has shown that an expensive SQL request can drastically impact the performance of all requests sharing the same connection. Based on these issues, the Sterling Multi-Channel Selling Solution has eliminated support for sharing of Query connections.

In Release 6.3 and later releases, there is one Query connection pool and one Update connection pool for each SQL-based data source. This allows further tuning and optimization of the connection pools.

Why are there separate query and update connection cools?

The use of separate pools for Query and Update Connections allows the Sterling Multi-Channel Selling Solution to optimize connections for read-only access.

How do I validate connections prior to reuse?

The following properties control connection timeout and validation:

- The ConnectTimeout element provides a timeout setting for connections in the pools. The value is the number of minutes for a connection to timeout. For example, if you set this value to “1”, then if a connection has been unused for more than one minute, it is validated before being used. A setting of 0 means that connections do not timeout.
- The ReconnectOnTimeout element controls what is done when a connection timeout.
    - A setting of “true” indicates that when a connection times out it will automatically reconnect the next time it is retrieved from the pool.
    - A setting of “false” indicates that a connection timeout will result in the connection being validated prior to reuse. If the validation fails, then a reconnect will occur.

How can I limit the number of connections used?

Each DataSource specified in the **DataServices.xml** configuration file supports a MaxConnections property. This specifies an absolute upper limit on the number of connections that will be used. A setting of “-1” indicates there is no limit. The **DataServices.xml** file is contained in the **WEB_INF\lib\cmgt-dataservices.jar** file.

How can I free up connections when demand drops?

Each DataSource specified in the **DataServices.xml** configuration file also supports a MaxPoolSize property. This provides a soft limit on the number of connections that will be pooled. A setting of “-1” indicates there is no limit. The pool size is not an absolute limit, but as connections are released the pool will gradually move back down to its maximum size.

The Sterling Multi-Channel Selling Solution does allow the number of connections to grow beyond the maximum pool size, but when the number of free connections exceeds a preset limit we will begin releasing connections until the number of connections eventually drops back to the maximum pool size. We do this gradually to avoid excessive connection requests when pool is at the boundary.

What happens when connection limits are reached?

If a connection is requested from the pool, but no free connections are available, then we would normally create a new connection. If the connection limit is reached, then we will instead wait for a connection to be returned to the pool.

Why are the connection limits on the data source?

Providing connection limits for each data source provides greater flexibility in allocating connection resources. For example, this allows you to limit the number of connections to a back-end ERP system, while providing higher limit when accessing the primary database server.

Common Problems

My database requests fail with a "connection reset by peer" message

This error is normally a result of either the database server timing out the database connection, or of the network connection being timed out by a firewall. This problem can be resolved by setting the ConnectionTimeout element to ensure validation of connections that exceed the timeout.

My database connections are not being released when traffic drops

Setting the MaxPoolSize property on the data source will allow the number of database connections to drop back to predefined limits as connections are freed.

I share a database with other applications. I cannot allow the Sterling Multi-Channel Selling Solution to use more than n *connections*

Setting the MaxConnections property on the data source will allow you to limit the maximum number of database connections used by the Sterling Multi-Channel Selling Solution.

Pagination Settings

The Sterling Multi-Channel Selling Solution supports the use of paginated data sets so that long lists can be displayed one page at a time. This functionality is implemented by saving a set of files to the Sterling Multi-Channel Selling Solution machine's file system. These represent the pages of data objects to be paged through. The location of the paginated file sets is determined by the rsCachePath element in the **DataServices.xml** file, contained in the **WEB_INF\lib\cmgt-dataservices.jar** file. The rsCachePathIsAbsolute element is used to specify whether the value of the rsCachePath element should be treated as a relative or absolute path. By default, its value is "false" and so the path is treated as being relative to ***debs_home*****/Sterling/**. The *adjustFileName()* method call is used to resolve this location to an absolute location in the servlet container's file system.

If your implementation of the Sterling Multi-Channel Selling Solution uses a cluster of servlet containers, then the location of the pagination directory must be accessible to all members of the cluster. See "High Availability and Clustering" on page 93.

Setting the Product Catalog Root

The product catalog in the Sterling Multi-Channel Selling Solution is structured as a tree of product categories. As a user browses the product catalog, they can navigate from each product category down to its child product categories. Similarly, an administrator can navigate through the product category hierarchy as they create and modify product categories and products.

The root node of the product category is created as part of the minimal data set. All product categories are descendants from this root node. When the product category is displayed, the root node label is: "Product Categories". This label is specified as the MTMenuText variable in the **cmgtProdMgrTreeViewParam.js** file located in the ***debs_home*****/Sterling/en/US/js/** directory. As part of implementation, you can change this value to appropriate text such as "Product Catalog" or "Departments".

Note that if your implementation of the Sterling Multi-Channel Selling Solution supports more than one locale, then you must make the corresponding changes in the same file in the other locale directories.

Setting the Session Timeout

Servlet containers and Web applications attach a session to each user interaction with the server. By this means, they can maintain information from one request to another as a user interacts with the application. To help ensure that a user's browser is not used by an unauthorized user, the servlet container will mark a session as being invalid once a certain time has elapsed from the time when the session was last accessed. This is referred to as the session timeout period. Sessions automatically become inactive if the time from the last access exceeds the session timeout setting.

You can set the session timeout period in the Sterling Multi-Channel Selling Solution **web.xml** configuration file using the session-timeout element. For example, to timeout sessions after 30 minutes, set the element to:

```
<session-timeout>30</session-timeout>
```

When setting the session timeout period, bear in mind the following:

- The longer the time out, the greater the risk that the servlet container will run out of memory. Each session takes up space in memory, and when objects are added to the session, then the memory usage increases. Often, users may not actively log out: their session will stay resident in memory until the servlet container times it out. If your Web site is likely to see heavy user traffic, then bear in mind this memory consumption when determining JVM memory settings.
- The longer the timeout, the greater security risk presented: either by an unauthorized person using an unattended Web browser or by an unauthorized person spoofing a session simply by guessing its session ID.
- The session timeout period must be sufficiently long to enable users to complete their tasks. If the tasks include activities such as using a third-party Web application or obtaining information from a third-party source, then allow for this amount of time so that a user is not inadvertently timed out of the Sterling Multi-Channel Selling Solution.

  Bear in mind that in some use cases, the cost to a user of losing their session may be high: if they have created a product inquiry list by punching out from a procurement application using Sterling MarketLink, then when their Sterling Multi-Channel Selling Solution session times out, they will lose access to any data created before the timeout.

For these reasons, we suggest setting a session timeout value of 30 (30 minutes). However, you must assess the needs of your implementation and select a value accordingly.

Modifying the URL for the Web application DTD

When you start the servlet container, the Sterling Multi-Channel Selling Solution is loaded as a Web application. The **web.xml** file configuration file is read to determine the basic configuration of the application. The **web.xml** file is validated against a DTD specified by its web-app element.

By default, the validating DTD is at the URL:

```
http://java.sun.com/dtd/web-app_2_3.dtd
```

However, access to this URL can be limited either by your network status or by Sun Microsystems. As an implementation step, we recommend that you modify the validating URL to point to a copy of the DTD whose location is assured by your implementation.

Our suggested solution is to use a relative URL to reference the DTD within the Sterling Multi-Channel Selling Solution context. For example:

```
/WEB-INF/lib/web-app_2_2.dtd
```

Note that the form of this relative URL is servlet container-specific.

Alternatively, you can add the DTD to a Web server and point to this location. For example, if you are using a Web server to act as a front-end to the Sterling Multi-Channel Selling Solution, then put the DTD on this Web server.

Managing Memory

In general, you should allocate as much memory as possible to the JVM running your application server. Typically, this is done by modifying the configuration parameters that are used to start the Java process, say:

```
-Xms256M -Xmx512M -XX:MaxPermSize=128M
```

However, if your system is likely to experience heavy load at times, then you can use a Sterling Multi-Channel Selling Solution configuration parameter to ensure that the system can recover from a burst of memory-intensive activity.

Set the memoryThreshold element of the C3_Commerce_Manager element of **Comergent.xml** to an integer value between 0 and the maximum allocated memory size (in Kilobytes). When memory usage exceeds this value, then new requests will be refused with the HTTP status of 503. The default value, -1, disables the parameter.

For example, suppose that you have set the maximum memory to -Xmx512M (524,288K). Suppose that you set memoryThreshold to 498074. Then when memory usage exceeds 498,074K, new requests are refused until memory usage has dropped back down to below this value.

Configuring Ehcache

The Sterling Multi-Channel Selling Solution uses Ehcache to manage data caching, both for clustering and for the global application cache. Out of the box, Ehcache manages pre-configured caches for applications such as pricing, product categories and features. You configure new caches and modify cache properties such as the amount of time to cache data and the maximum number of elements in memory by editing the **WEB-INF\properties\EhCache.xml** file.

High Availability and Clustering

The Sterling Multi-Channel Selling Solution can be deployed in a distributed environment in which more than one individual instances of the Sterling Multi-Channel Selling Solution run as a cluster. This provides support for ensuring high availability of the Sterling Multi-Channel Selling Solution and to support fail-over of individual machines. See CHAPTER 26, "Installing a Clustered Implementation" for more information.

Sharing Directories

In some deployments of the Sterling Multi-Channel Selling Solution, for example a clustered deployment, you must specify the location of directories to be used for uploaded and generated files. The locations of these directories is specified using the **web.xml** file to set context parameters.

You have these sets of attributes for the directories to specify:

- share-noshare: share directories can be accessed by two or more machines, noshare directories should be accessed only by the machine whose **web.xml** file specifies the location.
- public-private: public directories must be accessible by the Web server serving the static content, private directories should not be.
- loadable-noloadable: loadable directories can be used to upload files, noloadable directories should not be used for uploaded files.

The same directory can be used for more than one of these combinations of choices.

At minimum, you must specify the location of the share.public.loadable and share.public.noloadable directories. If you have two or more machines in a cluster, then these directories must be accessible from all of the cluster machines.

The **web.xml** file lets you specify how a front-end Web server can access files in the public directories. Use the WebPathToPublicLoadableWritableDirectory element to map a Web server virtual path to the directory identified by the share.public.loadable element. Use the WebPathToPublicNoLoadableWritableDirectory element to map a Web server virtual path to the directory identified by the share.public.noloadable element. These elements should reflect the Web server settings used to specify virtual paths.

To set these directories up, you typically perform these steps:

1. Select one of the machine as the “primary machine”. Allocate a directory on this machine to provide the shared location.
2. Share this location so that all member of the cluster have access to it:
   - Windows: share this directory to the other machines
   - UNIX: use NFS to share the directory
3. On all machines, mount the file system so that all cluster members have the same mount point to this directory. For example:

   ```
   /DEBS_shared
   ```
4. Under **DEBS_shared/**, create a sub-directory for each of the categories shown in the configuration file (loadable, writable, and so on). For example:

   ```
   /DEBS_shared/lw
   ```

   and set that value in the configuration file. For example:
   ```
   <loadable ...>/DEBS_shared/lw</loadable>
   ```

Directory and File Organization

When the **Sterling.war** Web application is deployed to the servlet container, it is deployed into a directory, ***debs_home***, that you specify during the deployment or which the servlet container sets. This section describes the organization of the sub-directories under ***debs_home***.

Beneath this directory, a sub-directory is created for the Sterling Web application when the Web application is deployed. This directory is the Web application directory for the Sterling Multi-Channel Selling Solution. We refer to it as ***debs_home*****/Sterling/**. It contains:

- A locale directory for each supported locale. Each locale may be expressed as <*la*>_<*CO*>, where *la* is one of the standard language codes and *CO* is one of the standard country codes, for example: en_US or fr_CA. For a locale, the corresponding directory is ***debs_home*****/Sterling/*****la*****/*****CO*****/**. This directory contains:
  - **css/**: holds the cascading style sheets used by the Sterling Multi-Channel Selling Solution.
  - **images/**: holds common images used by the Sterling Multi-Channel Selling Solution.
  - **js/**: holds Javascript libraries used by the Sterling Multi-Channel Selling Solution.

    - **htdocs/**: holds the HTML templates, images, and online help for the Sterling Multi-Channel Selling Solution.
- **dXML/**: holds the DTDs for the dXML message types.
- **htdocs/**: a directory for content that can be served up directly by the servlet container or Web server. Content stored here should not be locale-specific.
- **j2ee/**: a directory to hold local copies of the J2EE DTDs. See "Modifying the URL for the Web application DTD" on page 91 for more information.
- **WEB-INF/**: holds all the configuration files used by the server. It contains the following subdirectories:
    - **bizobjs/**: holds the business object DTDs. These DTDs are used to validate XML messages. The DTDs can be generated automatically by the generateDTD target provided by the SDK.
    - **classes/**: holds the Sterling Multi-Channel Selling Solution Java classes.
    - **commerceone/**: used as part of Commerce One integration.
    - **converters/**: holds the configuration files used in message conversion.
    - **data/**: holds data provided as part of the reference implementation.
    - **extralib/**: holds class libraries that are needed for implementation work, but that should not be used at runtime.
    - **integrator/**: holds the configuration files for Sterling Integrator.
    - **lib/**: holds the Java class libraries used by the Sterling Multi-Channel Selling Solution Web application.
    - **lib/winnt/**: this holds any Windows-specific DLL files that are required.
    - **messages/**: holds the DTDs for the Sterling message family. See the *Sterling Multi-Channel Selling Solution Reference Guide* for more information.
    - **properties/**: holds the **Comergent.xml** file and the other configuration files used to set the configuration of the server.
    - **reports/**: holds the files required for Sterling Analyzer.
    - **rosettanet/**: contains the DTD and XML files that define the RosettaNet messages supported by the Sterling Multi-Channel Selling Solution.

- **schema/**: holds the XML files that specify the schema for your implementation. See CHAPTER 21, "Integrating with External Data Sources" for more information.
- **stylesheets/**: holds the XSL files used to translate messages from one message family to another.
- **templates/**: holds the text templates used to generate messages such as email notifications.
- **web/**: holds most of the JSP pages, HTML pages and support files for the applications. It has the following structure:

  /*locale*/ directories for each of the Sterling Multi-Channel Selling Solution applications. Each locale supported by your implementation of the Sterling Multi-Channel Selling Solution must have its own set of JSP pages in its corresponding locale directory.

  Each locale may be expressed as <*la*>_<*CO*>, where la is one of the standard language codes and CO is one of the standard country codes, for example: en_US or fr_CA. For a locale, the corresponding directory is ***debs_home*/Sterling/WEB-INF/web/*la*/*CO*/**.
- **x509/**: holds the certificates used to authenticate SSL sessions.

Cron Job Setup

The Sterling Multi-Channel Selling Solution provides a number of cron jobs out of the box. These cron jobs require authentication with a username and password in order to run. The default username/password combination is admin/admin. After installing the Sterling Multi-Channel Selling Solution or upgrading or migrating from a previous release, you can change the username and password of the user who administers cron jobs. If you change the username and password, ensure that you also update the authentication information for each cron job and that the user is assigned appropriate roles to allow running the cron jobs.

The following cron jobs require that the user has the Enterprise.Administrator role:

- Maintain Indexsets
- Product Sync
- User Sync

The following cron jobs require that the user has the Enterprise.SegmentManager role:

- Nightly Segments Build
- Reprocess Segments

Setting Up Apache as a Front-end to Tomcat

This section describes how to set up an instance of Apache Web Server Release 2.0.59 so that it can act as a front-end to a deployment of the Sterling Multi-Channel Selling Solution on Tomcat 6.x. It uses the JK 1.2 connector supplied by the Apache Jakarta Project.

This section assumes that Apache and Tomcat are installed on two different machines, referred to as the Web server machine and servlet container machine respectively.

Prerequisites

1. Install Apache Web Server Release 2.0.59 on the Web server machine.
2. Deploy the Sterling Multi-Channel Selling Solution into the instance of Tomcat running on the servlet container machine.
3. You should confirm that both Apache and Tomcat can be started individually with no error. In particular, make sure that the deployment of the Sterling Multi-Channel Selling Solution in Tomcat works correctly using the Tomcat port.

Overview

JK 1.2 is a connector that connects an Apache instance with a Tomcat instance. This allows Apache to serve as a front-end Web server for Tomcat. There are several advantages to this kind of setup:

- You can configure Apache to manage page expiration (reducing the number of HTTP requests).
- You can configure Apache to compress responses (reducing the number actual bytes transmitted).

Once the connector is set up and configured to work properly, a typical request flow is as follows (using default ports):

1. The browser connects to Apache’s port 80 and submits its request.
2. Apache determines if the incoming URL needs to be managed by the JK 1.2 connector, mod_jk.

3. If so, then Apache initiates an AJP 1.3 connection to Tomcat's port 8009. The request is now sent to Tomcat.
4. Tomcat processes the request and returns the response through the same AJP 1.3 connection.
5. Apache in turn relays the same response to the browser.

Configuring Apache to Use mod_jk

1. Download a copy of **mod_jk-apache-2.0.58.so** for Apache 2.0.58 and later. At the time of release, the location is similar to http://tomcat.apache.org/download-connectors.cgi. For the rest of these instructions, we assume that you rename this file to **mod_jk.so**.
2. Put the **mod_jk.so** file in the Apache Web server ***apache_home*/modules/** directory.
3. Edit the ***apache_home*/conf/httpd.conf** file as follows:
   a. Add the following line in the LoadModule section:

   ```
   LoadModule jk_module modules/mod_jk.so
   ```

   Take care to provide the exact name of the mod_jk module file.

   b. Add an IfModule element to force Apache to set up the Tomcat servlet container connection and access to the Sterling Multi-Channel Selling Solution web application:

   ```
   <IfModule mod_jk.c>
       JkWorkersFile apache_home/conf/workers.properties
       JkLogFile apache_home/logs/mod_jk.log
       JkLogLevel info
       JkMount /Sterling/* ajp13
   </IfModule>
   ```

   c. Add the following line to the very end of httpd.conf:

   ```
   Include /tomcat-home/conf/auto/mod_jk.conf
   ```

4. Ensure that the sample Tomcat ***tomcat_home*/conf/workers.properties file** is similar to the following:

   ```
   # Set properties for Tomcat
   worker.list=worker1
   worker.worker1.port=8009
   worker.worker1.host=<servlet_container_machine_name>
   ```

```
worker.worker1.type=ajp13
```

Replace <*servlet_container_machine_name*> with the name of the servlet container machine.

5. Ensure that the corresponding ***apache_home*/conf/workers.properties** file is similar to the following:

```
# Define 1 real worker using ajp13
worker.list=ajp13
# Set properties for worker1 (ajp13)
worker.ajp13.type=ajp13
worker.ajp13.host=<servlet_container_machine_name>
worker.ajp13.port=8009
worker.ajp13.lbfactor=50
worker.ajp13.cachesize=10
worker.ajp13.cache_timeout=600
worker.ajp13.socket_keepalive=1
worker.ajp13.recycle_timeout=300
```

Replace <*servlet_container_machine_name*> with the name of the servlet container machine.

Configure Tomcat to Use mod_jk

By default, Tomcat is pre-configured to listen on port 8009 for ajp13 connections. Ensure that the following entry is in the Tomcat ***tomcat_home*/conf/server.xml** file:

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
    <Connector port="8009"
    enableLookups="false" redirectPort="8443" protocol="AJP/1.3" />
```

Edit Tomcat’s ***tomcat_home*/conf/server.xml** file:

1. Add the following line to the Listeners section:

```
<Listener className="org.apache.jk.config.ApacheConfig"
modJk="<apache-home>/modules/mod_jk.so" />
```

Starting Apache and Tomcat

1. Start up Apache, then start up Tomcat.
2. Try:

```
http://<web server>/Sterling/en/US/enterpriseMgr/matrix
```

to verify that you can access the Sterling Multi-Channel Selling Solution through Apache.

Setting up Apache to Support SSL

If you set up Apache as a front-end to Tomcat, then you can use the SSL capabilities of Apache to manage secure access to the Sterling Multi-Channel Selling Solution. The following steps provide an outline as to how to do this. Note that we do not provide a compiled binary of the Apache SSL module. You must either obtain this from a third-party such as: http://hunter.campbus.com/, or build it yourself using the OpenSSL source obtained from: http://www.openssl.org/source/. Once you have created **mod_ssl.so** and copied it to ***apache_home*/modules/**, then follow these steps:

1. Uncomment in the following line in ***apache_home*/conf/httpd.conf**:

   ```
   LoadModule ssl_module modules/mod_ssl.so
   ```

   and

   ```
   <IfModule mod_ssl.c>
       Include conf/ssl.conf
   </IfModule>
   ```

2. Create the file ***apache_home*/conf/ssl.conf**. This file is where you specify your SSL configuration using the SSL directives. It should look something like this:

   ```
   Listen 443
   <VirtualHost _default_:443>
   ServerName http://www.example.com
   SSLEngine on
   SSLCertificateFile /usr/local/apache2/conf/server.cert
   SSLCertificateKeyFile /usr/local/apache2/conf/server.key
   </VirtualHost>
   ```

3. Obtain or generate the certificates and keys for your site. You can use the openssl utility to generate a self-signed key and certificate using commands like this. First create the key by using:

   ```
   openssl req -new -nodes -out server.csr
   -keyout server.key -config openssl.cnf
   ```

   Then use the key to generate the certificate:

   ```
   openssl x509 -in server.csr -out server.crt -req
   -signkey server.key -days 365 -set_serial 1 -config openssl.cnf
   ```

   The -config parameter points to your **openssl.cnf** configuration file that can be used to maintain OpenSSL configuration information.

4. Copy the key and certificate to the location specified by the SSLCertificateFile and SSLCertificateKeyFile properties of the **ssl.conf** file.

5. Restart Apache.

Keep Alive Settings

In some circumstances, problems have been reported with Apache and SSL such as slow and dropped connections. If you encounter these, then consider these steps:

1. Make sure that the setting for KeepAlive is On in ***apache_home*/conf/httpd.conf**:

   ```
   KeepAlive On
   ```

   It appears that this setting is set to Off as default in some distributions of Apache.

2. Older versions of IE, in particular IE 5.x, have a bug in the SSL/TSL shutdown and keepalive feature. A work-around for these bugs is to configure Apache's SSL to behave in a non-standard way for these connections. In ***apache_home*/conf/ssl.conf**, add the following lines if they are not there already:

   ```
   SetEnvIf User-Agent ".*MSIE.*" \
   nokeepalive ssl-unclean-shutdown \
   downgrade-1.0 force-response-1.0
   ```

Filtering Static Content

In general, you should use a servlet container in conjunction with a Web server. The Web server can be used to serve other content for your Web site. In addition, the Web server can be used to serve static content from the Sterling Multi-Channel Selling Solution. In this way, you can enhance the performance of your Web site.

Setting up Apache to Serve Static Content

If you have Apache running as a Web server in front of your servlet container, then you can make use of Apache's capabilities to serve static content. In this section, we show how to use the expires_module module to mark images so that a client's browser caches images rather than re-requesting them each time a page displays the image. In particular, this approach can be used to prevent image-flicker if a user has their browser settings such that images are re-loaded from the server on every visit to the page.

These instructions assume that the Apache Web server and the Tomcat servlet container are running on different machines.

Follow these steps:

1. Edit the Apache **httpd.conf** configuration file to add or uncomment:

   ```
   LoadModule expires_module modules/mod_expires.so
   ```

2. Add the following expires rules:

   ```
   ExpiresActive On
   ExpiresByType image/gif "access plus 1 day"
   ExpiresByType image/jpg "access plus 1 day"
   ExpiresByType text/css "access plus 1 day"
   ExpiresByType text/js "access plus 1 day"
   ```

   In these lines, you are specifying that the expires module is active, and that by default all the static content served by the Apache Web server should be cached by browsers for one day after accessing it. You can change these settings to meet the needs of your deployment of the Sterling Multi-Channel Selling Solution.

3. Restart the Apache Web server.

Note:	Note that under certain circumstances, this may give rise to unwanted behavior. For example, if partner administrators frequently upload partner logos in the form of GIF files, then some storefront users who have the older version of the GIF file already cached will not see the new version of the GIF file until a day passes.

Creating a NSAPI Filter

We provide a small file of C code that can be used to ensure that certain files are served by the Web server rather than by the iPlanet Application Server. It uses the NSAPI.

1. Locate the **ctrans.c** file under ***debs_home***/.

2. Compile it to a dynamic library.

   For Solaris, the compile command is:
   ```
   gcc -DXP_Unix -DMCC_HTTPD -DNET_SSL -DSOLARIS -D_REENTRANT -Wall -
   c module.c * ld -G module.o -o module.so
   ```

   For Windows, the compile command is:
   ```
   cl -LD -MD -DMCC_HTTPD -DXP_WIN32 -DNET_SSL module.c -
   Insapi\include /link nsapi\examples\libhttpd.lib
   ```

3. Add the module to the NameTrans directive in the Web server's **obj.conf** configuration file as follows:

   a. Find the block where all the Inits are declared. Add line:

   ```
   Init fn="load-modules" shlib="/container_home/local/nsapi/comer-
   gent.so" funcs="handle_comergent_static"
   ```

      Replace the string /*container_home*/ with the appropriate path.

b. Find the block:

   ```
   <Object name="default">
   ```

c. Add the line:

   ```
   NameTrans fn="handle_comergent_static"
   ```

   By default, use the following values for the parameters:

   ```
   . prefix /NASApp/Comergent/
   . newPrefix /container_home/ias6/ias/APPS/modules/Comergent/
   . list *.css, *.gif, *.js, *.jpg
   ```

   If you need to override any of the above values, then append them to the “NameTrans” line. For example:

   ```
   NameTrans fn="handle_comergent_static"
   newPrefix="container_home/ias6/ias/TEST/modules/Comergent/"
   ```

Compressing Output From the Sterling Multi-Channel Selling Solution

If network performance is a high concern, then one step that you can take is to configure the Sterling Multi-Channel Selling Solution so that it returns compressed output to the users’ browsers, and the browsers decompress the output to render the page. This section describes how to use Apache to do this. Note that an alternate approach is to use Servlet Specification 2.3 filter to perform the compression.

These steps assume that you have set up Apache as a front-end to the servlet container in which the Sterling Multi-Channel Selling Solution is deployed. For example, see "Setting Up Apache as a Front-end to Tomcat" on page 97.

1. Edit the Apache **httpd.conf** configuration file to add or uncomment:

   ```
   LoadModule deflate_module modules/mod_deflate.so
   LoadModule headers_module modules/mod_headers.so
   ```

2. Copy the following text into ***apache_home*/conf/httpd.conf**. Putting it at the bottom of the file is fine.

   ```
   <Location /Sterling>

   # Insert filter
   SetOutputFilter DEFLATE

   # Netscape 4.x has some problems...
   BrowserMatch ^Mozilla/4 gzip-only-text/html

   # Netscape 4.06-4.08 have some more problems
   BrowserMatch ^Mozilla/4\.0[678] no-gzip
   ```

```
# MSIE masquerades as Netscape, but it is fine
# BrowserMatch \bMSIE !no-gzip !gzip-only-text/html

# NOTE: Due to a bug in mod_setenvif up to Apache 2.0.48
# the above regex won't work. You can use the following
# workaround to get the desired effect:
BrowserMatch \bMSI[E] !no-gzip !gzip-only-text/html

# Don't compress images
SetEnvIfNoCase Request_URI \.(?:gif|jpe?g|png)$ no-gzip dont-vary

# Make sure proxies don't deliver the wrong content
Header append Vary User-Agent env=!dont-vary

</Location>
```

If necesarry, change the context string “/Sterling” to the name of the context used for the Sterling Multi-Channel Selling Solution.

CHAPTER 6

Creating and Populating the Knowledgebase

This chapter describes how you create the Knowledgebase using the standard Sterling Multi-Channel Selling Solution schema. You must run the schema creation and data scripts to create and populate the Knowledgebase in your designated database server.

These steps enable you to test that the installation of the Sterling Multi-Channel Selling Solution has been successful. By the end of this chapter you will be able to log in to the Sterling Multi-Channel Selling Solution and verify that the basic functionality works.

If you are upgrading your installation of the Sterling Multi-Channel Selling Solution from an earlier release, then you can migrate the data to a Release 8.0 Knowledgebase. See "Data Migration" on page 117 for more information.

CHAPTER 9, "Customizing Your Sterling Multi-Channel Selling Solution", shows you how to customize the Sterling Multi-Channel Selling Solution and meet the needs of your implementation.

Gathering the Database Information

Identify the connection information for the database you are using. This includes:

1. Identify the connection information required to connect from the Sterling Multi-Channel Selling Solution machine to the database server.

- For an Oracle Server, this is the machine name or IP address of the database server, the port at which the database server is listening, and the SID (name of the database instance) of the database server. You must create a TNS alias for the database server on the Sterling Multi-Channel Selling Solution machine.
- For a SQL Server, this is the machine name or IP address of the database server.

2. Establish a database userid on the database server which is used by the Sterling Multi-Channel Selling Solution.

   Use this userid to perform all the Sterling Multi-Channel Selling Solution-related calls to the database. This userid must have sufficient privileges to create and modify database tables.

   You may use one of your existing database userids. However, we suggest that you set up a database userid that is dedicated to Sterling Multi-Channel Selling Solution-related tasks.

3. Check that you can connect to the database server using the userid and connection information:

   - For Oracle, you must be able to connect from the Sterling Multi-Channel Selling Solution machine to the database server using SQL*Plus and the TNS alias.
   - For SQL Server, you must be able to connect using OSQL and the JDBC data source name and userid.

Creating the Knowledgebase Schema

You run the schema creation script to create the Knowledgebase in your designated database server. You can run the schema creation script as a batch file as described in this section or from the SDK. See "To Run the Schema Creation Script with the SDK" on page 108.

Attention: Running the schema creation scripts directly is no longer supported. You must run the createDB target as described in "Installing the Sterling Multi-Channel Selling Solution Using the SDK" on page 43.

Before using the schema creation scripts and XML Loader scripts, you must decide what locales your implementation will support. See CHAPTER 11, "Internationalization" for more information about supporting locales and see

"Internationalization and Support for Locales" on page 112 for information about the locales created with the default scripts. You should remove any locales from the scripts that you do *not* intend to support. To begin with, you can try just creating the “en_US” locale and adding more as the implementation progresses.

Creating the Schema

Attention:	To run the database schema creation target, you must have the correct database client software installed: SQLPLUS for an Oracle server installation, or OSQL for Microsoft SQL Server,.

The schema creation script is a batch file that connects to the database server and then runs a sequence of SQL scripts to create the database objects.

- If you are running against an Oracle Server database server, then run one of the following scripts from the ***debs_home*/Sterling/** directory:
  - **OracleCreateSchema.bat** (for Windows)
  - **OracleCreateSchema.sh** (for UNIX)

  By default, the Oracle schema creation script creates indexes for tables in a separate tablespace called INDX. You can choose to create the indexes in the same tablespace as the main schema or in a tablespace of your choice. To do either of these two choices, you must edit the **oracle_indexes.sql** SQL script.
- If you are running against a Microsoft SQL Server database server, then run **MssqlCreateSchema.bat** from the ***debs_home*/Sterling/** directory. Make sure the database is the default database for the userid you use.

Locales and Loading Data Using the XML Loader

The current schema creation script creates seven locales as part of the table creation script. You should review this part of the table creation script and modify it to remove locales if need be before running the script. If you want to add locales, then follow the instructions provided in CHAPTER 11, "Internationalization".

For each locale, you must take care to ensure that the correct value in the DB_SORT_LOCALE_NAME column of the CMGT_LOCALE table is set. The following table summarizes the most frequently used values for this column:

TABLE 10. **Database Sorting Settings**

Database	Locale	Value
Oracle	en_US	BINARY
	de_DE	XGERMAN
	fr_FR	XFRENCH
	fr_CH	BINARY
	de_CH	BINARY
	ja_JP	BINARY
	zh_TW	BINARY
SQL Server	en_US	Latin1_General_BIN
	de_DE	FRENCH_CS_AS
	fr_FR	FRENCH_CS_AS
	fr_CH	SQL_Latin1_General_CP1_Cl_AS
	de_CH	SQL_Latin1_General_CP1_Cl_AS
	ja_JP	SQL_Latin1_General_CP1_Cl_AS
	zh_TW	Chinese_Taiwan_Stroke_CS_AS

For other locales, please contact your Sterling Support representative for further information.

To Run the Schema Creation Script with the SDK

As an alternative to running the schema creation scripts as batch files, you can also run the schema creation script from within the SDK as follows:

1. Edit the appropriate SDK project ***.properties** file to enter the connection information used by the createDB target. Typically, during an implementation cycle, this is the ***project*_dev.properties** file to be found in the ***sdk_home*/projects/*project*/templates/** directory.

   a. To create the Knowledgebase on an Oracle database server, enter:

```
DB_TYPE=Oracle
ORACLE_URL=jdbc:oracle:thin:@<Machine>:<Port>:<SID>
ORACLE_USERNAME=<Username>
ORACLE_PASSWORD=<Password>
ORACLE_DATABASE=<TNS alias>
```

b. To run the Knowledgebase on SQL Server, enter:

```
ODBC_URL=<ODBC DSN>
ODBC_USERNAME=<Username>
ODBC_PASSWORD=<Password>
```

2. If you have deployed the Sterling Multi-Channel Selling Solution in WebSphere, then you must edit the classpath setting portion of the script to allow for the fact that the class and library files are in the ***debs_home*/Sterling/ servlets/** directory.
3. Check that the appropriate database client software has been installed on your machine and that it is in your path. For example, if you are using an Oracle server for the Knowledgebase, then make sure that SQLPLUS is in your path. Make sure also that the **tnsnames.ora** file includes an alias as specified as the ORACLE_DATABASE parameter. You should be able to successfully run:

   ```
   tnsping <TNS alias>
   ```
4. Run the createDB target from the SDK.
5. Check the generated log files to verify that the script ran without error.

Populating the Knowledgebase

Having created the Knowledgebase, you need to load data into it, in order to run the Sterling Multi-Channel Selling Solution. Release 8.0 loads data into the Knowledgebase using a set of XML definitions of each data object. Data loading is invoked from the SDK using the loadDB and loadMatrixDB targets: these load the minimal and reference data sets respectively. These targets invoke the XMLLoader scripts as described in "XMLLoader Script" on page 110.

XML Data Format

The data to be loaded using the XMLLoader scripts must be created in the form of XML elements: one for each data object. The form of the XML elements closely matches the structure of the data object: The name of the top-level element is the name of the data object and each child element corresponds to a data field or child data object of the data object.

The top-level element has these attributes:

- A state attribute: set this value to “INSERTED” when you are creating new data. You can use the value “MODIFIED” if you are modifying an existing data object using the XML data loader.
- A type attribute: set this value to “BusinessObject”.

You can use a list data object to act as a container for a list of data objects to be loaded. You must provide a value for each element that is declared as mandatory in the data object definition.

The XMLLoader script essentially sets a classpath and then invokes an XMLLoader class, passing it parameters for the location of the **Comergent.xml** file, the operation (usually “persist”), the partner name (usually “matrix”), and a list of one or more files of data to be loaded.

The files must be in one of two forms:

- Either a set of XML elements: the root element must be named *DataObject*Data. For example:

```
<PromotionData>
    <!--Record 1 ------------------------------------------------>
    <Promotion state="INSERTED" type="BusinessObject">
        <PromotionKey state="INSERTED">126</PromotionKey>
        <PromoCode state="INSERTED">ID 360</PromoCode>
        <PromotionName state="INSERTED">Packages</PromotionName>
        ...
    </Promotion>
    <!--Record 2 ------------------------------------------------>
    <Promotion state="INSERTED" type="BusinessObject">
        <PromotionKey state="INSERTED">127</PromotionKey>
        <PromoCode state="INSERTED">ID 3837</PromoCode>
        ...
    </Promotion>
    ...
</PromotionData>
```

- Or each file can point to a list of files:

```
WEB-INF/xmldata/ProductCategoryList
WEB-INF/xmldata/ProductList
...
```

By convention, the files that provide lists of other files have the suffix “lst”.

XMLLoader Script

The XML data loading script invokes a Java class that uses the Sterling Multi-Channel Selling Solution DataManager to load each object.

The script is called **XMLLoader.bat** (or **XMLLoader.sh** on UNIX systems) and it is located in ***debs_home*/Sterling/WEB-INF/scripts/**. The Sterling Multi-Channel Selling Solution provides two sets of data that can be loaded: see "Data Sets" on page 114 for further information.

Encryption

You can encrypt sensitive data fields so that data stored in the Knowledgebase does not store the data in plain text. Use this mechanism for fields such as user passwords and credit card numbers, but any field can be encrypted provided that its corresponding database column can store strings.

Attention:	You *must* determine which fields are to be encrypted before loading any data into the Knowledgebase. See "Storing Data in Encrypted Form" on page 200 for more information.

When data is encrypted, a special file called **dcmsKey.ser** is created. You must ensure that this file is stored safely. If it is deleted or moved, then the encrypted data cannot be recovered. Note that you cannot export and re-import data that has encrypted fields. The encrypted fields will be garbled if you attempt to do this.

Defining the Knowledgebase as the Data Source

To run the data loading script, you must configure the Sterling Multi-Channel Selling Solution to access the Knowledgebase. You do this using the configuration files **prefs.xml**, and **DataServices.xml**. You must also ensure that the DsKeyGenerators element of the **DataServices.xml** file points to the correct key generator file: **DB2KeyGenerators.xml**, **MsSqlKeyGenerators.xml**, or **OracleKeyGenerators.xml**. Note that if you use the SDK to install the Sterling Multi-Channel Selling Solution, then the correct values will be set up automatically in these files.

Attention:	Please read the database server-specific instructions below. Set the logging level to INFO before running the data loading script.

The following sections provide a brief description of the syntax of the **DataSources.xml** file for the supported database servers.

MsSqlDataSources Syntax

```
<Primary DataService="MsSqlService" SubType="MS"
    ConnectionString="MSSQL_MACHINE"
    UserId="MSSQL_USERNAME" Password="MSSQL_PASSWORD" />
```

- MSSQL_MACHINE is the machine name or IP address of the machine on which SQL Server is running.
- MSSQL_USERNAME and MSSQL_PASSWORD are the username and password used to create the Sterling Multi-Channel Selling Solution

schema. Note that the default database for this user must be the database in which the schema was created.

OracleDataSources Syntax

You can use either the OCI JDBC driver or the Oracle thin client JDBC driver to connect from the Sterling Multi-Channel Selling Solution machine to the Oracle Server. Use:

- For Oracle 8*i*:

```
<Primary DataService="JdbcService" SubType="ORACLE"
    ConnectionString="jdbc:oracle:oci8:@ALIAS"
    UserId="ORACLE_USERNAME" Password="ORACLE_PASSWORD"/>
```

- For Oracle 9*i*:

```
<Primary DataService="JdbcService" SubType="ORACLE"
    ConnectionString="jdbc:oracle:oci:@ALIAS"
    UserId="ORACLE_USERNAME" Password="ORACLE_PASSWORD"/>
```

or

```
<Primary DataService="JdbcService" SubType="ORACLE"
ConnectionString="jdbc:oracle:thin:@ORACLE_MACHINE:ORACLE_PORT:SID"
    UserId="ORACLE_USERNAME" Password="ORACLE_USERNAME" />
```

- ALIAS is the TNSNAMES alias set up on the Sterling Multi-Channel Selling Solution machine.
- ORACLE_USERNAME and ORACLE_PASSWORD are the userid used to create the Sterling Multi-Channel Selling Solution schema.
- ORACLE_MACHINE is the machine name or IP address of the machine on which the Oracle Server is running.
- ORACLE_PORT is the port number at which the Oracle Server listener is listening for connections.
- SID is the Oracle SID of the database.

Internationalization and Support for Locales

Creating Locales

You must edit the file **LocaleDataList** (located in ***debs_home*/Sterling/WEB-INF/ xmldata/**)to specify the locales that you want the Knowledgebase to support. Each installation of the Sterling Multi-Channel Selling Solution can support one or more locales.

Make sure that the server locale (defined by the defaultSystemLocale element of the **Internationalization.xml** file) is included in the list of locales defined in **LocaleDataList**.

See CHAPTER 11, "Internationalization" for more information about creating and adding locales.

Updating Data using XMLLoader

When you load locale-specific data into the Knowledgebase, you can make use of the XMLLoader's ability to modify data objects. In each data object element and child elements, set the state attribute to "Modified". This will update business objects rather than inserting a new business object.

This feature is particularly useful when you are adding locale-specific information to an existing implementation of the Sterling Multi-Channel Selling Solution. For example, the following data object can be used to update a product category business object:

```
<ProductCategory state="MODIFIED">
    <ProductCategoryKey state="MODIFIED">1002</ProductCategoryKey>
    <ParentCategoryKey state="MODIFIED">-1</ParentCategoryKey>
    <SequenceId state="MODIFIED">5</SequenceId>
    <ResourceKey state="MODIFIED">3</ResourceKey>
    <StartDate state="MODIFIED">2000-10-06 17:20:28.0</StartDate>
    <EndDate state="MODIFIED">2100-10-06 17:20:28.0</EndDate>
    <OwnedBy state="MODIFIED">1</OwnedBy>
    <AccessKey state="MODIFIED">2006</AccessKey>
    <ProductCategoryLocale state="MODIFIED">
        <Locale state="MODIFIED">de_DE</Locale>
        <Name state="MODIFIED">Software</Name>
        <Description state="MODIFIED">
            Alle Anwendungspakete, die auf unserer Site vorhanden
sind, werden auf allen unsere Qualitätscomputersysteme geprüft und
bestätigt.
        </Description>
    </ProductCategoryLocale>
</ProductCategory>
```

Database Server-Specific Steps

When running the data loading scripts, the steps vary a little from one database server to another. This section covers the supported database servers.

SQL Server Steps

1. If you are using MS SQL Server as your database server, then make sure that you have copied the **MsSqlJNI.dll** file to the **Winnt\system32\** directory on the Sterling Multi-Channel Selling Solution machine.
2. Edit the **DataServices.xml** file so that all the JdbcDriver elements are commented out. Use the `<!--` and `-->` tags to comment out each element.
3. In **MsSqlDataSources.xml**, make sure that the connection information sets the same UserId and Password as were used to create the schema.

Oracle Steps

1. Edit the **DataServices.xml** file to specify the **OracleDataSources.xml** file and **OracleKeyGenerators.xml** file.
2. Make sure that the JdbcDriver1element takes the value of the name of the Oracle JDBC driver.

Data Sets

The Sterling Multi-Channel Selling Solution provides two sets of XML data objects:

- Reference implementation: this set populates the Knowledgebase with the complete reference implementation (Matrix Solutions) data set. You use this set if you want to deploy our reference implementation in order to familiarize yourself with the Sterling Multi-Channel Selling Solution.
- Minimal implementation: this set populates the Knowledgebase with the minimal data required to get the Sterling Multi-Channel Selling Solution up and running. You use this set when you want to deploy your production system using your data. See "Email Addresses" on page 56 for information about email addresses set in the minimal data set.

Attention:	You must always install the minimal data set: you can optionally layer the reference data on top. Use the SDK loadDB target to load the minimal data only; use the loadMatrixDB target to load both.

To Edit and Run the XML Data Loading Script

1. Configure the **Comergent.xml**, **DataServices.xml**, and appropriate **DataSources.xml** configuration files to point to the database server to be used for the Knowledgebase.

   a. Make sure that the DataServices element of the **Comergent.xml** file points to the correct location of the **DataServices.xml** file.
   b. Make sure that the DsDataSources element of the DataServices.xml file points to the correct location of your **DataSources.xml** file.
   c. Make sure that the appropriate connection information has been entered in the **DataSources.xml** file.

2. If your environment does not have a JAVA_HOME environment variable set, then edit the **XMLLoader.bat** to set the JAVA_HOME environment variable to point to your installation of the JDK. For example:

   ```
   >set JAVA_HOME=C:\JDK1.2.2
   ```

   If you are using Oracle as the database server, then make sure that the classpath is set to include the location of the appropriate JDBC driver class. Typically, you must ensure that the XMLLoader script includes a line of the form:

   ```
       CP=%CP%;%DH%/lib/oracle816_jdbc2.jar
   ```

   or

   ```
       CP=%CP%;%DH%/lib/db2runtime71.jar
   ```

   If you have deployed the Sterling Multi-Channel Selling Solution in WebSphere, then you must edit the classpath setting portion of the script to allow for the fact that the class and library files are in the ***debs_home*/Sterling/servlets/** directory.

3. Save the edited file to ***debs_home*/Sterling/**.
4. Run the XML data loading script from ***debs_home*/Sterling/**.

   - The syntax to load the reference data is:

   ```
   >XMLLoader persist
   ```

   - The syntax to load the minimal data is:

   ```
   >XMLLoader persist minimal
   ```

See CHAPTER 9, "Customizing Your Sterling Multi-Channel Selling Solution" for further information relating to the configuration files. In addition, the *Sterling Multi-Channel Selling Solution Reference Guide* provides a complete description of the Knowledgebase.

Removing Locales

Out of the box, the schema creation scripts and the minimal and reference data sets create data for several locales. Before going live with your implementation, you should remove from the Knowledgebase any locales not supported by your implementation.

To remove a locale, you must remove references to it from the following tables:

- CMGT_ANALYZER_TEXT: for example

  ```
  DELETE FROM CMGT_ANALYZER_TEXT WHERE LOCALE = 'de_DE';
  ```

- CMGT_CURRENCIES: for example

  ```
  DELETE FROM CMGT_CURRENCIES WHERE LOCALE = 'de_DE';
  ```

- CMGT_LOCALE: for example

  ```
  DELETE FROM CMGT_LOCALE WHERE LOCALE_NAME = 'de_DE';
  ```

- CMGT_LOCALE_CURRENCY: for example

  ```
  DELETE FROM CMGT_LOCALE_CURRENCY WHERE LOCALE_NAME = 'de_DE';
  ```

- CMGT_LOCALE_NAMES: for example

  ```
  DELETE FROM CMGT_LOCALE_NAMES WHERE LOCALE_NAME = 'de_DE';
  DELETE FROM CMGT_LOCALE_NAMES WHERE EFFECTIVE_LOCALE = 'de_DE';
  ```

- CMGT_LOOKUPS: for example

  ```
  DELETE FROM CMGT_LOOKUPS WHERE LOCALE = 'de_DE';
  ```

- CMGT_<*OBJECT*>_LOCALE: there are multiple tables that store locale-specific strings for data objects such as products, features, and so on. You must remove the references to the deleted locale from each such table. For example

  ```
  DELETE FROM CMGT__<OBJECT>_LOCALE WHERE LOCALE_NAME = 'de_DE';
  ```

Logging into the Sterling Multi-Channel Selling Solution

Point your browser to the standard login page. The standard URL to access this page is:

```
http://<server>:<port>/Sterling/enterpriseMgr/matrix
```

Irrespective of whether you have generated the reference data set or minimal data set, you can log in as the enterprise administrator whose username and password are “admin” and “admin”.

You can now administer the Sterling Multi-Channel Selling Solution through the standard browser interface. See the *Sterling Multi-Channel Selling Solution Administration Guide* for further details.

Attention:	Before going live with your implementation of the Sterling Multi-Channel Selling Solution, you *must* change the passwords of the admin and ERPAdmin users. Failure to so presents a security breach. Do not use the ERPAdmin user for administration tasks. It is intended only for integration with an ERP system. You should not use this user to log in to the system through the Web. See CHAPTER 19, "Implementing Order Management Integration" for more information.

When you have successfully completed your installation, proceed to the next chapter. Otherwise, use CHAPTER 8, "Troubleshooting and Backing Up the Sterling Multi-Channel Selling Solution" to troubleshoot your installation.

Data Migration

This section describes data migration from earlier releases of the Sterling Multi-Channel Selling Solution. Follow the instructions in each *Sterling Multi-Channel Selling Solution Implementation Guide* for each successive release to upgrade from your current release to the next until you reach the *Sterling Multi-Channel Selling Solution Implementation Guide* for Release 8.0.

General

Before starting any data migration, make sure that you back up your current Knowledgebase database schema and verify that you can recover your existing implementation from this backup.

Attention:	Do not attempt any data migration before backing up your current implementation.

The migration scripts are written to fix the partner key and group key of the enterprise partner as “1”. If you use “1” in your data set for either the partner key or group key of a different partner, then you must manually change this partner by updating the partner and/or group key to an unused value. Note that the group key and partner key are used to reflect relationships such as the assignment of price lists to partners and in ACLs, and these must all be changed for the existing partner before running the migration scripts. These uses include:

- users belong to a partner through the CMGT_USER_CONTACTS.PARTNER_KEY column
- users belong to groups through the CMGT_GROUP_USERS.GROUP_KEY column
- inquiry lists belong to partners through the CMGT_CARTS.PARTNER_KEY column
- price lists are assigned to partners through the CMGT_USER_PRICELIST.PARTNER_KEY column
- ACLs refer to specific groups using the CMGT_ACCESSLIST.GROUP_KEY column

Before running any of the migration scripts, you must identify an enterprise user in your existing implementation whose user key can be used to populate the OWNED_BY columns of the new tables and objects. For example, if you have not deleted the admin user created by the minimal data set, then you can use that user key.

You must replace all occurrences of the string "YOUR_ENTERPRISE_USER" with this value in all of the SQL scripts used in your migration effort. These are:

***debs_home*/Sterling/WEB-INF/sql/Oracle/migration/ DEBS_5.6.2_to_DEBS_6.0/mig_oracle_alter_tables.sql**

***debs_home*/Sterling/WEB-INF/sql/MsSql/migration/ DEBS_5.6.2_to_DEBS_6.0/mig_mssql_alter_tables.sql**

***debs_home*/Sterling/WEB-INF/xmldata/Migration/DEBS_5.6.2_to_DEBS_6.0/ oracle_update_existing_data.sql**

***debs_home*/Sterling/xmldata/Migration/DEBS_5.6.2_to_DEBS_6.0/ ms_sql_update_existing_data.sql**

***debs_home*/Sterling/WEB-INF/sql/Oracle/migration/ DEBS_5.1.3_to_DEBS_5.5/mig_oracle_modify_data.sql**

***debs_home*/Sterling/WEB-INF/sql/MsSql/migration/ DEBS_5.1.3_to_DEBS_5.5/mig_mssql_modify_data.sql**

Releases 5.0 and 5.1.x to Release 5.5

Oracle

1. From ***debs_home*/Sterling/WEB-INF/sql/Oracle/migration/ DEBS_5.1.3_to_DEBS_5.5/**, run **mig_oracle_manage_first.bat**.

2. Check the log file: **mig_oracle_first.log**.
3. Using the XMLImpExp script, add new data as follows:
   a. Copy ***debs_home*/Sterling/WEB-INF/scripts/XMLImpExp.bat** to ***debs_home*/Sterling/**.
   b. Point **OracleDataSources.xml** in ***debs_home*/Sterling/WEB-INF/schema/** to the migrated database.
   c. In a console window, navigate to ***debs_home*/Sterling/** and give following command:

```
XMLImpExp persist ./WEB-INF/xmldata/Migration/
DEBS_5.1.3_to_DEBS_5.5 ACL
XMLImpExp persist ./WEB-INF/xmldata/Migration/
DEBS_5.1.3_to_DEBS_5.5 ModelGroupsAndModelsList
```

4. From ***debs_home*/Sterling/WEB-INF/sql/Oracle/migration/DEBS_5.1.3_to_DEBS_5.5/**, run **mig_oracle_manage_second.bat**.
5. Check the log file: **mig_oracle_second.log**.

Microsoft SQL Server

1. From ***debs_home*/Sterling/WEB-INF/sql/MSSql/migration/DEBS_5.1.3_to_DEBS_5.5**, run **mig_mssql_manage_first.bat**.
2. Check the log files.
3. Using the XMLImpExp script, add new data as follows:
   a. Copy ***debs_home*/Sterling/WEB-INF/scripts/XMLImpExp.bat** to ***debs_home*/Sterling/**.
   b. Point **MsSqlDataSources.xml** in ***debs_home*/Sterling/WEB-INF/schema/** to the migrated database.
   c. In a console window, navigate to ***debs_home*/Sterling/** and run the following commands:

```
XMLImpExp persist ./WEB-INF/xmldata/Migration/
DEBS_5.1.3_to_DEBS_5.5 ACL
XMLImpExp persist ./WEB-INF/xmldata/Migration/
DEBS_5.1.3_to_DEBS_5.5 ModelGroupsAndModelsList
```

4. From ***debs_home*/Sterling/WEB-INF/sql/MSSql/migration/DEBS_5.1.3_to_DEBS_5.5/**, run **mig_mssql_manage_second.bat**.
5. Check the log files.

Release 5.5 to Release 5.5.3

Oracle

1. From ***debs_home*/Comergent/WEB-INF/sql/Oracle/migration/ DEBS_5_5_to_DEBS_5_5_3**, run **migrate.bat**.
2. Check the log file: **migration.log**.

   The section after 'About to execute verify_lookup_columns_values.sql' and before 'About to execute alter_to_not_null.sql' will show you which string values did not get associated lookup_code for them. If any rows are reported, then investigate why the corresponding lookup values do not exist in the corresponding lookup table and either make entries to them or change the string values.

Microsoft SQL Server

1. From ***debs_home*/Comergent/WEB-INF/sql/MsSql/migration/ DEBS_5_5_to_DEBS_5_5_3**, run **mssql_migrate.bat**.
2. Check the log files.

Release 5.5.1 to Release 5.5.3

Using your preferred SQL client, run the following SQL fragment against your Knowledgebase:

```
ALTER TABLE CMGT_LOCALE add DB_LOCALE_LANG_NAME varchar2(50) default
'AMERICAN';
```

Release 5.5.3 to Release 5.6

Oracle

1. From ***debs_home*/Comergent/WEB-INF/sql/Oracle/migration/ DEBS_5_5_3_to_DEBS_5_6**, run **mig_kl.bat**.
2. Check the log file: **mig_kl.log**.
3. Review and make a note of **mig_known_dif.txt**.
4. Using the XMLImpExp script, add new data as follows:
   a. Copy ***debs_home*/Comergent/WEB-INF/scripts/XMLImpExp.bat** (or **XMLImpExp.sh**) to ***debs_home*/Comergent/**.
   b. Point **OracleDataSources.xml** in ***debs_home*/Comergent/WEB-INF/ schema/** to the migrated database.

c. In a console window, navigate to ***debs_home*/Comergent/** and run the following commands:

```
XMLImpExp persist ./WEB-INF/xmldata/Migration/
DEBS_5_5_3_to_DEBS_5_6 ACL
XMLImpExp persist ./WEB-INF/xmldata/Migration/
DEBS_5_5_3_to_DEBS_5_6 LightWeightLookupList
```

Microsoft SQL Server

1. From ***debs_home*/Comergent/WEB-INF/sql/MSSql/migration/ DEBS_5_5_3_to_DEBS_5_6/**, run **mig_kl.bat**.
2. Check the log files.
3. Review and make a note of ***debs_home*/Comergent/WEB-INF/sql/Oracle/ migration/DEBS_5_5_3_to_DEBS_5_6/mig_known_dif.txt**.
4. Using the XMLImpExp script, add new data as follows:
   a. Copy ***debs_home*/Comergent/WEB-INF/scripts/XMLImpExp.bat** to ***debs_home*/Comergent/**.
   b. Point **MsSqlDataSources.xml** in ***debs_home*/Comergent/WEB-INF/ schema/** to the migrated database.
   c. In a command window, go to ***debs_home*/Comergent/** and run the following commands:

```
XMLImpExp persist ./WEB-INF/xmldata/Migration/
DEBS_5_5_3_to_DEBS_5_6 ACL
XMLImpExp persist ./WEB-INF/xmldata/Migration/
DEBS_5_5_3_to_DEBS_5_6 LightWeightLookupList
```

Release 5.6 to Release 5.6.2

Oracle

1. From ***debs_home*/Comergent/WEB-INF/sql/Oracle/migration/ DEBS_5_6_to_DEBS_5_6_2/**, run **migrate.bat**.
2. Check the **migrate.log** file.

Microsoft SQL Server

1. From ***debs_home*/Comergent/WEB-INF/sql/MsSql/migration/ DEBS_5_6_to_DEBS_5_6_2/**, run **migrate.bat**.
2. Check the **migrateAll.log** file.

Release 5.6.2 to Release 6.0

Oracle

1. From ***debs_home*/Comergent/WEB-INF/sql/Oracle/migration/DEBS_5.6.2_to_DEBS_6.0/**, run **mig_oracle_manage.bat**.
2. Check the log file: **mig_oracle.log**.
3. Modify existing data as follows:
   a. Navigate to ***debs_home*/Comergent/WEB-INF/xmldata/Migration/DEBS_5.6.2_to_DEBS_6.0/**.
   b. Edit the script **oracle_update_existing_data.bat** to point to the migrated database and run it.
   c. Check the log file: **oracle_update_existing_data.log**.
4. Using the XMLImpExp script, add new data as follows:
   a. Copy ***debs_home*/Comergent/WEB-INF/scripts/XMLImpExp.bat** to ***debs_home*/Comergent/**.
   b. Point **OracleDataSources.xml** in ***debs_home*/Comergent/WEB-INF/schema/** to the migrated database.
   c. In a command window, navigate to ***debs_home*/Comergent/** and run the following commands:

```
XMLImpExp persist ./WEB-INF/xmldata/-
Migration/DEBS_5.6.2_to_DEBS_6.0 LightWeightLookupList
XMLImpExp persist ./WEB-INF/xmldata/-
Migration/DEBS_5.6.2_to_DEBS_6.0 DSAnalyzerTextList
XMLImpExp persist ./WEB-INF/xmldata/-
Migration/DEBS_5.6.2_to_DEBS_6.0 DSAnalyzerPropsList
XMLImpExp persist ./WEB-INF/xmldata/-
Migration/DEBS_5.6.2_to_DEBS_6.0 ACL
```

   If the cache cleanup cron job does not exist in your implementation, then also run:

```
XMLImpExp persist ./WEB-INF/xmldata/-
Migration/DEBS_5.6.2_to_DEBS_6.0 CronConfigList
```

5. Create new access lists for existing ACLs as follows:
   a. Go to ***debs_home*/Comergent/WEB-INF/sql/Oracle/migration/tools/scripts/**.

b. Run **PopulateAclFile.bat** and respond the questions about the database location, username, password as follows:

```
Choose DBMS Type
-------------------------------------------------
1> Oracle 2> MsSql

Mark you choice number:1
Enter working directory Location : {Directory where the Input file 
is}
That would be by default: %debs_home%\WEB-INF\sql\Oracle\migra-
tion\tools\scripts

Enter Inputfile Name : {Input File Name Acl.xml }
Enter output directory Location :{Directory where you should pro-
duce the output}
Enter database Machine Name : { database machine name }
Enter database user Name : { database user name }
Enter database password :{database password }
Enter database SID : { database sid }
```

This will take the file **Acl.xml** as input and produce **DsAccess** file.

6. Run:

```
debs_home/Comergent/WEB-INF/scripts/XmlImpExp persist inputDirec-
tory DsAccess
```

where the *inputDirectory* parameter is the directory where **DsAccess** is located.

Note: make sure your **DataServices.xml** is pointing to the correct datasources file and that **OracleDataSources.xml** is pointing to the right database.

7. Execute this SQL statement against the database:

```
ALTER TABLE CMGT_SKU_MAPPING MODIFY (OWNED_BY NUMBER(20) NOT 
NULL);
```

Microsoft SQL Server

1. From ***debs_home*****/Comergent/WEB-INF/sql/MsSql/migration/ DEBS_5.6.2_to_DEBS_6.0/**, run **mig_mssql_manage.bat**.
2. Check the log files.
3. Modify the constraints by reading the **how_to_modify_constraint.txt** file and follow instructions from this file to modify the constraints.
4. Modify existing data as follows:

   a. In ***debs_home*/Comergent/WEB-INF/xmldata/migration/DEBS_5.6.2_to_DEBS_6.0/**, point **mssql_update_existing_data.bat** to the migrated database and run it.

   b. Check log file: **mssql_update_existing_data.log**.

5. Using the XMLImpExp script, add new data as follows:

   a. Copy ***debs_home*/Comergent/WEB-INF\scripts XMLImpExp.bat** to ***debs_home*/Comergent/**.

   b. Point **MsSqlDataSources.xml** in ***debs_home*/Comergent/WEB-INF/schema/** to the migrated database.

   c. In a command window, navigate to ***debs_home*/Comergent/** and run the following commands:

```
XMLImpExp persist ./WEB-INF/xmldata/-
Migration/DEBS_5.6.2_to_DEBS_6.0 LightWeightLookupList
XMLImpExp persist ./WEB-INF/xmldata/-
Migration/DEBS_5.6.2_to_DEBS_6.0 DSAnalyzerTextList
XMLImpExp persist ./WEB-INF/xmldata/-
Migration/DEBS_5.6.2_to_DEBS_6.0 DSAnalyzerPropsList
XMLImpExp persist ./WEB-INF/xmldata/-
Migration/DEBS_5.6.2_to_DEBS_6.0 ACL
```

   If the cache cleanup cron job does not exist in your implementation, then also run:

```
XMLImpExp persist ./WEB-INF/xmldata/-
Migration/DEBS_5.6.2_to_DEBS_6.0 CronConfigList
```

6. Create new access lists for existing ACLs as follows:

   a. Go to ***debs_home*/Comergent/WEB-INF/sql/Oracle/migration/tools/scripts/**.

   b. Run **PopulateAclFile.bat** and respond the questions about the database location, username, password as follows:

```
Choose DBMS Type
-------------------------------------------------
1> Oracle 2> MsSql

Mark you choice number:1
Enter working directory Location : {Directory where the Input file
is}
That would be by default: %debs_home%\WEB-INF\sql\Oracle\migra-
tion\tools\scripts

Enter Inputfile Name : {Input File Name Acl.xml }
Enter output directory Location :{Directory where you should pro-
duce the output}
Enter database Machine Name : { database machine name }
Enter database user Name : { database user name }
Enter database password :{database password }
Enter database SID : { database sid }
```

This will take file **Acl.xml** as input and product DsAccess file.

7. Run:

   ***debs_home*/Comergent/**`WEB-INF/scripts/XmlImpExp persist` *`inputDirectory`* `DsAccess`

   where the *inputDirectory* parameter is the directory where DsAccess is stored.

8. Execute this SQL statement against the database:

   ```
   ALTER TABLE CMGT_SKU_MAPPING ALTER COLUMN OWNED_BY NUMERIC(20) NOT
   NULL
   GO
   ```

Migrating Orders Data

You must also migrate your orders data from the CMGT_ORDERS table to the new CMGT_OIL and CMGT_ORDER_EXTN tables as follows:

1. Run:

```
INSERT INTO CMGT_OIL (CART_KEY,OWNED_BY, ACCESS_KEY,
ACTIVE_FLAG,DELIVERY_DATE, REQUESTED_DATE, SHIPPED_DATE,
TOTAL_NUMBER_ITEMS, CURRENCY_CODE,CURRENCY_KEY, CONTACT_REF_NUMBER,
SHIP_COMPLETE, TAXABLE, PAYMENT_TYPE,CREDIT_CARD_TYPE,
CREDIT_CARD_HOLDER, PAYMENT_EXPIRATION_DATE, PAYMENT_NUMBER,
SHIPPING_METHOD_CODE, SHIPPED_STATUS,SHIP_TYPE_FLAG, LAST_NAME,
FIRST_NAME, EMAIL_ADDRESS, PHONE_NUMBER, ENCRYPTION_SEED,
ERP_MESSAGE, VERTICAL_KEY,MEMO, CART_NAME, CART_STATUS_CODE,
PARTNER_KEY,CONTACT_KEY,TOTAL, ROUTE_STATUS_CODE, ROUTE_USER_KEY,
ROUTE_NOTES, ROUTE_FROM_USER_KEY, ROUTE_DATE,MEMBER_LEVEL,
CUSTOMER_TYPE_CODE, SELLER_KEY, PO_NUMBER, OIL_TYPE,
UPDATE_DATE,UPDATED_BY,CREATION_DATE,CREATED_BY)
SELECT CART_KEY, OWNED_BY, ACCESS_KEY, ACTIVE_FLAG, DELIVERY_DATE,
REQUESTED_DATE, SHIPPED_DATE,TOTAL_NUMBER_ITEMS, CURRENCY_CODE,
CURRENCY_KEY, CONTACT_REF_NUMBER, SHIP_COMPLETE, TAXABLE,
PAYMENT_TYPE, CREDIT_CARD_TYPE, CREDIT_CARD_HOLDER,
PAYMENT_EXPIRATION_DATE, PAYMENT_NUMBER, SHIPPING_METHOD_CODE,
SHIPPED_STATUS, SHIP_TYPE_FLAG,LAST_NAME, FIRST_NAME, EMAIL_ADDRESS,
PHONE_NUMBER, ENCRYPTION_SEED, ERP_MESSAGE, VERTICAL_KEY,MEMO,
CART_NAME, CART_STATUS_CODE, PARTNER_KEY, CONTACT_KEY,TOTAL,
ROUTE_STATUS_CODE, ROUTE_USER_KEY, ROUTE_NOTES, ROUTE_FROM_USER_KEY,
ROUTE_DATE, MEMBER_LEVEL,CUSTOMER_TYPE_CODE, 1, PO_NUMBER, 2,
```

```
UPDATE_DATE, UPDATED_BY, CREATION_DATE, CREATED_BY FROM CMGT_ORDERS;
```

2. Run:

```
INSERT INTO CMGT_ORDER_EXTN (CART_KEY, ORDER_REF_NUMBER,
INTEGRATION_STATUS, ORDER_DATE, ORDER_KEY, ORDER_STATUS, PO_NUMBER,
PARTNER_LEVEL_CODE, PARTNER_REF_NUMBER, SHIPPING_CHARGES, TAX,
TOTAL_AMOUNT, ACTIVE_FLAG)
SELECT CART_KEY, ORDER_REF_NUMBER, INTEGRATION_STATUS, ORDER_DATE,
ORDER_KEY, ORDER_STATUS, PO_NUMBER, PARTNER_LEVEL_CODE,
PARTNER_REF_NUMBER, SHIPPING_CHARGES, TAX, TOTAL_AMOUNT, ACTIVE_FLAG
FROM CMGT_ORDERS WHERE ORDER_STATUS != 20;
```

3. Run:

```
INSERT INTO CMGT_OIL_LI (CART_LINE_KEY, CART_KEY, DELIVERY_DATE, SKU,
DESCRIPTION, SKU_AUTHORITY, TOTAL_AMOUNT, SHIPPING_METHOD_CODE,
SHIPPING_CHARGES, SHIPPED_STATUS, QUANTITY, UNIT_OF_MEASURE_CODE,
CONFIG_FLAG, PARENT_KEY, CONFIG_CONTAINER, LIST_PRICE, MEMO,
ACTIVE_FLAG, CONFIG_ID, REQUESTED_DATE, SHIP_COMPLETE, RETURN_CODE,
RETURN_STATUS, RETURN_REASON, SPECIAL_INSTRUCTIONS_FLAG,
CONFIGURATION_KEY, BUYER_SKU_AUTHORITY, UPDATE_DATE, UPDATED_BY,
CREATION_DATE, CREATED_BY)
SELECT CART_LINE_KEY, CART_KEY, DELIVERY_DATE, SKU, DESCRIPTION,
SKU_AUTHORITY, TOTAL_AMOUNT, SHIPPING_METHOD_CODE, SHIPPING_CHARGES,
SHIPPED_STATUS, QUANTITY, UNIT_OF_MEASURE_CODE, CONFIG_FLAG,
PARENT_KEY, CONFIG_CONTAINER, LIST_PRICE,MEMO, ACTIVE_FLAG,
CONFIG_ID, REQUESTED_DATE, SHIP_COMPLETE, RETURN_CODE, RETURN_STATUS,
RETURN_REASON, SPECIAL_INSTRUCTIONS_FLAG, CONFIGURATION_KEY,
BUYER_SKU_AUTHORITY, UPDATE_DATE, UPDATED_BY, CREATION_DATE,
CREATED_BY FROM CMGT_ORDER_LINE_ITEMS;
```

4. Run:

```
INSERT INTO CMGT_ORDER_LI_EXTN (CART_LINE_KEY, QUANTITY_SHIPPED,
QUANTITY_RETURNED, SHIPPED_DATE, RETURN_REQUEST_DATE,
RETURN_APPROVAL_DATE, SERIALIZABLE_FLAG, ORDER_STATUS, ACTIVE_FLAG)
SELECT CART_LINE_KEY, QUANTITY_SHIPPED, QUANTITY_RETURNED,
SHIPPED_DATE, RETURN_REQUEST_DATE, RETURN_APPROVAL_DATE,
SERIALIZABLE_FLAG, ORDER_STATUS, ACTIVE_FLAG FROM
CMGT_ORDER_LINE_ITEMS WHERE ORDER_STATUS != 20;
```

5. Run:

```
UPDATE CMGT_OIL SET OIL_TYPE=100 WHERE CART_KEY NOT IN (SELECT
CART_KEY FROM CMGT_ORDER_EXTN WHERE CMGT_OIL.CART_KEY =
CMGT_ORDER_EXTN.CART_KEY);
```

CHAPTER 7

Customer Segmentation

This chapter describes how to customize the customer segmentation feature of the Sterling Multi-Channel Selling Solution.

Overview

The segmentation feature of the Sterling Multi-Channel Selling Solution groups users into segments based on the criteria specified by a segment administrator. A marketing manager can then direct business activities such as promotions and pricing rules to users in a specific segment.

Various types of criteria can define a segment. Such criteria include information from user/organization profiles, user-defined attributes, and even membership in other segments. See the *Sterling Multi-Channel Selling Solution Administration Guide* for more information about segments.

Customization

BHC

The Sterling Multi-Channel Selling Solution ships with several Behavioral/ Historical Calculations (BHC) for use in segmentation. You can create your own custom BHC by creating a backend processing unit. You can then make your BHC available in the user interface for use by a segment administrator by modifying the BHC Registry.

Tracking Events

You can create a tracking event to track user activities and then use the stored tracking event data in a BHC.

Profile Data

Several fields in the user/organization profiles are available by default for use in segmentation. You can make other fields in the user/organization profiles available by editing the Entity Field Registry.

Behavioral/Historical Calculation (BHC)

One special type of segment is the Behavioral/Historical Segment (BHS), which uses a Behavioral/Historical Calculation (BHC) as a criterion.

A BHC is a criterion based on information about the shopping and browsing behavior of users and organizations. Examples of the kind of information that can be specified by a BHC: Total amount ordered by a user; Number of times a user has abandoned a specific item in a shopping cart; Number of times a user has browsed a specific category.

The Sterling Multi-Channel Selling Solution transfers this behavioral/historical data from the transactional database (Knowledgebase) to the segmentation database and calculates membership in a BHS/BHC according to the schedule specified by an administrator.

A segment administrator who wants to use a BHC as a criterion in a “regular segment” must first create a BHS that contains the BHC as a criterion and then include the BHS in the regular segment.

Custom BHCs

The Sterling Multi-Channel Selling Solution includes several BHCs as part of the standard deployment. A developer can create custom BHCs as well.

A Behavioral/Historical Calculation (BHC) is a backend processing unit. The BHCs that ship with Sterling Multi-Channel Selling Solution are stored procedures that read from and write to the segmentation database.

BHC Registry and the Segment Management UI

The BHCRegistry.xml file specifies the BHCs available to segment administrators in the Segment Management user interface of the Sterling Multi-Channel Selling Solution. An entry for a BHC in BHCRegistry.xml includes the segment group, name and description, input parameters, user interface controls, and the name of the backend processing unit.

BHC Registry

A segment administrator who wants to create a Behavioral/Historical Segment (BHS) must select a Behavior Group and Behavioral Criterion from the “New Behavioral Segment” user interface. Once the BHC has been selected and loaded, the interface will display form elements for data input. The backend processing unit for the BHC will use this input data to create the segment.

The groups, criteria, and inputs available to the segment administrator are specified by the BHC Registry file located at **WEB-INF/properties/ SegmentationBHCRegistry.xml**.

The BHC Registry also specifies the backend processing unit that processes the inputs and creates the segment.

Furthermore, the BHC Registry specifies which business rules (if any) a BHC requires. The segmentation interface will indicate if a business rule required by a BHC is not turned on.

Sample BHC Registry

The following is a sample SegmentationBHCRegistry.xml file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Registry ID="BHCRegistry" resourceBundle="com.comergent.refer-
ence.apps.customerSegmentation.SegmentationBHCRegistry">
    <Category ID="userOrderItemPC"
name="User_ordered_items_in_category">
       <BHC ID="userAmountItemPC" name="Amount_of_items_in_category"
backendProcess="SP_USER_CAT_ORD_AMT" dependingOn="">
            <Input ID="pc" name="Product_category" dataType="Hierar-
chy" lookupType="ProductCategory" inputID="0">
                <Display type="singleSelectProductCategoryPicker"/>
            </Input>
            <Input ID="amount" name="Amount" dataType="Double"
inputID="1"/>
           <Input ID="currencyCode" name="Currency" dataType="Lookup"
inputID="2">
                <Display type="currencySelector"/>
            </Input>
            <Input ID="timeFrame" name="Time_frame" dataType="Time-
Frame" inputID="3">
                <Display type="timeFrame"/>
            </Input>
        </BHC>
        <BHC ID="userNumberItemPC"
name="Number_of_times_items_ordered_in_category" backendPro-
cess="SP_USER_CAT_ORD_NUM" dependingOn="">
            <Input ID="pc" name="Product_category" dataType="Hierar-
chy" lookupType="ProductCategory" inputID="0">
                <Display type="singleSelectProductCategoryPicker"/>
            </Input>
            <Input ID="numberOfTimes" name="Number_of_times"
dataType="Long" inputID="1"/>
            <Input ID="timeFrame" name="Time_frame" dataType="Time-
Frame" inputID="2">
                <Display type="timeFrame"/>
            </Input>
        </BHC>
        <BHC ID="userQtyItemPC"
name="Quantity_of_items_ordered_in_category" backendPro-
cess="SP_USER_CAT_ORD_QTY" dependingOn="">
            <Input ID="pc" name="Product_category" dataType="Hierar-
chy" lookupType="ProductCategory" inputID="0">
                <Display type="singleSelectProductCategoryPicker"/>
            </Input>
            <Input ID="qty" name="Quantity" dataType="Long"
inputID="1"/>
            <Input ID="timeFrame" name="Time_frame" dataType="Time-
Frame" inputID="2">
                <Display type="timeFrame"/>
            </Input>
        </BHC>
    </Category>
</Registry>
```

BHC Registry Schema

The following is an explanation of the elements and attributes in the BHC Registry:

- **Registry:** Root element of the BHC Registry.
    - **ID:** Value must equal “BHCRegistry”.
    - **resourceBundle:** Resource bundle which indicates the localized strings displayed on the interface. To set the text displayed for a BHC, edit src/com/comergent/reference/apps/customerSegmentation/SegmentationBHCRegistry.properties.

- **Category:** Behavior group. Categories will be available as drop-down menus in the segment management interface. Once a category/group is selected, the user will be able to chose a BHC in the selected category.
    - **ID:** Unique identifier for the category.
    - **name:** Name of the category. This name is mapped to a string in the resource bundle to determine the text that the segmentation management interface displays in the drop-down menu.
- **BHC:** Behavioral/Historical Criterion.
    - **ID:** Unique identifier for the BHC.
    - **name:** Name of the BHC. This name is mapped to a string in the resource bundle to determine the text that the segmentation management interface displays in the drop-down menu.
    - **backendProcess:** The backend processing unit is the stored procedure that the engine invokes to process the inputs and create the criteria.
    - **dependingOn:** Business rule required for the BHC. If the required business rule is not turned on, this will be indicated in the segmentation user interface. If no business rule is required this value should be blank.
- **Input:** Input information required by the BHC. The segmentation administration interface displays form controls for the input.
    - **ID:** Unique identifier for the input.
    - **name:** Name of the input. The segmentation administration interface displays this name.
    - **dataType:** Data type of the input. The dataType attribute also determines the kind of input control that the interface displays.
    - **lookupType:** Further information about the data type of the input.
    - **inputID:** Indicates a sequence for the inputs in a BHC. The user interface will display the inputs with lower inputID numbers first.
- **Display:** Optional child node of Input that specifies an input control.
    - **type:** The input control. This usually indicates a type of picker or selector.

Modification of the BHC Registry

You may add new categories and BHCs to the BHC Registry, but you should not *modify* existing BHC entries as this could affect the logic of existing segmentation criteria.

Whenever the BHC Registry is modified, the server must be restarted before the changes will be applied.

Interface Controls for Inputs

The type of user interface control that appears for the inputs depends on the *dataType* attribute of the *Input* element. If, however, an *Input* element has a child *Display* element, then the input control will be determined by the *type* attribute of the *Display* element.

The following table describes the types of interface controls specified in the *type* attribute of the *Display* element:

TABLE 11. Description of Interface Controls for the Display Element

Display Type	Description
currencySelector	Currency type
multiSelectFeaturePicker	Product features. Multiple features may be chosen.
singleSelectProductCategoryPicker	Category node
singleSelectProductPicker	Individual product
timeFrame	Date range

BHC Stored Procedures

The backend processing unit that builds a BHC is a stored procedure that accesses the tables of the segmentation database and then writes the segment data to a segment table. See Chapter 9 "Segmentation Database" in the *Sterling Multi-Channel Selling Solution Reference Guide* for details about the tables in the segmentation database.

Several BHC stored procedures ship with the Sterling Multi-Channel Selling Solution. Developers who wish to create their own custom BHCs can use these stored procedures (such as SP_USER_CAT_ORD_QTY or SP_USER_ITEM_BROWS_NUM) as a model when developing their own. You can find the segmentation stored procedures script at **WEB-INF/sql/*DBType*/ segment/*dbtype*_stored_procs.sql**.

The following are general steps that will likely form the pattern of a backend processing unit:

1. Read values from the CMGT_SEGMENT_LINE_BHC and CMGT_SEGMENT_LINE_BHC_LIST tables in the segmentation database. These tables contain the conditions the segment administrator set in the segment administration interface.
2. Use the conditions read from the tables to create your query.
3. Use your query to extract the segment users from the segmentation database. This will be run against tables containing behavioral/historical information about users, such as CMGT_SEGMENT_ORDER, CMGT_SEGMENT_ORDER_LINE, and the event tracking tables.
4. Write the segment users to the CMGT_SEGMENT_BUILD table.

Note	The BHC stored procedures that ship with the Sterling Multi-Channel Selling Solution first write results to a temporary table, such as CMGT_SEGMENT_CAT_ORD_QTY_TMP, and then compares the users in that table to the users in CMGT_USER_CONTACTS before writing to the CMGT_SEGMENT_BUILD. All tables in the Segmentation DB whose name ends with _TMP are these temporary segment results tables.

Activity Tracking Events

One useful way to employ BHCs is to take advantage of user data generated by tracking events. Tracking events track such user activities as browsing and searching. The application sends tracking event data to the Tracking Service module; the Tracking Service then stores this data in the segmentation database. A BHC backend processing unit can then access this data to create a user segment.

The Sterling Multi-Channel Selling Solution ships with several tracking events available. In addition, you can create your own tracking events for use in segmentation.

The following sections describe how to set up a tracking event and how to read the event data from the segmentation database for use by a BHC.

Set up a Tracking Event

To create a tracking event, you must create a class that implements the ITrackingEvent interface, which is part of the com.comergent.api.appservices.trackingService package. (The trackingService package also contains the AbstractTrackingEvent class, which implements ITrackingEvent. You may choose to subclass AbstractTrackingEvent.)

The ITrackingEvent interface includes the method Long getEventType(), which must return a number identifying this event. The segmentation engine uses this number to distinguish this type of event from others being used. The numbers 100, 200, and 300 already are used by tracking events that ship with the Sterling Multi-Channel Selling Solution. Be careful not to use those number or any others that may already be used by tracking events to identify their event types.

The tracking event must be passed to the Tracking Service using the ITrackingService method publishTrackingEvent(ITrackingEvent event).

You can create a tracking service object by using TrackingServiceFactory.getService().

The following lines of code show an event being passed to the Tracking Service:

```
//trackingEvent is an object that implements ITrackingEvent
ITrackingService trackingService = TrackingServiceFactory.getSer-
vice();
trackingService.publishTrackingEvent(trackingEvent);
```

Event specific data are stored in name, value pairs. Currently only one event specific data can be set, with the name set as “EVENT_DATA”.

The following tracking events are shipped with the Sterling Multi-Channel Selling Solution:

Event	Type Number	Class
Category Browse Event	100	CatalogCategoryBrowseEvent.java
Product Detail View	200	CatalogProductDetailViewEvent.java
Catalog Search	300	CatalogSearchEvent.java

Note A tracking event class is not required follow the pattern of Sterling Multi-Channel Selling Solution events that use the EventBus mechanism. (See the *Sterling Multi-Channel Selling Solution Developer Guide*, Chapter 9 “Events.”) The tracking events that ship with the Sterling Multi-Channel Selling Solution use the EventBus mechanism, but this is not required for tracking events.

Tracking Service Configuration

The Tracking Service receives tracking event objects and stores the data in tables in the segmentation database.

The Tracking Service module is automatically installed with the Sterling Multi-Channel Selling Solution. Therefore, in a clustered environment each node will have its own Tracking Service with its own independent in-memory queues.

There are, however, some configuration properties that you can set. These properties are located in the **TrackingServiceConfig.xml** file located in the **WEB-INF/properties** folder.

The available properties include:

- **maxSession:** The number of sessions that are supported. A negative value indicates an unlimited number of sessions are supported.
- **maxEventPerSession:** Number of events per session that are supported. A negative value indicates an unlimited number of events per session are supported.

The default value of both properties is -1.

The Tracking Service is initialized in **WEB-INF/properties/init.xml** folder with the following code:

```
<module name="TrackingService" initClass="com.comergent.appser-
vices.trackingService.TrackingServiceInitHelper">
            <config name="TrackingService">/WEB-INF/properties/
TrackingServiceConfig.xml</config>
</module>
```

The installation automatically includes this code in the init.xml file, so you will not need to add it. But if issues arise with the Tracking Service, you may want to ensure that his code is in fact included.

Tracking Event Tables in the Segmentation Database

The Tracking Service stores data about user activity in the CMGT_SEGMENT_TRACKING_EVENT staging tables. The segmentation engine then uses the stored procedure SP_EVENT_DATA_TRANSFER to permanently store the data from each type of tracking event to its own table. When working with a custom tracking event, you must create a table in the segmentation database for the permanent storage of the event data, and modify the SP_EVENT_DATA_TRANSFER stored procedure to facilitate transfer of the data from the CMGT_SEGMENT_TRACKING_EVENT staging tables to the permanent storage table.

You can examine the permanent storage tables for the tracking events that ship with the Sterling Multi-Channel Selling Solution:

- CMGT_SEGMENT_EVENT_CAT for the Category Browse event.

- CMGT_SEGMENT_EVENT_PROD for the Product Detail View event.
- CMGT_SEGMENT_EVENT_SEARCH for the Search event.

At a minimum the table for a tracking event should have columns for storing the USER_KEY, EVENT_DATE, STOREFRONT_KEY, and EVENT_DATA from the CMGT_SEGMENT_TRACKING_EVENT tables.

The segmentation engine uses the stored procedure SP_EVENT_DATA_TRANSFER to transfer event data from the staging to the permanent storage tables. The mapping between the two tables is performed based on the EVENT_TYPE number.

For example the stored procedure for Oracle contains the line:

```
V_SQL_STATEMENT1 := 'INSERT INTO CMGT_SEGMENT_EVENT_CAT SELECT
USER_KEY, EVENT_DATE, STOREFRONT_KEY, EVENT_DATA FROM ' ||
V_EVENT_TABLE_NAME || ' WHERE EVENT_TYPE = 100';
```

This statement transfers all events with an EVENT_TYPE of 100 (Category Browse) from the staging table to the CMGT_SEGMENT_EVENT_CAT table.

For a custom event you will add a similar statement, indicating the EVENT_TYPE number and storage table for your tracking event and a line executing that event.

Note	The transfer of custom tracking event data should be done by modifying the SP_EVENT_DATA_TRANSFER stored procedure and not by creating a new stored procedure.

Once the tracking event permanent storage tables are available they can by used by a BHC backend processing unit for creating a BHC segment.

Entity Field Registry

The Entity Field Registry (SegmentationEntityFieldRegistry.xml) makes fields from the user and organization profiles available in the user interface for use in building segments. The default implementation of the Sterling Multi-Channel Selling Solution makes several fields in the user/organization profile available for use in segmentation. You can make more profile fields available by adding them to the SegmentationEntityFieldRegistry.xml file.

The Entity Registry file is located at **WEB-INF/properties/ SegmentationEntityFieldRegistry.xml**.

Sample Entity Field Registry

The following is a sample SegmentationBHCRegistry.xml file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Registry ID="EntityFieldRegistry" resourceBundle="com.comergent.ref-
erence.apps.customerSegmentation.SegmentationEntityFieldCriteriaLa-
bels">

    <Category ID="userCriteria" name="User_criteria">

        <Entity ID="jobTitle" name="Job_title_entity" dataOb-
ject="UserContact" keyField="UserKey">
            <Input ID="JobTitle" name="Job_title_input"
dataType="String" inputID="0"/>
        </Entity>
        <Entity ID="deptName" name="Department_name_entity" dataOb-
ject="UserContactDepartment" keyField="UserKey">
            <Input ID="DeptName" name="Department_name_field"
dataType="String" inputID="0"/>
        </Entity>

    </Category>

    <Category ID="partnerCriteria" name="Partner_criteria">

        <Entity ID="partnerType" name="Partner_type_entity" dataOb-
ject="Partner" keyField="PartnerKey">
            <Input ID="PartnerTypeCode" name="Partner_type_field"
dataType="Lookup" lookupType="PartnerType" inputID="0"/>
        </Entity>
        <Entity ID="partnerLevel" name="Partner_level_entity" dataOb-
ject="Partner" keyField="PartnerKey">
            <Input ID="PartnerLevelCode" name="Partner_level_field"
dataType="Lookup" lookupType="PartnerLevel" inputID="0"/>
        </Entity>
            </Category>

</Registry>
```

Entity Field Registry Schema

The following is an explanation of the elements and attributes in the BHC Registry schema:

- **Registry:** Root element.
  - **ID:** Must be set to "EntityFieldRegistry".

    - **resourceBundle:** Resource bundle that determines the label for the fields that appear in the user interface. To set the string displayed for an entity field, edit the src/com/comergent/reference/apps/customerSegmentation/SegmentationEntityFieldCriteriaLabels.properties file.

- **Category:** Groups entities as being part of either the user profile or partner profile.
    - **ID:** Must be either "userCriteria" or "partnerCriteria".
    - **name:** Must be either "User_criteria" or "Partner_criteria"
- **Entity:** Available field from a profile.
    - **ID:** Unique indentifier. This ID is provided by the registry author; it does not map to anything else.
    - **name:** Name for the entity. This name is mapped to a string in the resource bundle to determine the text that the segmentation management interface displays in the Entity Name drop-down menu.
    - **dataObject:** Data object associated with the entity.
    - **keyField:** Key field in the data object associated with the entity.
- **Input:** Input information required. The segmentation administration interface displays form controls for the input.
    - **ID:** Corresponds to the "name" attribute of the "DataField" element in the data object.
    - **name:** Name for the entity. This name is mapped to a string in the resource bundle to determine the text label that the segmentation management interface displays at the input line.
    - **dataType:** Data type of the input.
    - **lookupType:** Used only with the lookup data type. Specifies a type in the lookup table.
    - **lookupResultMethod:** One of the methods for returning data from the lookup table provided by the com.comergent.api.apps.appUtils.util.AppsLookupHelper object. This is an alternative to using the "lookupType" attribute to specify a lookup type. Some lookup types may be available only through the methods of the AppsLookupHelper object.

- **operator:** Used with the Boolean data type; generally set to "EQUALS". Used when a second input needs to clarify the main input. For example, a "CountryCode" input might be followed by a "DefaultShipTo" input, which would differentiate it from a "DefaultSoldTo" address. These type of inputs will be hidden and not appear in the user interface.
- **value:** Used with the Boolean data type; generally set to 1.
- **inputID:** Indicates a sequence for the inputs. The user interface will display the inputs with lower inputID numbers first.

Specifying Fixed Data Inputs

Sometimes the data field specified by the ID of an input needs to be distinguished from other uses of that field. For example, a CountryCode may be used as either part of a DefaultShipTo address or a DefaultSoldTo address. In such a circumstance, a second input element should be added to the registry. The *ID* will indicate the designated field (for example, DefaultShipTo); the *dataType* will be Boolean, the *operator* will be EQUALS and the *value* will be 1. This will send fixed read-only data along with the value specified by the first input. The input for this fixed data will be hidden and not appear in the user interface.

For example:

```
 <Entity ID="defaultBillToPostalCode"
name="Default_bill_to_postal_code_entity" dataObject="Address" key-
Field="PartnerKey">
            <Input ID="PostalCode" name="Postal_code_field"
dataType="String" inputID="0"/>
            <Input ID="DefaultBillTo" name="Default_bill_to_field"
dataType="Boolean" operator="EQUALS" value="1" inputID="1"/>
        </Entity>
```

CHAPTER 8

Troubleshooting and Backing Up the Sterling Multi-Channel Selling Solution

Troubleshooting

Testing with the Administration URL

You can make sure that the various parts of the installation are functioning by pointing your browser to the URL used to access the administration pages:

```
http://<server>:<port>/Sterling/en/US/enterpriseMgr/matrix
```

Email Server

You must make sure that the SMTP Mail Server used to send email from the Sterling Multi-Channel Selling Solution is up and running. Make sure that you can ping the SMTP Mail Server from the Sterling Multi-Channel Selling Solution machine using the machine name specified in the SMTPHost element of the **Comergent.xml** file. Storefront administrators can configure the SMTP host by setting the SMTP Host Machine system property to the appropriate value. To set the SMTP Host Machine system property, navigate from the System Administration panel of the home page to System Services, then the Commerce Manager category, then the SMTP category, and enter the appropriate value in the SMTP Host Machine field.

Certain UTF-8 characters may not display well in the subject lines of email sent from the Sterling Multi-Channel Selling Solution to users. This is due to email clients that are not configured to display UTF-8 characters correctly. If the problem persists, review the characters being used in the subject lines of the email and provide information to users about suitable email clients.

General Troubleshooting Tips

This section includes general diagnosis approaches that can help to quickly pinpoint the source of your problem.

Tomcat Server

The following considerations apply to running Apache Tomcat:

- SESSIONS.ser: when Tomcat is shut down, the server saves the current session information to a file called ***container_home*/work/Standalone/localhost/Sterling/SESSIONS.ser.** You should delete copies of this file before restarting Tomcat.
- By default, Tomcat does not recompile JSP pages if it determines that its compiled version has a timestamp newer than the corresponding JSP page. If you see errors relating to MethodNotFound exceptions, then the likely cause is an old compiled page. You can solve this problem by deleting the ***container_home*/work/Standalone/localhost/Sterling/** directory to force re-compilation of all the JSP pages.

Common Problems

This section covers problems that commonly occur during startup and runtime. You can use the messageTypeValidate element to validate all the message types as the system starts. The element is set to TRUE by default. You should set this element to FALSE once the system has passed its acceptance tests.

Errors at Startup Time

When the Sterling Multi-Channel Selling Solution is started by the servlet container, it logs its progress through initialization. Look for these errors in the console window or event log:

TABLE 12. Startup Errors

Error	Cause and Solution
`InputStream: 24, 384: "</Comergent>" expected.`	A syntax error in one of the configuration files has caused the DataManager to fail to initialize. You must correct the syntax error. The InputStream line provides the exact location of the error.
`java.io.FileNotFoundException: C:\jakarta-tomcat\webapps\Sterling\WEB-INF\properties\Comergent.xml (The system cannot find the file specified)`	The main **Comergent.xml** file is not in the correct location as specified by the propertiesFile element of the **web.xml** file.
`Env/main:W1:DATASERVICES Primary Connection Failed: Io exception: Connection refused(DESCRIPTION =(TMP=)(VSN-NUM=135290880)(ERR=12505)(ERROR_STACK=(ERROR =(CODE=12505)(EMFI=4))))`	The server has failed to connect to the database server. Perform the following checks: Ping the database server machine; there may be a network failure. Enter the IP address of the machine to see whether the host name can be resolved. If you can ping the database server machine, then check that the database connection information that you have entered in the **DataSources.xml** file is correct. If possible, use an alternate database connection method (such as SQLPLUS or SQL Server's Enterprise Manager). If this fails, then either the database is down or you have incorrect connection information.
`Primary Connection Failed: No suitable driver`	The server has failed to find a valid Driver class in its classpath. Check that any JDBC driver specified in the **DataServices.xml** file lies in one of the classpath directories or archive files. In particular, check that an appropriate JDBC driver has been specified to connect to the database server: For Oracle Servers, you may use oracle.jdbc.odbc.OracleDriver.
`Cannot find file=web-inf/HostedPartner.xml`	The initialization servlet has failed to find one of the properties files referred to in the main **Comergent.xml** file. Check the names and paths to the properties files. In a UNIX installation, check for case-sensitive file names.

TABLE 12. Startup Errors (Continued)

Error	Cause and Solution
`Failed to create a NameKeyTable. com.comergent.dcms.util.ICCException: [CMGT_E_SCHEMA_KEY_GEN_NOT_FOUND] error: " Schema Error - DataObject: Promotion ExternalObject: PromotionControl specifies KeyGenerator: ControlKey which does not exist."`	A problem lies in your definition of the XML schema. On startup, the Sterling Multi-Channel Selling Solution reads the schema files and attempts to load the schema as an internal data structure. Exceptions are most commonly thrown when the definition of an element is omitted from the schema. In this example, the Sterling Multi-Channel Selling Solution has read the **DsRecipes.xml** file, and attempted to load the Promotion DataObject. This DataObject includes in its definition file, **Promotion.xml**, a reference to a KeyGenerator element called "ControlKey". Inspection of the DsKeyGenerators file shows that no KeyGenerator element called ControlKey is declared.
`java.net.BindException: Address in use`	A process is already bound to one of the ports that the Sterling Multi-Channel Selling Solution is attempting to use. You must either stop the existing process that is using the port or use a different port.
`com.comergent.dcm.util.ICCException: [CMGT_E_UNKNOWN_ELEMENT] error: "Element: Comment of DataObject: Comment is not in the DCMS schema"`	An error has occurred while the DataManager initializes the schema. Check the definition of business and data objects. Check that a header DsElement has been declared for the data object and that DsElements are declared for each data element.
`Comergent Init Servlet: DataManager NOT initialized com.comergent.dcm.util.ICCException: [CMGT_E_SCHEMA_KEY_GEN_NOT_FOUND] error: "Schema Error - DataObject: OrderAddress ExternalObject: OrderAddress specifies KeyGenerator: OrderAddressKey which does not exist."`	You have declared a KeyGenerator in the data object definition, but it is not defined in the **KeyGenerators.xml** file. Make sure that you have modified the correct **KeyGenerators.xml** file: this is usually **OracleKeyGenerators.xml** or **MsSqlKeyGenerators.xml**. Also make sure that your **DataServices.xml** file points to the correct **KeyGenerators.xml** file.

TABLE 12. Startup Errors (Continued)

Error	Cause and Solution

`2002.10.22 13:33:03:459 Env/Thread-2:ER:DATASERVICES JDBCService.restore Error: ORA-00600: internal error code, arguments: [ttcgcshnd-1], [0], [], [], [], [], [], [] java.sql.SQLException: ORA-00600: internal error code, arguments: [ttcgcshnd-1], [0], [], [], [], [], [], []`
`at oracle.jdbc.dbaccess.DBError.throwSqlException(DBError.java:168)`
`at oracle.jdbc.ttc7.TTIoer.processError(TTIoer.java:208)`
`...`
	There is a mismatch between the Oracle database version (usually 8i or 9i) and the JDBC driver that is being used to connect from the Sterling Multi-Channel Selling Solution. Check the classpath that the servlet container is using and remove any references to JDBC driver JAR files that come before the **oraclejdbc.jar** file in ***debs_home*****/Sterling/ WEB-INF/**.

`Env/main:ER:MSGT Illegal Unresolved Reference to a MessageType: contentFrame`	A MessageTypeRef element references a message type definition that does not exist.
`Env/main:ER:MSGT com.comergent.api.dcm.messageType.MessageTypeInstantiationException: Failed instantiating or locating com.comergent.apps.partnerMkt.blc.MissingBLC in MessageType GenericLoginDisplay`	A message type failed validation: typically, this means that one of its elements is missing such as a missing BLC, controller class, or JSP page.

`2003.08.05 15:35:28:359 Env/Thread-6:ER:CORE`
`java.lang.NoClassDefFoundError`
`java.lang.NoClassDefFoundError`
` at com.comergent.dcm.authentication.UserPasswordCredentials.verify(UserPasswordCredentials.java:38)`
` at com.comergent.dcm.authentication.LoginController.execute(LoginController.java:59)`
	On startup, the Sterling Multi-Channel Selling Solution has tried to re-instantiate a session stored when the servlet container was last stopped. Before starting the servlet container, make sure that you have deleted any stored sessions.
	For example, in Tomcat 4.1, check the ***container_home*****/ work/Standalone/localhost/Sterling/** directory, and delete any files called **SESSIONS.ser**.

Errors at Runtime

Some errors are listed here that have occurred infrequently in running instances of the Sterling Multi-Channel Selling Solution.

TABLE 13. **Runtime Errors**

Error	Cause and Solution
Assertion failed: 1 == pConnectionObject->fCallCheck, file q:\SPHINX\NETLIBS\nt\ssock\src\ntssockc.c, line 1039	This error has been observed when the Sterling Multi-Channel Selling Solution is run with SQL Server. Make sure that you have applied the latest Windows and SQL Server Service Packs to the machine on which SQL Server is running, and make sure that the client SQL Server software installed on the Sterling Multi-Channel Selling Solution machine matches your version of SQL Server.
On Solaris, the servlet container cannot find a certain servlet or URL.	First make sure that you did not make a typo. If you are certain that there was no mistake, then do the following: 1. Run the following command on **web.xml**: java com.comergent.dcm.util.CheckWebXML web.xml > newWeb.xml 2. Edit the file **newWeb.xml**. Look for the following string <!-- (8192) XXX BOUNDARY BREAK --> The start of the comment <!-- is the start of a 8192 boundary break. If it falls within a value for an XML node, then that node will get truncated. A work around is to pad the **web.xml** file such that the boundary break will fall inside a comment. For more information, see the comments at the start of file CheckWebXML.java.

TABLE 13. **Runtime Errors (Continued)**

Error	Cause and Solution
You see parser errors such as: `java.lang.NoSuchMethodError at org.apache.xpath.DOM2Helper.getNamespaceOfN-ode (DOM2Helper.java:348) at org.apache.xml.utils.TreeWalker.startNode (TreeWalker.java:281) at org.apache.xml.utils.TreeWalker.traverse (TreeWalker.java:119) at org.apache.xalan.transformer.TransformerI-dentityImpl.transform (TransformerIdentity-Impl.java:320)`	Check that you have followed the instructions to copy the XML parser-related JAR files to the servlet container's **lib/** directory, and that you have removed any default **parser.jar** files.
Running iPlanet, you see the following in your browser: `GX Error (GX2GX) socket result code miss-ing!!!`	There is a mismatch between the **web.xml** and **ias-web.xml** files. All servlets mentioned in **web.xml** must have a corresponding entry in the **ias-web.xml** file. Use the kguidgen utility to generate a GUID for the servlet.

Backing Up the Sterling Multi-Channel Selling Solution

It is good practice to plan for the possibility of a catastrophic failure that renders the Sterling Multi-Channel Selling Solution machine unusable. In this eventuality, you need to be able to restore the Sterling Multi-Channel Selling Solution as rapidly as possible.

We suggest taking the following steps:

1. Replicate the servlet container: keep the installable for the exact release of the servlet container, together with any patches applied. Back up any changes to archive files, or startup scripts that might affect the order of class-loading for example. A copy of the JDK used to run the servlet container would also be useful.

2. Back up the Sterling Multi-Channel Selling Solution itself: that is, create a WAR file of the running Sterling Web application directory. This will capture any changes to system properties, business rules, as well as the XML model files, resource files (product images and so on) and other files (such as uploaded GIF files).

   Note that the Sterling Multi-Channel Selling Solution enables a fair amount of customization that can place files outside of the Sterling Multi-Channel Selling Solution Web application directory. If you take advantage of this capability, then you have to backup these directories too. In particular, a clustered installation of the Sterling Multi-Channel Selling Solution requires the creation of a shared location that will be external to the Web application directory on any particular machine.

3. Take a snapshot of the database at regular intervals: verify that the Sterling Multi-Channel Selling Solution Knowledgebase can be restored from this backup.
4. Make a copy of the **dcmsKey.ser** file and put this in a very safe place. Encrypted data will be unrecoverable if this file is lost. For customers with Release 6.3 or higher, this instruction needs to be modified depending on your choice of encryption scheme and key management policy.

CHAPTER 9

Customizing Your Sterling Multi-Channel Selling Solution

The previous two chapters described the process of installing the Sterling Multi-Channel Selling Solution using the configuration and data provided by Sterling Commerce. This chapter covers the steps that you perform to customize your implementation of the Sterling Multi-Channel Selling Solution. By the time you start work in this chapter, you must have installed the Sterling Multi-Channel Selling Solution and have verified that it can connect to a populated Knowledgebase. If you have not done so, then refer to CHAPTER 5, "Installing the Sterling Multi-Channel Selling Solution" and CHAPTER 6, "Creating and Populating the Knowledgebase".

The *Sterling Multi-Channel Selling Solution Reference Guide* and *Sterling Multi-Channel Selling Solution Developer Guide* provide additional information that you may need to consult as you customize your Sterling Multi-Channel Selling Solution. The *Sterling Multi-Channel Selling Solution Reference Guide* provides a complete description of the underlying configuration files, XML schema, database schema, and related reference data of the Sterling Multi-Channel Selling Solution. The *Sterling Multi-Channel Selling Solution Developer Guide* provides a detailed description of the architecture and principal Java classes of the Sterling Multi-Channel Selling Solution to enable systems integrators to customize the system.

Configuration Overview

The following figure is a schematic diagram of the Sterling Multi-Channel Selling Solution architecture.

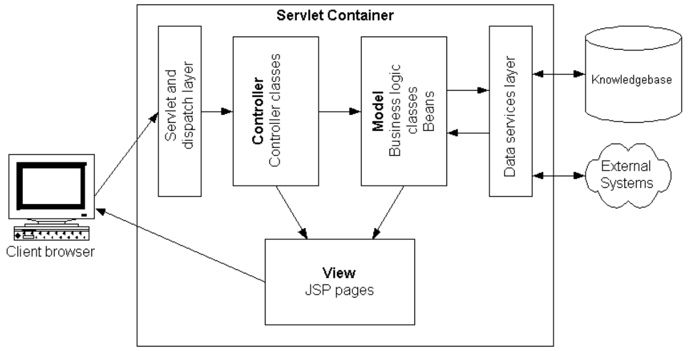


FIGURE 7. **Sterling Multi-Channel Selling Solution Architecture**

The principal areas of customization are customer-facing applications such as Sterling Advisor and Sterling Quotes, and the applications that provide integration with existing ERP systems such as Sterling Integrator and Sterling Pricing.

In performing customization, you have to consider:

- to support a multi-locale experience for your customers, you can implement the Sterling Multi-Channel Selling Solution to serve different data and Web pages for different locales. See CHAPTER 11, "Internationalization" for more information.
- Data Objects: modifications to existing data objects or the creation of new ones. You must also consider what security to enforce on the business objects by using the ACLs.
- Controllers: modifications to existing controllers or the creation of new ones.
- Business Logic Classes: modifications to existing business logic classes or the creation of new ones.
- JSP Pages: modifications to existing JSP pages or the creation of new ones.

- Message Types, Entitlements, and User Roles: you must create new message types to support additional application logic and you must consider what roles may execute the message types.
- Configuration Files: general configuration information is maintained in the configuration files. You must review the configuration settings for your implementation.
- Populating the Knowledgebase Tables: If you plan to populate some or all of the Knowledgebase using existing data, then you must consider your strategy to import the data into the Sterling Multi-Channel Selling Solution.

Data Objects

Customizing Data Objects

A set of reference data objects are defined by the XML schema provided with the Sterling Multi-Channel Selling Solution. These may be sufficient for your implementation. However, you may need to add fields to existing data objects or implement new data objects. See the *Sterling Multi-Channel Selling Solution Reference Guide* and *Sterling Multi-Channel Selling Solution Developer Guide* for further information about data objects.

If you make changes to data objects or create new ones, then you must run the generateDTD target to generate the DTDs for the data objects. Then you must also run the generateBean target. The generated Bean classes and interfaces are compiled into the com.comergent.bean.simple package.

Controllers

Controller classes are the main classes used to perform the business logic of the Sterling Multi-Channel Selling Solution and to process requests. If you want to modify the business logic of the system or to change the structure of data objects, then you must modify the related controllers or create new ones.

Business Logic Classes

Business logic classes (BLCs) manage some of the business logic of the Sterling Multi-Channel Selling Solution applications.

Note:	As of Release 6.0, the use of BLCs is deprecated. Use controller or bizAPI classes to manage application business logic.

If you want to modify existing business logic or add new functionality, then you can modify or create BLCs. See the *Sterling Multi-Channel Selling Solution Reference Guide* and *Sterling Multi-Channel Selling Solution Developer Guide* for more information.

As well as modifying and creating BLCs, you must also ensure that the BLCs are called in response to particular message requests. This involves modifying the **MessageTypes.xml** configuration file. See "Message Types, Entitlements, and User Roles" on page 154 for more information.

JSP Pages

JSP (JavaServer Pages) pages are used to dynamically generate the HTML content that is returned to users' browsers. You should not need to modify the administration pages, but you may have to customize the customer-facing JSP pages for the following reasons:

- To add company and branding to the pages.
- To add navigation and modified work flow links.
- To add modified or new business object information to the pages.

See the *Sterling Multi-Channel Selling Solution Developer Guide* for examples on how to customize JSP pages.

Customizing Catalog Navigation and Display

Navigation Levels

You can customize the catalog navigation menu to manage the number of category levels visible to users as they navigate down through the catalog.

1. Set the Sterling Advisor business rule Sterling Advisor Maximum Number of Menu Shown to the requisite number of levels to display.

2. Edit the **catalog/CatalogNavigateBody.jsp** JSP page to set the values of gA_COORDS_TOP and gA_COORDS_LEFT. These determine exactly where the menu is positioned on the page.

Catalog Display Styles

You can customize how product categories are displayed to end-users by associating display styles to product categories. Each display style is defined as a message type: this determines the JSP page used to display the product category. The **CatalogDisplayStyle.xml** configuration file is used to define the display styles. This file is described in the *Sterling Multi-Channel Selling Solution Administration Guide*.

Enterprise product managers can use the Sterling Product Manager UI to assign display styles to particular product categories. As well as specifying the display style, they can also specify parameters to be used as part of the request, and these can be used by the controller and JSP page that are used to display the product category.

Disabling Catalog Access for Anonymous User

You can customize the Sterling Multi-Channel Selling Solution home page so that the anonymous user can only log in or register, and cannot browse the catalog. To prevent the anonymous user from browsing the catalog, ensure that you update the **WEB-INF\properties\Entitlements.xml** file to grant only the CreateNewUser message type to the anonymous user and remove the grants for all other message types.

Customizing Partner Selector

The Sterling Multi-Channel Selling Solution provides a means for end-users to request that partners contact them: the user specifies some criteria to determine what partners receive their inquiry: this is known as the Partner Selector. You can change how the Partner Selector works as follows:

1. Modify the **ResellerSelectorSelectData.jsp** page to add or remove the selection criteria that you want users to be able to specify.
2. Modify the ResellerSelectorListController.java controller class: this class extracts the selected criteria when a user submits the selection form from the **ResellerSelectorSelectData.jsp** page.
3. Modify the ResellerSelectorListSelectController.java controller class to build up the query that should be used to retrieve the list of selected partners.

Message Types, Entitlements, and User Roles

When a user clicks a link on a Sterling Multi-Channel Selling Solution page, the URL includes a message type (as the "cmd" parameter). This message type is used to determine which controller is invoked to process the request, and it also determines the bizAPI classes, controller classes, and JSP pages that are used to process the request and generate the response.

You maintain the mapping from message type to bizAPI and controller classes and JSP page in the **MessageTypes.xml** configuration files. If in modifying or creating new business logic, you create a new message type, then make sure that you add the message type to **MessageTypes.xml** and that it is mapped to the appropriate bizAPI class, controller class, and JSP page.

Entitlements

The Sterling Multi-Channel Selling Solution ensures that only authorized users may execute message types. When you create each user, you can assign them one or more functions: these map to entitlement roles and these roles determine what message types the user may execute.

Each message type belongs to a message group and each message group may be granted or denied to a user role. You define the assignment of message groups to roles in the **Entitlements.xml** file.

User Roles

The Sterling Multi-Channel Selling Solution makes use of user roles in two different ways in the security mechanisms.

- First, the roles are assigned message groups as described above.
- Second, user roles are used to filter access to business objects defined in the access control lists (ACLs). See the *Sterling Multi-Channel Selling Solution Reference Guide* for more information.

Configuration Files

The Sterling Multi-Channel Selling Solution maintains configuration information in configuration files. You must edit these files to provide the configuration information specific to your implementation of the Sterling Multi-Channel Selling Solution.

Customizing the Configuration Files

Once you have prepared the Knowledgebase, you must make sure that the pointers to the database are set appropriately in the configuration files. You should use the SDK to work with making changes in these files, but you can also modify them manually as described in this section.

To Configure your Sterling Multi-Channel Selling Solution

1. Modify the main configuration file, **Comergent.xml**, to match your installation.

   By default, this file is in the ***debs_home*/Sterling/WEB-INF/properties/** directory. Check that the **Comergent.xml** file has correct entries for the following elements:

   - entitlementFilename: this element points to the file in which you manage user entitlements.
   - messageTypeFilename: this element points to the files that map message requests to the BLCs, PLCs, controllers, HTML templates, and JSP pages.
   - ConverterMap: this element points to the file that determines which Converter class is to be used to process the XML messages.
   - DataServices: this element points to the file that controls access to the data layer.
   - ServerName: set this element to the external name for this machine.
2. Edit **DataServices.xml**.
   - If you are running against an IBM DB2 Universal Server database:
     - Change the DsDataSources element to point to **DB2DataSources.xml**.
     - Change the value of the DsKeyGenerators element to **DB2KeyGenerators.xml**.
   - If you are running against an Oracle Server database:
     - Change the DsDataSources element to point to **OracleDataSources.xml**.
     - Change the value of the DsKeyGenerators element to **OracleKeyGenerators.xml**.
   - If you are running against an SQL Server database:

    - Change the DsDataSources element to point to **ODBCDataSources.xml**.
    - Change the value of the DsKeyGenerators element to **ODBCKeyGenerators.xml**.

   Note that earlier releases used these files to connect to a SQL Server deployment of the Sterling Multi-Channel Selling Solution.

    - Change the DsDataSources element to point to **MsSqlDataSources.xml**.
    - Change the value of the DsKeyGenerators element to **MsSqlKeyGenerators.xml**.

3. Edit the appropriate **DataSources.xml** file.
   - If you are running against an IBM DB2 Universal Server database, edit the **DB2DataSources.xml** file:
     - Change the value of the ConnectionString attribute to point to the machine name, port, and database of the DB2 instance.
     - Change the UserId and Password attributes to connect to the appropriate DB2 database.
   - If you are running against an Oracle Server database, edit the **OracleDataSources.xml** file:
     - Change the value of the ConnectionString attribute to point to the machine name, port, and SID of the Oracle instance.
     - Change the UserId and Password attributes to connect to the appropriate Oracle database.
   - If you are running against an SQL Server database, edit the **ODBCDataSources.xml** file:
     - Change the value of the ConnectionString attribute to the machine name of the SQL Server instance.
     - Change the UserId and Password attributes to connect to the appropriate SQL Server database.
4. Modify the **MessageTypes.xml** files.

   By modifying this file, you enable the Sterling Multi-Channel Selling Solution to support the appropriate message versions.

Populating the Knowledgebase Tables

Data Objects Data

The following are the principal ways in which you can enter data into the Sterling Multi-Channel Selling Solution:

- You can use the administration pages to create partner profiles, users, products, price lists, and so on. See the *Sterling Multi-Channel Selling Solution Administration Guide* for more information.
- You can use the Sterling Integrator to synchronize data with an existing ERP system. See CHAPTER 19, "Implementing Order Management Integration" for more information.
- You can use a scripting technique to load the data into the Knowledgebase.

In general, the scripting technique requires you to create a “data bridge” from the source of the data to the Sterling Multi-Channel Selling Solution.

XML Data Loading

You can load the Knowledgebase data using XML documents that define the data of each business object. Data for both the reference and minimal implementations of the Sterling Multi-Channel Selling Solution is loaded this way.

XML data loading exercises the business logic of the Sterling Multi-Channel Selling Solution. You create the definition of each data object as part of an XML document, and then use the XML data loader batch script to load the data.

A typical form of the data object definition is:

```
<Promotion state="INSERTED" type="BusinessObject">
    <PromotionKey state="INSERTED">1</PromotionKey>
    <PromoCode state="INSERTED">ID1</PromoCode>
    <PromotionName state="INSERTED">Free Support</PromotionName>
    <Description state="INSERTED">Get Free Support</Description>
    <URL state="INSERTED">STATIC_URL/promotions/PartnerA1.htm</URL>
    <AddToCartSku state="INSERTED">MXWS-7600</AddToCartSku>
    <AddToCartQty state="INSERTED">3</AddToCartQty>
    <OwnedBy state="INSERTED">1</OwnedBy>
    <AccessKey state="INSERTED">206</AccessKey>
    <PromotionUpdatedBy state="INSERTED">3</PromotionUpdatedBy>
    <PromotionCreatedBy state="INSERTED">3</PromotionCreatedBy>
    <PromotionControl state="INSERTED">
        <ControlKey state="INSERTED">1</ControlKey>
        <ControlName state="INSERTED">New PControl</ControlName>
        <StartDateActive state="INSERTED">
```

```
            2000-01-01 12:00:00.0
        </StartDateActive>
        <EndDateActive state="INSERTED">
            2000-12-31 23:59:00.0</EndDateActive>
        <Priority state="INSERTED">5</Priority>
        <EnabledFlag state="INSERTED">true</EnabledFlag>
        <PartnerType state="INSERTED">Systems Integrator</PartnerType>
        <MemberLevel state="INSERTED">Platinum</MemberLevel>
        <ControlUpdatedBy state="INSERTED">3</ControlUpdatedBy>
        <ControlCreatedBy state="INSERTED">3</ControlCreatedBy>
    </PromotionControl>
    <PromotionSkuList state="INSERTED">
        <PromotionSku state="INSERTED">
            <SKU state="INSERTED">MXLP-7410</SKU>
            <UpdatedBy state="INSERTED">3</UpdatedBy>
            <CreatedBy state="INSERTED">3</CreatedBy>
        </PromotionSku>
    </PromotionSkuList>
</Promotion>
```

In general, the data bridge should generate the XML data object definitions automatically. See "Populating the Knowledgebase" on page 109 for more information on running the XML data loading scripts.

SQL Scripts

It is possible to write SQL scripts to insert data directly into the Knowledgebase. For example, use SQL statements along these lines to insert partner profile data into the Knowledgebase:

```
INSERT INTO CMGT_PARTNERS (UPDATED_BY, CREATED_BY, PARTNER_NAME,
LEGAL_NAME, PARENT_COMPANY, STATUS, ADDRESS_KEY, EMAIL_ADDRESS,
DUNBRAD_ID, BUSINESS_ID, TYPE, MEMBER_LEVEL, BUSINESS_TRANSACTION,
NET_WORTH, NUM_OF_EMPLOYEES, POT_REV_CURR_FY, POT_REV_NEXT_FY,
REFERENCE_USE_FLAG, COTERM_DAY_MONTH, URL,YEAR_ESTD, ANALYSIS_FY,
FISCAL_YEAR_END_MONTH, OWNED_BY, ACCESS_KEY, XML_MESSAGE_VERSION,
DISTI_ACCESS) VALUES (1, 1, 'AffinityNet', 'Affinity Net Technolo-
gies','Affinity Corporation', 'A', 2,'gmehra@stercomm.com', 'bus0231-
a34f', 'M00212G', 'Reseller', 'Gold', 'MDF', 10000000.00, 60,
12500000.00, 15000000.00, 'Y', sysdate, 'www.affinitynet.com',
'December 1989','','November',1,2, 'Comergent 2.0', 1);
```

However, this is a delicate and time-consuming process: you have to bear in mind when primary keys are being generated automatically. When you need to use a foreign key in an INSERT statement that refers to the primary key of another business object make sure that the primary key has been created.

SQL scripts are also database server-specific: you cannot use the same script to load data into DB2 Universal Server, SQL Server, and Oracle Server databases. We suggest that you use XML data loading as described above.

Common Attributes

In maintaining information in the Knowledgebase, you make use of commonly used attributes that are referenced by different applications. These attributes include:

- currencies such as US dollars, Japanese Yen, and so on
- locales
- order statuses
- partner types, such as “Distributor”, “Reseller”, and so on
- partner levels, such as “Gold”, “Silver”, and so on
- services that each partner offers
- skills that each partner offers
- territories in which each partner works
- customer types (formerly vertical markets) in which each partner operates

To ensure that the correct values for these attributes are available for selection when an administrator is working with the Sterling Multi-Channel Selling Solution, you must populate the corresponding database tables with the available values. This section describes the steps required to do this with XML data loading.

The **LightWeightLookupList** XML file in the XML data loading directory provides a template to demonstrate how data may be loaded into the CMGT_LOOKUPS table. See the *Sterling Multi-Channel Selling Solution Reference Guide* for information about this table.

You load some attribute data into their own dedicated tables. These include: currencies, locales, ratings, services, territories, and customer types (vertical markets). You must edit the corresponding XML data loading files to make modifications to this data.

Locales

You must make sure that only the locales that you want to support are declared in the **LocaleDataList** file.

Partners

Every implementation of the Sterling Multi-Channel Selling Solution must create these partners:

- Enterprise
- AnonymousUserPartner
- RegisteredUserPartner

Make sure that you do not change the names of these three partners.

Request and Message Processing

The Sterling Multi-Channel Selling Solution responds to incoming *requests* received from user's browsers and to incoming *messages* received from other Sterling Multi-Channel Selling Solution servers.

Each request is the result of a user clicking a link or button on a Sterling Multi-Channel Selling Solution Web page or of an external system posting an XML message to the Sterling Multi-Channel Selling Solution. Each request is processed by a *controller*, and sometimes a BLC, and usually uses a JSP page to display the result of the request in the user's browser.

Messages are sent from one Sterling Multi-Channel Selling Solution to another. When the Sterling Multi-Channel Selling Solution receives a message from another server, it must know how to process the message. It first converts the message from its external message type to the internal Comergent XML message type. Then, for each message type, the server must assign a BLC that processes messages of that type.

Sterling Multi-Channel Selling Solution configuration files, **MessageTypes.xml**, specify the mapping from requests to BLCs and controllers. Each request may be also mapped to a JSP page as specified in the **MessageTypes.xml** files.

Each Sterling XML message is converted into business objects by a converter class. To ensure that each message is converted into the correct business objects, you must specify which converter class is used for a family of messages. You do this using the **ConverterMap.xml** configuration file.

Each external message must be converted from its external form to Sterling XML. This is done through the MessagingServlet, the MessageCracker, and Converter classes. See CHAPTER 18, "Message Conversion" for more information.

If you do not change any message types or business objects, then you do not need to implement any changes to message conversion. If however, you change the definition of business objects used in processing external messages, then you should also consult CHAPTER 18, "Message Conversion".

Configuring the Data Services Layer

This section describes the most important configuration parameters associated with the data services layer. It covers:

- How do I control what JDBC Drivers are loaded?
- How do I set up a Custom Data Service?
- How can I control connection behavior?
- How do limit my query results and control pagination?
- How do change where result set page files are written?
- What is the ServerId used for?
- How do I ensure locale specific sorting of query results?
- What is the purpose of the database specific properties?
- What does MaxRequestsPerConnect control?
- What are the other database specific properties?

All the elements referred to in this section are specified in the **DataServices.xml** configuration file, and may be set through the System Administration user interface.

How do I control what JDBC Drivers are loaded?

The <JdbcDriver1> through <JdbcDriverN> elements control which JDBC drivers are loaded. If no JDBC driver is required, then the property can be commented out.

How do I set up a Custom Data Service?

It is possible to plug-in custom services to provide access to special data sources. These services must support the DataInterface interface for non-relational data stores, or its extension the SQLDataInterface interface if they access relational data stores.

If you have created a custom Data Service class that implements the DataInterface interface, then it can be registered with DataServices by adding an entry in the **DataServices.xml** properties file.

The entry will be of the form:

```
<CustomService controlType="text" runtimeDisplayed="true"
    ChangeOnlyAtBootTime="true" visible="true" boxsize="45"
    displayQuestion="Custom Service Class Name"
    defaultChoice="com.comergent.dcm.dataservices.CustomService"
    help="Enter the Service class that implements DataInterface.">
    com.comergent.dcm.dataservices.CustomService
</CustomService>
```

The **DataSources.xml** schema file may now refer to a “Custom” service and the property will be used to determine the class path and load the service dynamically.

How can I control connection behavior?

The ConnectTimeout element controls how many minutes pass before a connection is revalidated. If the property is set to 0, then there is no timeout. This property works in combination with ReconnectOnTimeout.

The ReconnectOnTimeout element controls whether the connection is discarded when the timeout period has elapsed. When a connection is retrieved from the pool, if this property is set to true and the timeout period has elapsed the connection will be closed and a new connection will be established automatically. If this property is set to false, then the connection will be tested and will be reestablished if the connectivity test fails.

These properties can be used to resolve dropped connections or firewall restrictions.

How do limit my query results and control pagination?

The setting of the MaxResults element determines the maximum number of records that can be retrieved during a restore. When that number is reached the request is freed and any additional results are discarded.

The setting of the NumPerPage element determines how many records are saved in each result cache page. If the number found is less than NumPerPage, then no result cache is created.

Note that this combination of attributes now allow the application to retrieve a set of paginated results while still specifying a maximum number of records to retrieve.

These properties control the system default behavior. They can be overridden by the application using the DataContext.

How do change where result set page files are written?

The rsCachePath element provides the file path used for page files.

What is the ServerId used for?

The ServerId element is used to assist with the generation of unique key values in a clustered environment. This is only necessary if key generators use the MaxValue option to generate key values (normally used for Microsoft SQL Server).

With the MaxValue approach, the current key value is seeded using the maximum key value obtained from the database at startup. The next available key is then maintained in memory. In a clustered environment this would normally cause a conflict since multiple machines are keeping track of keys independently. To eliminate this problem, the last two digits of the key contain the ServerId.

The two-digit ServerId can contain any unique value from 00 to 99.

How do I ensure locale specific sorting of query results?

The UseLocalizedSort element determines whether we ask any underlying database to perform locale-specific sorting of query results.

Turning on locale specific sorting will slow down many queries since the database is no longer able to rely on index ordering to control the result set order. The database may instead need to perform a secondary sort that reflects the locale specific sort sequence. By default this property is set to false.

What is the purpose of the database specific properties?

While most RDBMS follow ANSI standards for behavior and SQL support, there are minor variations between them. There are also differences in efficiency for certain operations. The database specific properties allow us to tune our behavior to suit each specific database type.

What does MaxRequestsPerConnect control?

Our connection pooling mechanism allows read-only connections to be shared for multiple concurrent requests. With some database drivers this can improve throughput by reducing connection resources. The drivers currently supported by the Sterling Multi-Channel Selling Solution work more efficiently if there is only one concurrent request per connection. We recommend leaving this at the default setting of 1.

What are the other database specific properties?

The DATE element specifies the JDBC datatype returned for a DATE. This is necessary due to deviations from the JDBC standard by some drivers.

The SupportsIntersect element indicates if the underlying database recognizes the SQL INTERSECT operator. If it does not, then we will attempt to transform an INTERSECT into an equivalent request using sub-queries.

The LowerCase element indicates the SQL function that converts strings to lowercase for the selected database. For Microsoft SQL Server we leave this property empty since SQL Server can be set to perform case insensitive string comparisons.

The UpperCase element indicates the SQL function that converts strings to uppercase for the selected database. For Microsoft SQL Server we leave this property empty since SQL Server can be set to perform case insensitive string comparisons.

Order Process Modeler

The Order Process Modeler is a mechanism to manage the business logic of the order management cycle. The Order Process Modeler manages the state transitions of an order as it proceeds through the stages after it has been placed by a user. See CHAPTER 20, "Order and Return Management" for more details.

CHAPTER 10

Managing Sterling Multi-Channel Selling Solution Logging

This chapter provides a description of the logging service used to manage logging messages in the Sterling Multi-Channel Selling Solution. The *Sterling Multi-Channel Selling Solution Developer Guide* provides a more detailed view of how to use the logging API to write logging messages.

Logging

Use the logging settings of the Sterling Multi-Channel Selling Solution to monitor activity of the Sterling Multi-Channel Selling Solution and to help diagnose problems.

Logging Preferences and Configuration

The log4j API handles logging and uses the Preferences API to retrieve logging configuration properties. The basic configuration file for the log4j API is **log4j.properties**. A copy of this file with default logging properties is included in the **WEB-INF/lib/cmgt-logging.jar** JAR file packaged with the Sterling Multi-Channel Selling Solution and is also placed in the **WEB-INF/classes/** directory. To override the default properties permanently, you must modify the **log4j.properties** file. Any values that you specify in this file will overwrite the corresponding values in the **log4j.properties** file in the **cmgt-logging.jar** file. You can override the default properties on a transient basis for testing purposes by logging in to the System Administration site of the Sterling Multi-Channel Selling Solution as a site administrator and modifying the System Logging properties. See "Making Transient Logging Configuration Changes" on page 167 for more information.

The following sections describe some typical changes you may want to make:

- "Logging to the Console" on page 166
- "Changing Logging Level for a Package" on page 166
- "Formatting Logging" on page 167
- "Logging File Size" on page 167

Logging to the Console

If you want logging output to the standard output stream, rather than to a logging file, specify the use of the STDOUT appender:

```
log4j.rootCategory=info, STDOUT
```

Depending on the configuration of the servlet container, the logging output will be directed to the standard destination of the System.out output stream. Note that when you specify a different appender, then you must include the appender’s properties in the custom **log4j.properties** file too. For example:

```
log4j.appender.STDOUT=org.apache.log4j.ConsoleAppender
log4j.appender.STDOUT.layout=org.apache.log4j.PatternLayout
log4j.appender.STDOUT.layout.ConversionPattern=[%t] (%c{2}) - %m%n
```

Note that you can also force logging to be output to the System.out output stream by specifying -Dcomergent.console.logging=true as part of the command line that starts the servlet container. This overrides any logging properties specified in the **log4j.properties** configuration file.

Changing Logging Level for a Package

If you want to see more detailed logging information from just one Java package as it is executed, then you can specify this by overriding the root logging level. By default, all logging is done at the INFO level, because the rootCategory is defined as follows:

```
log4j.rootCategory=info, CMGT
```

For example, to specify DEBUG level logging for the visualModeler API package, enter:

```
log4j.logger.com.comergent.api.apps.visualModeler=DEBUG
```

The specification of logging is hierarchical following the package organization of the Sterling Multi-Channel Selling Solution. Thus if you specify:

```
log4j.logger.com.comergent.api.apps=WARNING
```

Then, all logging at the API level will be done at the WARNING level except for the visualModeler package.

Formatting Logging

You can format the logging output to suit your needs. For example, the following will provide a more compact logging format than the standard default layout:

```
log4j.appender.CMGT.layout.ConversionPattern=[%r] [%t] (%c{1}) - %m%n
```

Logging File Size

If log files get too large, then consider modifying the logging preferences to rotate the log files. For example, you can specify that log files are rotated once they reach 10MBytes in size as follows:

After the line:

```
log4j.appender.CMGT=com.comergent.logging.ComergentRollingFileAp-
pender
```

add:

```
log4j.appender.CMGT.MaxFileSize=100KB
```

Alternatively, to specify that log files should be rotated daily, change:

```
log4j.appender.CMGT=com.comergent.logging.ComergentRollingFileAp-
pender
```

to:

```
log4j.appender.CMGT=com.comergent.logging.ComergentDailyRolling-
FileAppender
```

Making Transient Logging Configuration Changes

If you want to change the logging configuration settings for troubleshooting or other testing purposes, then make the changes as a site administrator using the **System Logging (log4j dynamic)** page of the site administration's System Properties page. Changes that you make remain in effect until you restart the Sterling Multi-Channel Selling Solution. If you are working in a clustered environment, then the logging configuration changes will be propagated to all the nodes in the cluster, and will also remain in effect until you restart the Sterling Multi-Channel Selling Solution.

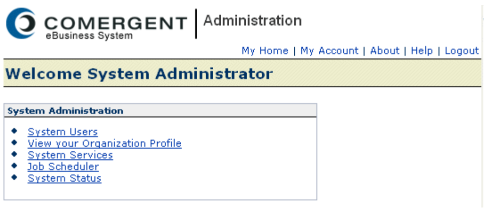
To make permanent logging level changes, you must modify the **log4j.properties** file, as described in "Logging Preferences and Configuration" on page 165.

To make transient logging configuration changes:

1. Navigate to the System Administration URL and log in as a site administrator. The System Administration URL is similar to:

`http://`*`server`*`:`*`port`*`/Sterling/en/US/enterpriseMgr/admin`

   The System Administration home page displays, similar to the following figure:



COMERGENT eBusiness System | Administration

My Home | My Account | About | Help | Logout

Welcome System Administrator

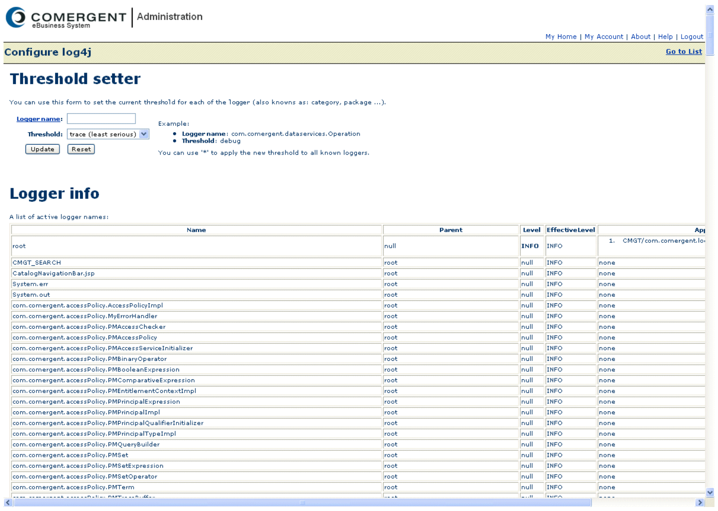
System Administration

- System Users
- View your Organization Profile
- System Services
- Job Scheduler
- System Status

FIGURE 8. System Administration Home Page

2. Click **System Services**, then click **System Logging (log4j dynamic)**.

   The **Configure log4j** page displays, similar to the following figure.



COMERGENT eBusiness System | Administration

My Home | My Account | About | Help | Logout

Configure log4j — Go to List

Threshold setter

You can use this form to set the current threshold for each of the logger (also knowns as: category, package ...).

Logger name:

Threshold: trace (least serious)

Update | Reset

Example:

- **Logger name**: com.comergent.dataservices.Operation
- **Threshold**: debug

You can use '*' to apply the new threshold to all known loggers.

Logger info

A list of active logger names:

Name	Parent	Level	EffectiveLevel	Ap
root	null	**INFO**	INFO	1. CMGT/com.comergent.lo
CMGT_SEARCH	root	null	INFO	none
CatalogNavigationBar.jsp	root	null	INFO	none
System.err	root	null	INFO	none
System.out	root	null	INFO	none
com.comergent.accessPolicy.AccessPolicyImpl	root	null	INFO	none
com.comergent.accessPolicy.MyErrorHandler	root	null	INFO	none
com.comergent.accessPolicy.PMAccessChecker	root	null	INFO	none
com.comergent.accessPolicy.PMAccessPolicy	root	null	INFO	none
com.comergent.accessPolicy.PMAccessServiceInitializer	root	null	INFO	none
com.comergent.accessPolicy.PMBinaryOperator	root	null	INFO	none
com.comergent.accessPolicy.PMBooleanExpression	root	null	INFO	none
com.comergent.accessPolicy.PMComparativeExpression	root	null	INFO	none
com.comergent.accessPolicy.PMEntitlementContextImpl	root	null	INFO	none
com.comergent.accessPolicy.PMPrincipalExpression	root	null	INFO	none
com.comergent.accessPolicy.PMPrincipalImpl	root	null	INFO	none
com.comergent.accessPolicy.PMPrincipalQualifierInitializer	root	null	INFO	none
com.comergent.accessPolicy.PMPrincipalTypeImpl	root	null	INFO	none
com.comergent.accessPolicy.PMQueryBuilder	root	null	INFO	none
com.comergent.accessPolicy.PMSet	root	null	INFO	none
com.comergent.accessPolicy.PMSetExpression	root	null	INFO	none
com.comergent.accessPolicy.PMSetOperator	root	null	INFO	none
com.comergent.accessPolicy.PMTerm	root	null	INFO	none

FIGURE 9. Configure log4j Page

3. Copy the name of the logger you want to change and paste it into the **Logger name:** field, then choose a logging level from the **Threshold** drop-down list.

   Logging levels range from trace (logging all activity) to fatal (logging only fatal errors). You can also use the drop-down list to turn logging off.

4. Click **Update** to update the logging level. The new logging level displays in the Level column for the logger.

Logging File Locations

The location of logging files varies from one servlet container to another: here are some standard locations:

TABLE 14. **Servlet Container Log Files**

Servlet Container	Log File Location
Tomcat	***container_home*/logs/**
WebLogic	***container_home*/user_projects/domains/mydomain/**
WebSphere	***container_home*/logs/**

CHAPTER 11 Localization

You can provide your customers with an e-commerce experience in their preferred language and location, or locale.This chapter describes how to add support for locales other than United States English to an implementation of the Sterling Multi-Channel Selling Solution.

Individual locales are provided as localization pack JAR files that you install into your release using the SDK.

Attention: There are known issues with the Sterling Multi-Channel Selling Solution using SQL Server to support locales other than en_US. See "Microsoft SQL Server Requirements" on page 22 for more information.

This chapter covers the following topics:

- "Localization Concepts" on page 172
- "Localization Pack Installation Overview" on page 175
- "Localization Pack Installation Steps: New Implementation" on page 175
- "Localization Pack Installation Steps: Existing Implementation" on page 182

Localization Concepts

This section introduces localization concepts and the Sterling Multi-Channel Selling Solution support for localization.

Built-in Localization Support

The Sterling Multi-Channel Selling Solution has built-in support for:

- multiple currencies
- multiple languages
- number and date formats
- character sets

You can manage other aspects of localization for specific markets, such as:

- local laws and regulations
- currency processing
- shipping and export information
- tax

Locale Specification

You manage support for internationalization using locales. Each locale identifies a language and country. By identifying the locale to use when displaying information to a user, you ensure that the user sees information that is specific to their locale: they see your site’s Web pages in their preferred language, with numbers and dates in their expected format.

A locale comprises a language and a country: for example, “English and United States” or “Italian and Switzerland”. The same language may be used in more than one country: French in France, Switzerland, and Canada for example. In one country there may be speakers of more than one language: French, German, Italian, and Romansch in Switzerland for example.

The ISO standards 639 and 3166 specify a list of standard abbreviations for languages and countries that you must use. Some common language abbreviations are: Arabic (ar), Chinese (zh), English (en), French (fr), German (de), Hindi (hi), Japanese (ja), and Spanish (es).

Some common country abbreviations are: Canada (CA), China (CN), France (FR), Germany (DE), India (IN), Indonesia (ID), Japan (JA), United Kingdom (GB), and United States (US).

By combining a language and a country, you can uniquely specify a locale. For example: en_US (English-United States), it_CH (Italian-Switzerland), and zh_TW (Chinese-Taiwan). Locales are stored in the Sterling Multi-Channel Selling Solution using this representation.

Using Locales

Each installation of the Sterling Multi-Channel Selling Solution defines a *system default locale* in its **Internationalization.xml** file using the defaultSystemLocale element.

User Locales

When a user works in the Sterling Multi-Channel Selling Solution, their *current locale* determines the look-and-feel of Web pages and the locale-specific data (such as product descriptions) to use to display business object data to the user.

Each user has a *preferred locale* specified in their user profile. When a user first enters the Sterling Multi-Channel Selling Solution, their current locale is set to their preferred locale. If they change their current locale as they work, they see the Web pages in the new locale.

Users change locale by selecting a new locale from a drop-down list of available locales in their user profile. The display names for each locale in the drop-down list depend on the user's current locale. For example, if the user's current locale is en_US, the display name for fr_FR can be "French-France", and if the user's current locale is fr_FR, then the display name for en_US can be "Anglais-Etats Unis".

Default Locales for Languages

The Sterling Multi-Channel Selling Solution enables you to specify a default locale for each language so that if JSP pages are not available for the specific locale, the language's default locale's JSP pages are used instead. You specify the language's default locale using the defaultCountry elements of the Languages element of the **Internationalization.xml** configuration file.

The defaultSystemLocale element determines which JSP pages to serve if they are not provided in the language's default locale directory.

Sorting in Locales

The Sterling Multi-Channel Selling Solution enables users to sort displayed data in a number of ways while they perform their tasks. For example, they can sort the display of partners by name or inquiry lists by inquiry list ID. When a column is sorted based on a String value, you can specify whether the sorting is performed using the binary value of the String or whether a locale-specific sort is used. This switch is set at the system level, so the same method is used for all such sorts.

Note:	SQL Server support for user-locale sorting is limited. You can set only a single collating sequence which is then used for all users.

The sorting behavior is controlled by the UseLocalizedSort element of the **DataServices.xml** configuration file. The types of sorting behavior are:

- Binary: sorting is based on the binary value of the String value
- Locale specific

By default, the value of the UseLocalizedSort element is “false”: binary sorting is used. To use locale-specific sorting, set UseLocalizedSort to “true”. Note that binary sort is always fastest.

You can change the value of the UseLocalizedSort element by editing the **DataServices.xml** configuration file.

Site administrators can also change the UseLocalizedSort value as follows:

1. Navigate to the Sterling Multi-Channel Selling Solution administration site as a site administrator
2. Click **System Services** to display the System Properties page, then click **DataServices**
3. Scroll to Use Localized Sorting, then select the **false** or **true** radio button as required for your site

Note that you must stop and restart the Sterling Multi-Channel Selling Solution to enable the change.

The sorting behavior must be supported by the knowledgebase on which the Sterling Multi-Channel Selling Solution runs. See your database administrator for information about the type of sorting behavior supported for your implementation.

Localization Pack Installation Overview

The localization pack installation instructions describe how to install a Sterling Multi-Channel Selling Solution Release 8.0.1 localization pack using SDK 3.5.1. You can install into a new implementation or add support for a locale to an existing implementation. You install the localization pack into your release, not into your project.

These instructions assume that you are adding support for one or more locales to your new or existing implementation, but the default locale remains en_US.

What's contained in a localization pack:

- The resource bundle properties files (*.properties) for the locale.
- The locale's version of the look-up information from the **xmlloader** file, such as Keytype property names and values (descriptions).
- All the Javascript (.js) files.
- All the properties files with the locale appended to the file names, for example, **AttribMgrAGGENResources_fr.properties**.
- The .js files translated into the locale. The .js files retain their original names but are placed in locale-specific directories, such as fr\FR\js.

Note that the Matrix reference data, online help and online help images are not localized.

Localization Pack Installation Steps: New Implementation

These steps apply to a new implementation of the Sterling Multi-Channel Selling Solution.

These steps assume that you completed the Release 8.0.1 Sterling Multi-Channel Selling Solution set-up steps, including:

- Database configuration, privileges grants, and so on
- Installation of the Release 8.0.1 JAR file into your SDK
- Creation of a project using the sdk newproject target
- Installation of the appropriate database driver using the sdk install*DB* target, where *DB* is Oracle or MSSQLJDBC

See CHAPTER 5, "Installing the Sterling Multi-Channel Selling Solution" for detailed installation instructions.

1. Install the localization pack JAR file. For example:

```
sdk install SterlingSellingSuite-Locale-DEBS-8.0.1-fr-FR.jar
```

   This installs the localization pack in the **releases\debs-8.0.1** directory.

2. Modify the **web.xml** file:

   a. Enter the following in a command window:

```
sdk customize web.xml
```

   b. Open ***sdk_home*\projects\project-name\WEB-INF\web.xml** in a text editor.

   c. Copy the section that begins with the following:

```
 <!-- Start of English US mapping -->
```

   And ends with the following:

```
<!-- End of English US mapping -->
```

   d. Paste the entire section after the English US mapping section.

   e. Modify the new section’s comments to refer to the locale name you are adding. For example:

```
<!-- Start of French France mapping -->
```

   f. Modify the new section’s file path references to refer to the locale name file path, such as **/fr/FR**. For example:

```
<servlet-mapping>
      <servlet-name>DispatchServlet</servlet-name>
      <url-pattern>/fr/FR/catalog/*</url-pattern>
</servlet-mapping>
```

3. Configure the **Internationalization.xml** file.

   a. Enter the following in a command window:

```
sdk customize Internationalization.xml
```

   b. Open **projects\*project-name*\templates\WEB-INF\properties\Internationalization.xml** in a text editor.

   c. Add the locale designation to the Presentation element’s supportedLocales field. For example, for the French France locale:

```
<supportedLocales controlType="text" runtimeDisplayed="true"
ChangeOnlyAtBootTime="true" visible="true" boxsize="60"
```

```
displayQuestion="Presentation Locales"
displayOptions="en_US,en_US (English-United States),
zh_TW,zh_TW (Chinese-Taiwan),fr_FR,fr_FR
(French-France),fr_BE,fr_BE (French-Belgium),
de_DE,de_DE (German-Germany)" defaultChoice="en_US"
help="Supported presentation locales.">en_US,fr_FR</supportedLo-
cales>
```

d. Add the following to the Languages element. Replace ***la*** with the language code and ***CO*** with the country code for your locale:

```
<la visible="false">
<defaultCountry controlType="text" runtimeDisplayed="true"
ChangeOnlyAtBootTime="true" visible="false" boxsize="60"
displayQuestion="URL for the Help Files" defaultChoice="US"
help="This is the default country for a specific language">CO
</defaultCountry>
</la>
```

For example, for the French France locale:

```
<fr visible="false">
<defaultCountry controlType="text" runtimeDisplayed="true"
ChangeOnlyAtBootTime="true" visible="false" boxsize="60"
displayQuestion="URL for the Help Files" defaultChoice="FR"
help="This is the default country for a specific language">FR
</defaultCountry>
</fr>
```

4. Add the locale table entry.

   The locale table entry includes the DB_SORT_LOCALE_NAME field, which specifies how to sort data that your users are viewing. Possible values are BINARY (sort by the binary value of the string data value) and LATIN_GENERAL_BIN (perform locale-specific sorting). See "Sorting in Locales" on page 174 for information about determining sorting behavior for your implementation. Check with your Database Administrator to ensure that the sorting behavior that you select is supported for your database implementation.

   For Oracle-based implementations:

   a. Enter the following in a command window:

   ```
   sdk customize oracle_tables.sql
   ```

   b. Open **projects\*project-name*\WEB-INF\sql\Oracle\setup\oracle_tables.sql** in a text editor.

c. Search for the text "INSERT INTO CMGT_LOCALE"

d. Add an INSERT statement for the locale you are installing after the INSERT INTO CMGT_LOCALE statement. The format is:

```
INSERT INTO CMGT_LOCALE
(LOCALE_KEY,LOCALE_NAME,LOCALE_DESCRIPTION,ACTIVE_FLAG,DB_SORT_LO
CALE_NAME )
VALUES ( key,'la_CO','Country Language','Y','BINARY')
/
```

- *Key* is the locale key. The locale key must be a unique numeric value.
- *la_CO* is the language code and country code. Use the appropriate language_COUNTRY encoding for the locale you are installing, for example, fr_FR for French France, jp_JP for Japanese Japan, and so on.

  For example, the following INSERT statement is for supporting the fr_FR (French France) locale:

```
INSERT INTO CMGT_LOCALE
(LOCALE_KEY,LOCALE_NAME,LOCALE_DESCRIPTION,ACTIVE_FLAG,DB_SORT_LO
CALE_NAME )
VALUES ( 2,'fr_FR','France French','Y','BINARY')
/
```

e. Save and close the file.

For SQL Server 2005-based implementations:

a. Enter the following in a command window:

```
sdk customize mssql_schema.sql
```

b. Open **projects\*project-name*\WEB-INF\sql\MSSql\setup\mssql_schema.sql** in a text editor.

c. Search for the text "INSERT INTO CMGT_LOCALE"

d. Add an INSERT statement for the locale you are installing after the INSERT INTO CMGT_LOCALE statement. The format is:

```
INSERT INTO CMGT_LOCALE
(LOCALE_KEY,LOCALE_NAME,LOCALE_DESCRIPTION,ACTIVE_FLAG,
DB_SORT_LOCALE_NAME ) VALUES ( key,'la_CO','Country Lan-
guage','Y','LATIN_GENERAL_BIN')
GO
```

- *Key* is the locale key. The locale key must be a unique numeric value.
- *la_CO* is the language code and country code. Use the appropriate language_COUNTRY encoding for the locale you are installing, for example, fr_FR for French France, jp_JP for Japanese Japan, and so on.

  For example, the following INSERT statement is for supporting the fr_FR (French France) locale:

```
INSERT INTO CMGT_LOCALE
(LOCALE_KEY,LOCALE_NAME,LOCALE_DESCRIPTION,ACTIVE_FLAG,
DB_SORT_LOCALE_NAME )
VALUES ( 2,'fr_FR','France French','Y','LATIN_GENERAL_BIN')
GO
```

e. Save and close the file.

5. Add the country translations:

a. Enter the following in a command window:

```
sdk customize LocaleNameDataList
```

b. Open **projects\*project-name*\WEB-INF\xmldata\LocaleNameDataList** in a text editor.

c. The **LocaleNameDataList** file should contain only English and the language mappings for the locale(s) that you support. Remove all other Locale Mappings from the **LocaleNameDataList** file.

d. At the end of the LocaleNameDataList clause, add lines to supply translations for the United States and the country for which you are installing a locale. For example, to provide French translations for the United States and France:

```
<LocaleNameData state="INSERTED">
  <DisplayName state="INSERTED">France</DisplayName>
  <EffectiveLocale state="INSERTED">en_US</EffectiveLocale>
  <LocaleName state="INSERTED">fr_FR</LocaleName>
</LocaleNameData>
<LocaleNameData state="INSERTED">
  <DisplayName state="INSERTED">La France</DisplayName>
  <EffectiveLocale state="INSERTED">fr_FR</EffectiveLocale>
  <LocaleName state="INSERTED">fr_FR</LocaleName>
</LocaleNameData>
<LocaleNameData state="INSERTED">
  <DisplayName state="INSERTED">Les Etats-Unis</DisplayName>
  <EffectiveLocale state="INSERTED">fr_FR</EffectiveLocale>
```

```
      <LocaleName state="INSERTED">en_US</LocaleName>
    </LocaleNameData>
```

If your implementation supports other locales, follow the same pattern so that each supported locale has translations for each country name.

6. Add loading of the **LightWeightLookupList** to the minimal data load.

   a. Enter the following in a command window:

   ```
   sdk customize LightWeightLookupList.lst
   ```

   b. Open **projects\*project-name*\WEB-INF\scripts\LightWeightLookupList.lst** in a text editor.

   c. Add a line specifying the language_COUNTRY code:

   ```
   WEB-INF/xmldata/I18N/la_CO/LightWeightLookupList
   ```

   For example, to add French France:

   ```
   WEB-INF/xmldata/I18N/fr_FR/LightWeightLookupList
   ```

   d. Save and close **LightWeightLookupList.lst**.

7. Configure the SearchConfigurationProperties.xml file.

   a. Enter the following in a command window:

   ```
   sdk customize SearchConfigurationProperties.xml
   ```

   b. Open **projects\*project-name*\WEB-INF\properties\SearchConfigurationProperties.xml** in a text editor.

   c. Add the following section to the the <Locales> element. Replace ***la_CO*** with the appropriate language_COUNTRY code for the locale you are installing, for example, fr_FR for French France.

   ```
   <Locale id="la_CO" queryParserClass="com.comergent.api.appser-
   vices.search.queryParser.standard.CmgtQueryParser">

   <Analyzers>
   <Analyzer analyzerClass="com.comergent.api.appser-
   vices.search.analysis.CatalogSearchAnalyzer"
   description="CatalogAnalyzer" id="search"/>
   <Analyzer analyzerClass="com.comergent.api.appser-
   vices.search.analysis.CatalogSearchAnalyzer"
   description="CatalogAnalyzer" id="build"/>
   </Analyzers>
   <DictionaryFile file="CatalogDictionary.mappings"/>
   </Locale>
   ```

   d. Save and close the file.

8. Rebuild the project:

```
sdk merge -clean
```

9. Load the database with the new locale information:

```
sdk createDB
sdk loadDB or sdk loadMatrixDB
sdk createSegDB
sdk loadSegDB or sdk loadSegMatrixDB
```

10. Build the deployable (.war file) image:

```
sdk distWar
```

11. Deploy the .war file to your servlet container.

12. Navigate to the ***sdk_home*\dist*timestamp*-WAR** directory, where *timestamp* has the form YYYYMMDD and is the date on which you issued the sdk distWar command. Rename the **prefs_dev.xml** file to **prefs.xml**, then copy it to the home directory of the user who is running the servlet container: ***user_home*/cmgt/debs/conf/** directory.

13. Restart your servlet container.

14. Verify that the installation succeeded:

    a. Navigate to your implementation home page. The URL is similar to:

    ```
    http://<server>:<port>/Sterling/en/US/enterpriseMgr/matrix
    ```

    b. Log in as an administrator user, then click My Account.

       The User Detail page displays.

    c. The Preferred Locale drop-down list in the User Locale panel should include the locales that you just installed.

    d. Choose a locale from the drop-down list, click **Save**, log out, and log back in.

       The administrator user's home page should display with localized text. You can now create users who use the new locales.

Localization Pack Installation Steps: Existing Implementation

These steps apply to an existing implementation of the Sterling Multi-Channel Selling Solution.

Adding support for a locale to an existing implementation requires that you enter SQL commands directly to modify and populate the Knowledgebase. To complete these steps, you must have access to a SQL client such as Microsoft SQL Server Management Studio Express or SQLPlus for Oracle.

These steps assume that your implementation was installed using the SDK and that you have an existing release structure within which to work.

Before you begin, stop your servlet container instance.

1. Install the localization pack JAR file. For example:

```
sdk install SterlingSellingSuite-Locale-DEBS-8.0.1-de-DE.jar
```

   This installs the localization pack in the **releases\debs-8.0.1** directory.

2. Modify the **web.xml** file:

   a. If you have not already customized your project's **web.xml** file for other locales, enter the following in a command window:

```
sdk customize web.xml
```

   b. Open **projects\*project-name*\WEB-INF\web.xml** in a text editor.

   c. Copy the section that begins with the following:

```
 <!-- Start of English US mapping -->
```

   And ends with the following:

```
<!-- End of English US mapping -->
```

   d. Paste the entire section after the English US mapping section.

   e. Modify the new section's comments to refer to the locale name you are adding. For example:

```
<!-- Start of German Germany mapping -->
```

   f. Modify the new section’s file path references to refer to the locale name file path, such as /**de/DE**. For example:

```
<servlet-mapping>
      <servlet-name>DispatchServlet</servlet-name>
      <url-pattern>/de/DE/catalog/*</url-pattern>
```

```
</servlet-mapping>
```

3. Configure the **Internationalization.xml** file.

   a. If you have not already customized your project's **Internationalization.xml** file for other locales, enter the following in a command window:

```
sdk customize Internationalization.xml
```

   b. Open **projects\*project-name*\templates\WEB-INF\properties\Internationalization.xml** in a text editor.

   c. Add the locale designation to the Presentation element's supportedLocales field. For example, for the German Germany locale:

```
<supportedLocales controlType="text" runtimeDisplayed="true"
ChangeOnlyAtBootTime="true" visible="true" boxsize="60"
displayQuestion="Presentation Locales"
displayOptions="en_US,en_US (English-United States),
zh_TW,zh_TW (Chinese-Taiwan),fr_FR,fr_FR
(French-France),fr_BE,fr_BE (French-Belgium),
de_DE,de_DE (German-Germany)" defaultChoice="en_US"
help="Supported presentation locales.">en_US,de_DE</supportedLo-
cales>
```

   d. Add the following to the Languages element. Replace ***la*** with the language code and ***CO*** with the country code for your locale:

```
<la visible="false">
<defaultCountry controlType="text" runtimeDisplayed="true"
ChangeOnlyAtBootTime="true" visible="false" boxsize="60"
displayQuestion="URL for the Help Files" defaultChoice="US"
help="This is the default country for a specific language">CO
</defaultCountry>
</la>
```

   For example, for the German Germany locale:

```
<de visible="false">
<defaultCountry controlType="text" runtimeDisplayed="true"
ChangeOnlyAtBootTime="true" visible="false" boxsize="60"
displayQuestion="URL for the Help Files" defaultChoice="DE"
help="This is the default country for a specific language">DE
</defaultCountry>
</de>
```

4. Modify the **I18NLookup.lst** file:

   a. Enter the following in a command window:

```
sdk customize I18NLookup.lst
```

b. Open **projects\*project-name*\WEB-INF\xmldata\I18N\I18NLookup.lst** in a text editor.

c. Add a line specifying the LightWeightLookupList file to load for the locale you are adding:

```
WEB-INF/xmldata/I18N/la_CO/LightWeightLookupList
```

For example, to load the LightWeightLookupList file for German Germany:

```
WEB-INF/xmldata/I18N/de_DE/LightWeightLookupList
```

5. Configure the SearchConfigurationProperties.xml file:

a. If you have not already customized your SearchConfigurationProperties.xml file for other locales, enter the following in a command window:

```
sdk customize SearchConfigurationProperties.xml
```

b. Open **projects\*project-name*\WEB-INF\properties\ SearchConfigurationProperties.xml** in a text editor.

c. Add the following section to the the <Locales> element. Replace ***la_CO*** with the appropriate language_COUNTRY code for the locale you are installing, for example, de_DE for German Germany.

```
<Locale id="de_DE" queryParserClass="com.comergent.api.appser-
vices.search.queryParser.standard.CmgtQueryParser">

<Analyzers>
<Analyzer analyzerClass="com.comergent.api.appser-
vices.search.analysis.CatalogSearchAnalyzer"
description="CatalogAnalyzer" id="search"/>
<Analyzer analyzerClass="com.comergent.api.appser-
vices.search.analysis.CatalogSearchAnalyzer"
description="CatalogAnalyzer" id="build"/>
</Analyzers>
<DictionaryFile file="CatalogDictionary.mappings"/>
</Locale>
```

d. Save and close the file.

6. Set the location of the XML loader script and prefs.xml file:

a. Enter the following in a command window:

```
sdk customize loadI18NFromXML.bat (For Windows systems)
```

or:

```
sdk customize loadI18NFromXML.sh (For Unix systems)
```

b. Open **\projects\*project_name*\WEB-INF\scripts\loadI18NFromXML.bat** or **.sh** in a text editor.

c. Search for the line containing:

```
set LOADER_JAR=%DEBS_RELEASE_DIR%/cmgt-xmlloader-tool.jar (For
Windows systems)

LOADER_JAR=$DEBS_RELEASE_DIR/cmgt-xmlloader-tool.jar (For Unix
systems)
```

d. Replace %DEBS_RELEASE_DIR% (Windows) or $DEBS_RELEASE (Unix) with the full pathname of your project's **image/data** directory location. Ensure that the directories are separated by forward slashes. For example: **C:/SDK351/releases/debs-8.0.1/image/data** (Windows) or **/debs/sdk351/releases/debs-8.0.1/image/data** (Unix).

e. Specify the location of the prefs.xml preferences store file. Search for the line containing:

```
-DDataServices.General.ServerId=1
```

f. Add the following parameter to the JAVA_OPTS parameter list:

```
-Dcomergent.preferences.store="prefs.xml_full_pathname"
```

Where *prefs.xml_full_pathname* is the location of the **prefs.xml** file in your project. For example, if your project name is matrix and your sdk_home is **/debs/sdk351**, the command is as follows:

```
$JAVA $JAVA_OPTS -DDataServices.General.ServerId=1
-Dcomergent.preferences.store="/debs/sdk351/matrix/prefs.xml"
-classpath $CP ${MAIN} dummy persist PARTNER_NAME=matrix
$PHASE1_LIST MODE=QUIET
```

g. Save and close the file.

7. Rebuild the project:

```
sdk merge -clean
```

8. Load the database with the new locale information:

a. Use your SQL client tool to connect to the existing knowlegebase. For connection information, consult the values in the **projects\*project_name*\\ *project_name*-dev.properties** file.

b. Update the CMGT_LOCALE table entry. In your SQL client tool, run the following SQL command:

```
INSERT INTO CMGT_LOCALE
(LOCALE_KEY,LOCALE_NAME,LOCALE_DESCRIPTION,ACTIVE_FLAG,DB_SORT_LO
CALE_NAME )
VALUES ( key,'la_CO','Country Language','Y','BINARY')
```

- *Key* is the locale key. The locale key must be a unique numeric value.
- *la_CO* is the language code and country code. Use the appropriate language_COUNTRY encoding for the locale you are installing, for example, de_DE for German Germany, fr_FR for French France, jp_JP for Japanese Japan, and so on.

  For example, the following INSERT statement is for supporting the de_DE (German Germany) locale:

```
INSERT INTO CMGT_LOCALE
(LOCALE_KEY,LOCALE_NAME,LOCALE_DESCRIPTION,ACTIVE_FLAG,DB_SORT_LO
CALE_NAME )
VALUES ( 3,'de_DE','Germany German','Y','BINARY')
```

To check that your INSERT into the CMGT_LOCALE table is correct, run the following SQL statement:

```
select * from CMGT_LOCALE
```

9. Add the country name translations:

a. Update the CMGT_LOCALE_NAME Table. In your SQL client tool, run the following SQL commands:

```
insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
DISPLAY_NAME, ACTIVE_FLAG) values ('la_CO', 'la_CO',
'Locale_Country_Name', 'Y')
insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
DISPLAY_NAME, ACTIVE_FLAG) values ('la_CO', 'defaultLA_defaultCO',
'default_Country_Name', 'Y')
insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
DISPLAY_NAME, ACTIVE_FLAG) values ('defaultLA_defaultCO', 'la_CO',
'Country', 'Y')
```

Where:

- *la_CO* is the language_COUNTRY combination for the locale you are adding, such as de_DE
- *Locale_Country_Name* is the name of the country in the locale's language, such as Deutschland
- *defaultLA_defaultCO* is the language_COUNTRY combination for the default locale, such as en_US
- *default_Country_Name* is the name of the default country in the language of the locale that you are adding, such as Vereinigte Staaten
- *Country* is the name of the country in the default locale's language, such as Germany

For example, to add country name translations for German:

```
insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
DISPLAY_NAME, ACTIVE_FLAG) values ('de_DE', 'de_DE', 'Deut-
schland', 'Y')
insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
DISPLAY_NAME, ACTIVE_FLAG) values ('de_DE', 'en_US', 'Vereinigte
Staaten', 'Y')
insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
DISPLAY_NAME, ACTIVE_FLAG) values ('en_US', 'de_DE', 'Germany',
'Y')
```

b. If there are several locales installed on your system, add a country name translation for the new locale to each of the existing locales, and add a country name translation for each existing locale to the new locale. For example, if you already support the fr_FR locale and are adding support for the de_DE locale, run the following SQL:

```
insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
DISPLAY_NAME, ACTIVE_FLAG) values ('fr_FR', 'de_DE', 'L''Alle-
magne', 'Y')
insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
DISPLAY_NAME, ACTIVE_FLAG) values ('de_DE', 'fr_FR', 'Frankreich',
'Y')
```

To check the results of the INSERT commands, run the following SQL command:

```
select * from CMGT_LOCALE_NAMES
```

10. In a command window, navigate to your ***sdk_home*\workspaces*project-name*** directory and run the following command:

    ```
    WEB-INF\scripts\loadI18NFromXML.bat I18N jdbc_driver (for Windows
    Systems)
    ```

    or

    ```
    WEB-INF\scripts\loadI18NFromXML.sh I18N jdbc_driver (for Unix Sys-
    tems)
    ```

    where ***jdbc_driver*** is the full pathname of your JDBC jar file.

    For example:

    ```
    WEB-INF\scripts\loadI18NFromXML.bat I18N oraclejdbc.jar
    ```

11. In a command window, navigate to your ***sdk_home*** directory and build the deployable (.war file) image:

    ```
    sdk distWar
    ```

12. Deploy the .war file to your servlet container.
13. Navigate to the **projects\*project_name*\dist\timestamp-WAR** directory, where *timestamp* has the form YYYYMMDD and is the date on which you issued the sdk distWar command. Rename the **prefs_dev.xml** file to **prefs.xml**, then copy it to the home directory of the user who is running the servlet container: ***user_home*/cmgt/debs/conf/** directory.
14. Restart your servlet container.
15. Verify that the installation succeeded:
    a. Navigate to your implementation home page. The URL is similar to:

    ```
    http://<server>:<port>/Sterling/en/US/enterpriseMgr/matrix
    ```

    b. Log in as an administrator user, then click My Account.

    The User Detail page displays.

    c. The Preferred Locale drop-down list in the User Locale panel should include the locales that you just installed.

d. Choose a locale from the drop-down list, click **Save**, log out, and log back in.

   The administrator user's home page displays with localized text. You can now create users who use the new locales.

CHAPTER 12

General Security Considerations

This chapter covers:

General Architectural Concerns

When you design your implementation environment, you should bear in mind the physical and network configuration of your data center, and your security policies to determine what people can perform what activities. In particular, you must distinguish carefully between what a person can do as an administrator in your data center environment, and what a person can do as a Sterling Multi-Channel Selling Solution user.

Administration Model

This section describes the entities assumed to be present in the administrative domain (the data center) in which the Sterling Multi-Channel Selling Solution resides, including networks, servers, and administrative roles. This is likely not an exhaustive list. It is likely that various network devices will exist within this environment, and perhaps other servers.

Networks

The following network zones are assumed to exist. These networks are connected to themselves as outlined below through gateways.

- External network: Directly visible from the internet. It hosts the Web servers and static content. The External network is accessible to the internet through a firewall. It is assumed that the firewall and appropriate standard security practices are sufficient to prevent shell level access from the internet. The External network has a gateway to the De-Militarized Zone (DMZ) that permits highly controlled access from the Web server to the application server(s).
- DMZ: This network is not directly visible from the Internet. A constrained gateway permits the Web server(s) residing on the External network to access the Application server(s) residing on this network and another, similar, gateway permits access from the DMZ network to the Internal network. The Web server routes messages to application servers through a dedicated port.
- Internal network: Not visible from the Internet, nor from the External network. Database resources reside here. Application servers in the DMZ connect to Database servers in this network through a constrained gateway.

Servers

The following servers are assumed to exist. The term server here indicates a software application that is more or less continuously listening on one or more network ports responding to requests received on the ports. Software servers, of course reside on computer hardware. Generally, though not necessarily, there will be a one-to-one relationship between a server software system, and a server hardware entity.

- Web server resides in the External network. It responds to HTTP (possibly using SSL) requests from the Internet or internal corporate Intranets.

- Application server resides in DMZ. Some http and https requests are delegated to the Application server for dynamically generated response. The Application server maintains connections with the Database server.
- Database server resides in the Internal network.

Roles

This section describes roles within the administrative context of the Sterling Multi-Channel Selling Solution. These are roles assigned to data center personnel acting as employees or agents of the Enterprise. They are distinguished from the roles of individuals who interact directly with the Sterling Multi-Channel Selling Solution ("Online Users"). Online users have capabilities managed directly by three Sterling Multi-Channel Selling Solution Entitlement services. Dispatch (or "MessageType") Entitlement Service manages page flow privileges. The Access Policy Service and ACL Service together manage fine-grained data-level access.

- Database Administrator
    - Responsible for Database servers.
    - Can log into database server.
    - Can read, create, update, or delete databases, database tables, indexes, and other database resources.
    - Can create backups and restore from backups.
    - Can create Database users and manage them.
    - Does not have root level authority in server operating system.
    - Does not have direct access to Application server machine (or DMZ).
    - Does not have access as Sterling Multi-Channel Selling Solution user.
- System Administrator
    - Responsible for server hardware, and server software.
    - Has root access to server machines within his/her zone of responsibility.
    - Has the authority to start and stop server processes.
    - As root, can read, write, update, or delete files in file systems.
    - Can back up and restore files.
    - Can create operating system level users and manage them.
    - Does not have access to log in to database server.

    - Does not have access as Sterling Multi-Channel Selling Solution user.
- Developer
    - Responsible for preparation of deployment Web archives (WAR files).
    - Has the authority to create Web archives representing the Sterling Multi-Channel Selling Solution executable.
    - Can set properties and business rules governing Sterling Multi-Channel Selling Solution operation, including properties that configure access to the database, properties that configure the JCE Key store, and so on.
    - Has the authority to create or modify the initial Sterling Multi-Channel Selling Solution dataset. This dataset is a part of the deployment archive.
    - Does not have any kind of access to the Production Database server or Application servers.
    - Does not have access to the production Sterling Multi-Channel Selling Solution as a Sterling Multi-Channel Selling Solution user.
    - Does not move code from development and QA environments to production.
- Network Administrator
    - Configures and manages network.
    - Has authority to create and assign network resources, including domain names, IP addresses, firewall policies, and so on.
    - Does not have Database server access.
    - Does not have access as Sterling Multi-Channel Selling Solution user.

Data Center Roles

The following are assumed about data center administrative roles:

1. Roles are segregated. System Administrators cannot be Developers, Network Administrators, nor Database Administrators. Similarly, Database Administrators cannot be Developers, System Administrators, nor Network Administrators, and so on.
2. System Administrator Roles should be partitioned on network boundaries. A system administrator for the DMZ should not be a system administrator for the Internal network.

3. Data center administrators do not have Sterling Multi-Channel Selling Solution userids with administration roles.

Securing Users

When you load either the minimal or reference data set into the Sterling Multi-Channel Selling Solution, you create two enterprise users: admin and ERPAdmin. Before permitting the Sterling Multi-Channel Selling Solution to go live in production, you must change the passwords of both users. If you do not do this, then it represents a serious security hole in your application.

You can change the passwords by logging in after the Sterling Multi-Channel Selling Solution is started using the administration interface. See the *Sterling Multi-Channel Selling Solution Administration Guide* for more information.

SSL support

The Sterling Multi-Channel Selling Solution supports communication using the SSL protocol between a user's browser and the Sterling Multi-Channel Selling Solution. In particular, as part of your implementation of the Sterling Multi-Channel Selling Solution, you must consider which pages (if any) should be SSL-protected.

This section discusses SSL support in the Sterling Multi-Channel Selling Solution. See the Best Practices Guide for information about ensuring the security of user sessions in Java servlet-based applications using HTTPS.

If you are not using SSL in your implementation of the Sterling Multi-Channel Selling Solution, then you do not need to change the out-of-the-box port settings for non-SSL and SSL ports. If you are using SSL to protect some or all access to the system, set port-related properties in the **Comergent.xml** template file, located in **templates/WEB-INF/Comergent.xml.** Use the following guidelines to set the port-related properties:

- If you are using the standard ports (80 for non-SSL access and 443 for SSL access, respectively) for both schemes, then set the port settings to "." for both.
- If you are using non-standard ports for one or the other scheme, then you must explicitly set the port number for each.

Setting Up Secure Message Types

You can choose whether a group of message types require SSL access or not. A particular web site can have mixed secure and non-secure access requirements. The access requirements follow the type of request received by the JSP page. If only a section of a web site requires SSL, you must set the relevant message type to require SSL/HTTPS. You must also specify where users revert to non-SSL/HTTPS access.

When a user clicks a link on a Sterling web application JSP page, the JSP page looks up the link in a repository to see if the link requires SSL. If it does, the JSP page generates the link as "https://" and SSL handles all the necessary encoding.. Otherwise, the JSP page generates the link as "http".

You can ensure that pages require SSL access by setting their associated message types to a security level that requires “https”. When a link is generated using the *link()* methods provided by the Sterling Multi-Channel Selling Solution, then the command parameter is used to identify the message type, and the SecurityLevel child element is used to ensure that the appropriate schema (http or https) is reflected in the URL.

The SecurityLevel element can be used at any level in the message type group and type hierarchy. Specifying it at a message type group level means that all message types that belong to the group inherit the security level unless it is overwritten at a lower level in the hierarchy.

The Level attribute of the SecurityLevel element can take the following values:

- any: both http and https can be used to access the message type. This is the default value if nothing is specified.
- useHttp: can be accessed by http and subsequent URLs will be generated with http.
- useHttps: the page may be accessed without https, but any URL generated by the Sterling Multi-Channel Selling Solution will specify https.
- requireHttps: any URL requesting this message type must use https, otherwise the request is rejected and an error page is displayed. Ensure that wherever this message type is used, it is used in the *link()* method to form the link to the message type, especially in any forms which use this message type.

For example, suppose that the following message type is declared in **MessageTypes.xml**:

```
<MessageType Name="SysUserDetailDisplay">
    <SecurityLevel Level="requireHttps" />
    ...
</MessageType>
```

Then link("partnerMkt", "SysUserDetailDisplay") will generate the URL:

```
https://<server>:<serverSSLPort>/Sterling/partnerMkt/
matrix?cmd=SysUserDetailDisplay
```

By default, that is if no SecurityLevel is specified for a message type or for any group to which the message type belongs, the *link()* methods will generate URLs using the same protocol used to access the referring page.

You can use the SecurityLevel element to specify that https is required to access a given page by setting the security level to “requireHttps”: the Sterling Multi-Channel Selling Solution verifies that each message type is being accessed using the appropriate protocol.

You must set the C3_Commerce_Manager.General.ServerSSLPort element in the **Comergent.xml** file to the appropriate value for your servlet container. If the servlet container is set up to use the standard SSL port (443), then you do not have to specify it. Consult your servlet container documentation for any steps that are required to set up a port to accept SSL connections.

Example Usages

In these examples, we suggest message groups and types that might be candidates to protect using the SSL protocol. In general, you need to determine the possible page flows that users can perform and identify the entry and exit message types that surround the area to be SSL-protected.

Protecting the Authenticated Environment

You should consider protecting the entire Web experience of the Sterling Multi-Channel Selling Solution presented to authenticated users. You can do this by adding:

```
<SecurityLevel Level="requireHttps" />
```

to the EnterpriseHomeGroup and PartnerHomeGroup message groups.

Protecting the Enterprise Environment

Suppose that you want to protect your enterprise administration pages behind the https schema. You can add:

```
<SecurityLevel Level="requireHttps" />
```

to the following message types:

- LoginDisplay
- GenericLoginDisplay
- HomePageDisplay

Protecting Credit Card Information

Users are highly sensitive to passing credit card information over the Web, and you may be required to ensure that any requests that include credit card information are SSL-protected. For example, users may enter credit card information when they edit an order header. The URL used to submit the order header information uses the OILAddrChangeProcess message type. By adding:

```
<SecurityLevel Level="useHttps" />
```

to the OILDisplayGroup message group, you ensure that when URLs are formed with message types from this group that they specify the https schema.

Protecting User Administration Pages

You should consider using the SSL protocol to protect pages in which users enter personal information. For example, add:

```
<SecurityLevel Level="requireHttps" />
```

to the UserAdminGroup message group.

Installing Certificates for SSL

To support SSL communication, you must ensure that you have determined the level of security you want to support between users and your Sterling Multi-Channel Selling Solution and between the Sterling Multi-Channel Selling Solution and any of your partners' enterprise servers. You can enable SSL communication between users' browsers and the enterprise server and between the enterprise server and one or more of your partners' enterprise servers.

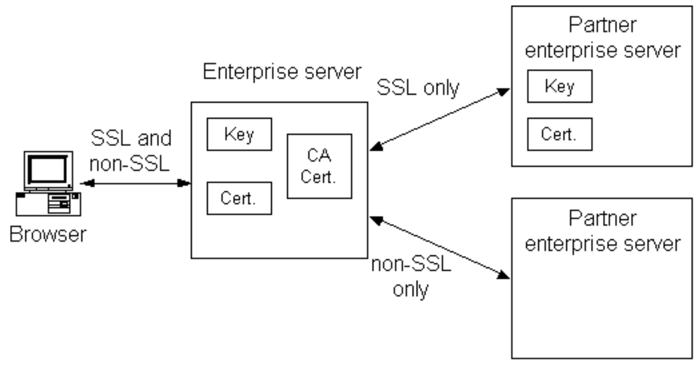


FIGURE 10. SSL Communication in the Sterling Multi-Channel Selling Solution

Overview

When two systems elect to use the SSL protocol to communicate, the entity that initiates the communication is referred to as the client and the other entity is the server. For example, if the enterprise server is set up to support SSL communication with users' browsers, then the browser acts as the SSL client and the enterprise server is the SSL server. If the enterprise server and a partner's enterprise server use SSL to transfer price and availability requests, then the enterprise server is acting as the SSL client and the partner's enterprise server is the SSL server.

The SSL protocol requires that the SSL client maintains certifying authority certificates from certificate authorities from whom they accept certificates. When a client attempts to open an SSL communication session with an SSL server, the server must send to the client its certificate and key. The client may choose to verify against the certifying authority certificate, and then uses the key to encrypt messages sent back to the server.

A client makes a request to open an SSL communication by posting to the server using the https schema: that is, by using a URL of the form

```
https://myserver.com:<SSL port>
```

Consequently:

- If you want your installation of the enterprise server to support SSL communication from Web browsers, then you must obtain a server certificate and server key and use the appropriate servlet container mechanism to specify an SSL port.

- If you want to enable SSL communication between an enterprise server and a partner's enterprise server, then the partner's enterprise server must obtain a server certificate and server key. The partner's enterprise server must be configured to receive messages from your enterprise server on an SSL port. Your enterprise server must be configured to accept the SSL certificates offered by the partner server. In general, if the partner server's certificates match the domain of the partner server's URL, then the enterprise server accepts them automatically.

Storing Data in Encrypted Form

This section covers:

- "General Setup" on page 201
- "Changing Encryption Algorithms" on page 205
- "Key Stores and System Initialization" on page 206
- "Wrapper Classes for Standard Algorithms" on page 208
- "Key Rotation" on page 208
- "Release 6.0 and Earlier Releases" on page 212

The Sterling Multi-Channel Selling Solution lets you store sensitive business data in the Knowledgebase in an encrypted form. This is done by setting the Encryption attribute of the corresponding DataElement to "1-way" or "2-way". See the *Sterling Multi-Channel Selling Solution Reference Guide* for more information.

Attention: If you deploy the reference implementation of the Sterling Multi-Channel Selling Solution without making changes to the schema, then note the following:

Credit card numbers stored as part of a user's profile are 2-way encrypted. This means that a **dcmsKey.ser** file is created on your system.

If you upgrade the Java Virtual Machine (JVM) at any time after creating encrypted data, then you should check that the data can be retrieved correctly. For example, if upgrading the JVM means that the **.jceKeystore** file is regenerated, then 2-way encrypted data will have to be recovered.

Securing data in the persistent storage system should be considered when the data travels over insecure internal or external networks or when access to the database server cannot be restricted to authorized individuals. In general, it is preferable to secure data using the facilities provided by the database server. These are likely to provide a more comprehensive and higher performance solution.

You must decide which fields are to be encrypted before loading the data and creating more data. In general, you cannot change to using encrypted data for a data field after any data objects have been created of that type.

Attention: You cannot encrypt data that is used in ***Sterling***Analyzer reports without breaking the reports that use the data. You can identify which database columns are used in reports by reviewing the view creation scripts. All report data is accessed using views and so the view scripts provide a complete list of the columns accessed by reports.

General Setup

This section describes the basic steps to implement data encryption.

Note: Release 6.3 introduced more flexible encryption support than previous releases. If you are working on an earlier release, see "Release 6.0 and Earlier Releases" on page 212.

You should use JDK 1.5 or a subsequent compatible version because this has the Java Cryptography Extension (JCE) built in.

1. You should download the Unlimited Strength Jurisdiction Policy Files 1.5 available from the SUN Java Web site. Follow the instructions provided by the **Readme.txt** file to install the JCE jurisdiction policy JAR files into your Java environment.
2. To encrypt data that is to be stored in the Knowledgebase, you must specify the encryption method that is to be used:
   - For two-way encrypted fields this must be a symmetric encryption algorithm so that the data can be retrieved in its unencrypted form.
   - For one-way encrypted fields, you must use a digester. This must effectively provide a highly probable guarantee that if two source strings are digested and the digested strings are the same, then the source strings must have been the same to start with.

At any one time, the data services layer determines that only one active symmetric encryption algorithm and only one active digester can be in use.

- You can switch from one symmetric encryption algorithm to another as your encryption needs change. Data encrypted using an earlier encryption algorithm can be retrieved, and if it is re-saved, then it is persisted using the appropriate active symmetric encryption algorithm. See "Changing Encryption Algorithms" on page 205 for more information.

- Once you have selected your digester, then this cannot be changed. By its nature, data encrypted using a digester cannot be retrieved in order to re-encrypt it using a different digester.

You cannot change the status of data fields from encrypted to unencrypted or the other way round. In summary:

1. Decide which data fields are to be encrypted. You cannot add or remove fields from this list once it is set up and data objects have been persisted using the encryption methods.
2. Decide which of these fields are to be one-way encrypted and which are to be two-way encrypted.
3. Select a secure digester to be used for one-way encryption. You must keep this digester.
4. Select a secure symmetric encryption algorithm: you can change this later if your encryption needs change.

Symmetric Encrypter

You declare the active symmetric encryption algorithm using the TwoWayEncrypter element in the **DataServices.xml** file. For example:

```
<TwoWayEncrypter>DefaultEncrypter</TwoWayEncrypter>
```

The value of this element must match the Name attribute of an Alias element or a SymmetricEncrypter element declared in the **CryptographyService.xml** configuration file. For example:

```
<Alias Name="DefaultEncrypter" OriginalName="InlineDES">
    <Description>Alias to the default encrypter.</Description>
</Alias>
```

The OriginalName attribute points to the SymmetricEncrypter element that defines the encrypter:

```
<SymmetricEncrypter Name="InlineDES"
    Class="com.comergent.cryptography.JCESymmetricEncrypter"
    KeyManager="InlineKeyManager" KeyName="myDesKey" Tag="IDES">
    <Description>
        DES Encrypter using key from inline key store.
    </Description>
    <Algorithm Name="DES" />
</SymmetricEncrypter>
```

Alternatively, you can reference a symmetric encrypter directly by its name. For example:

```
<SymmetricEncrypter Name="JCE DES" Tag="DES"
    Class="com.comergent.dcm.cryptography.JCESymmetricEncrypter"
    KeyManager="JCEKeyManager" KeyName="myKey">
    <Algorithm Name="DES" Provider="SunJCE"/>
</SymmetricEncrypter>
```

Each SymmetricEncrypter element declared in the **CryptographyService.xml** file can be used to encrypt and decrypt data.

- The Name attribute in the element is used to identify the symmetric encrypter in the TwoWayEncrypter element. The Name attribute must be unique among all the SymmetricEncrypter and Digester elements declared.
- The Tag attribute of the SymmetricEncrypter element is used to prefix the encrypted strings in the persistent data store. The Tag attribute must be unique among all the SymmetricEncrypter elements declared, but the same Tag value can be used for a Digester. For example, if the string “ajones” is encrypted to “hg$y&7606(7gfj” by the symmetric encrypyter whose tag is “DES”, then the value stored in the database is “DES:hg$y&7606(7gfj”. In this way, each stored encrypted value provides an indication of which symmetric encrypter can be used to decrypt it.
- The Class attribute of each SymmetricEncrypter element specifies the class to be used to perform the encryption and decryption: this class and its dependent classes must be in the Sterling Multi-Channel Selling Solution classpath. The specified class must implement the com.comergent.api.dcm.cryptography.SymmetricEncrypter interface.
- If the symmetric encryption algorithm requires a key manager, then the SymmetricEncrypter element also specifies its key manager using the KeyManager attribute. The value of this element must match the Name attribute of one of the declared KeyManager elements.

  ```
  <KeyManager Name="JCEKeyManager"
      Class="com.comergent.dcm.cryptography.JCEKeyManager">
      <Algorithm Name="DES"/>
  </KeyManager>
  ```

Each KeyManager element in the **CryptographyService.xml** configuration file declares the class to be used to manage keys for a symmetric encrypter. Typically, these classes are used to access keys managed in a key store. The name of the key to be retrieved from the key store is specified by the KeyName attribute of the SymmetricEncrypter element. Thus, by having two SymmetricEncrypter elements declaring the same Class attribute, but different KeyName attributes, you can use different keys to encrypt data. See "Key Stores and System Initialization" on page 206 for more information about key stores and how keys can be retrieved when the Sterling Multi-Channel Selling Solution is starting up.

Digester

You declare the active digester using the OneWayEncrypter element in the **DataServices.xml** file. For example:

```
<OneWayEncrypter>MD5</OneWayEncrypter>
```

The value of this element must match the Tag attribute of a Digester element declared in the **CryptographyService.xml** configuration file.

Each Digester element declared in the **CryptographyService.xml** file can be used to encrypt data.

- The Name attribute in the element is used to identify the digester in the OneWayEncrypter element.
- The Tag attribute is used to prefix the encrypted strings in the persistent data store. The Tag attribute must be unique among all the Digester elements declared, but the same Tag value can be used for a SymmetricEncrypter element. For example, if the string “ajones” is encrypted to “Ta$y&%lN7gL5” by the digester whose tag is “MD5”, then the value stored in the database is “MD5:Ta$y&%lN7gL5”. In this way, each stored encrypted value provides an indication of which digester was used to encrypt it.
- The Class attribute of each Digester element specifies the class to be used to perform the encryption: this class and its dependent classes must be in the Sterling Multi-Channel Selling Solution classpath. The specified class must implement the com.comergent.api.dcm.cryptography.Digester interface.

Default Symmetric Encrypter and Digester

By default, Release 7.0 and higher of the Sterling Multi-Channel Selling Solution uses the one-way and two-way encryption schemes used in the Sun JCE implementation: these reference MD5 and DES respectively. Earlier releases used the legacy encryption schemes provided by the crysec packages. See "Release 6.0

and Earlier Releases" on page 212 for further information. These schemes are identified by empty Tag attributes (that is, Tag="").

Note:	You should consider replacing the legacy digester with either SHA or MD5 digesters. Both offer a higher level of security against cryptographic attack. However, you must make the decision to change to a different digester before implementing the Sterling Multi-Channel Selling Solution. See "One-Way Encrypted Data" on page 206.

Changing Encryption Algorithms

You can change the symmetric encryption algorithm used if your encryption needs change. If you do this, data that has previously been encrypted using the earlier symmetric encryption algorithm is not lost.

Note:	You must use JDK 1.5 or a subsequent compatible version to use AES encryption.

Two-Way Encrypted Data

Suppose that a data field of a data object is marked for two-way encryption and the active symmetric encryption scheme is set to "DES". If a new data object is persisted, then the data field value is set to something like "DES:hfd8kUH9*". Suppose that you decide to switch to using the symmetric encryption algorithm identified by the Tag value "AES", and so you modify the **CryptographyService.xml** file to declare a new default encrypter. Specifically, modify the Alias element whose Name attribute is DefaultEncrypter so that the OriginalName attribute is set to "JCE_AES". For example:

```
<Alias Name="DefaultEncrypter" OriginalName="JCE_AES">
   <Description>Alias to the default encrypter.</Description>
```

If new data objects are persisted, then their data is encrypted using the AES symmetric encryption scheme. If the earlier data object is restored, then the encryption service recognizes that the field was encrypted using the DES encryption schema, and invokes the corresponding symmetric encrypter class to decrypt it. If the restored data object is subsequently persisted, then the AES scheme is used to perform the encryption and the value of the data field will be something like "AES:8(HH$DygK" in the persistent data store.

You can also perform "key rotation" to update all your encrypted data to make use of a new key. See "Key Rotation" on page 208 for more details.

One-Way Encrypted Data

Data that is encrypted using a digester is not intended to be used to retrieve the original value. You cannot easily change digesters once data has been encrypted using your choice of digester.

For example, suppose that you choose to one-way encrypt user passwords in the Password field of the UserContact data object, and suppose that you have chosen to use the legacy digester for this purpose. If subsequently you decide to change digesters to the more secure SHA or MD5 digesters, then you would have to proceed along these lines:

1. Notify users that their passwords will be changed at a certain date.
2. At that date, stop the Sterling Multi-Channel Selling Solution, and set the OneWayEncrypter element to the new Digester name. Suppose that the Tag for the new digester is “SHA”.
3. Generate a new password value for each user: say, their username and a randomly selected integer: for example, “ajones67854”.
4. Offline, use the new digester to encrypt the new password for each user: suppose that “ajones67854” is encrypted to “hjkYF*&5NF0”.
5. Using a SQL script, update the CMGT_USER_CONTACT table to enter the encrypted form of their password for each user:

   ```
   UPDATE CMGT_USER_CONTACTS SET PASSWORD = 'SHA:hjkYF*&5NF0' WHERE
   USER_NAME = 'ajones';
   ```
6. Restart the Sterling Multi-Channel Selling Solution.
7. Notify each user by email that their password has now changed, and give them the new unencrypted value. Ask each user to log in using their new password, and ask them to change their password immediately.

Note that any other data in the Sterling Multi-Channel Selling Solution that was also one-way encrypted (such as credit card numbers) would be rendered inaccessible and would have to be re-created if required.

Key Stores and System Initialization

Almost all encryption schemes use a key store to hold the keys used to encrypt and decrypt data. In symmetric schemes the same keys are used to both encrypt and decrypt data. Consequently, it is important that you take great care to protect your key stores and ensure that they are not corrupted or deleted.

Attention:	Loss or corruption of your key store can lead to complete loss of your encrypted data.

Each encryption scheme make use of different types of keys and key stores, and you must consult the documentation that comes with your choice of encryption scheme carefully. Typically, the process is to create a key store, and then generate keys that you add to the key store. Each key has a name that is used to retrieve the key from the key store.

When the Sterling Multi-Channel Selling Solution is started or re-started, it must retrieve the appropriate keys from a key store. If it fails to do so, then the Sterling Multi-Channel Selling Solution fails to initialize and will not permit any logins.

Key stores and keys in the key store can be encrypted. If you choose to encrypt either, then as part of the initialization process, the cryptography service must decrypt them to retrieve the keys for the symmetric encrypters.

Attention:	Take extreme care in encrypting key stores. A key store is effectively impossible to decrypt without the appropriate passwords. Data will be impossible to retrieve if the keys in the key store are inaccessible.

The cryptography service is initialized during the initialization of the InitServlet class. It attempts to decrypt the key stores and keys for each symmetric encrypter using the password “Comergent”. If it fails for one or more of the symmetric encrypters, then the initialization of the Sterling Multi-Channel Selling Solution stops at this point. Requests posted to the DispatchServlet (the main servlet class used to process requests), are sent a 503 response: Service Unavailable.

You can complete initialization of the cryptography service and hence of the whole Sterling Multi-Channel Selling Solution by posting a request to the InitServlet that includes the relevant parameters: typically, for each symmetric encrypter the passwords used to encrypt the key store and the key. For example if one symmetric encryption scheme requires a keyStorePassword parameter and a keyPassword parameter, and a second scheme used just a storePassword parameter, then the following post would provide the initialization information:

```
http://<machine:port>/Sterling/init?keyStorePassword=password&key-
Password=password&storePassword=password
```

Wrapper Classes for Standard Algorithms

SymmetricEncrypter Class

The JDK 1.5 provides implementations of standard symmetric encryption algorithms such as AES and DES. The Sterling Multi-Channel Selling Solution enables you to use these through the JCESymmetricEncrypter class. This class implements the SymmetricEncrypter interface and so may be specified in the SymmetricEncrypter element.

Digester Class

The JDK 1.5 provides implementations of standard digester algorithms such as MD5 and SHA. The Sterling Multi-Channel Selling Solution enables you to use these through the JCEDigester class. This class implements the Digester interface and so may be specified in the Digester element.

Key Rotation

Key Rotation Procedure

This procedure describes how to rotate encryption keys that protect data on the Sterling Multi-Channel Selling Solution. The procedure is designed so that it can be executed as needed or incorporated as an operating system level cron job. It must be executed from a shell on one of operation systems because it references files located on these systems.

Background

The Sterling Multi-Channel Selling Solution can be configured to efficiently encrypt selected persistent data using any of a variety of Symmetric Encryption algorithms. The purpose of this feature is to protect confidential data from internal users who may need access to the Sterling Multi-Channel Selling Solution Knowledgebase.

The encryption keys are stored in a password-protected keystore. A keystore is usually a file in the filesystem of the application server. The identical keystore must be present on all members of a clustered system. This can be arranged by placing the keystore on a shared filesystem or by copying the keystore by hand whenever it changes.

A Sterling Multi-Channel Selling Solution configuration file maintains a mapping between keystore keys and internal logical “encrypters”. This file, named, **CryptographyService.xml**, is colocated with preferences files, usually in

***user_home*/cmgt/debs/conf/**. Again, it must be identical among members of a cluster.

The Sterling Multi-Channel Selling Solution tags encrypted values. By this means it knows which encrypter encrypted a particular value.

Why Rotate?

The Sterling Multi-Channel Selling Solution is designed to use a standard encrypter using an internal keystore during development and deployment. This standard encrypter encrypts using 56-bit DES encryption, which is adequate for those purposes. It is not, however recommended for a production system for two reasons: the keystores on the production systems should not be shared with development systems, because this compromises the data protected by the keys, and production systems should have stronger encryption.

Many customers may want to rotate keys at regular intervals, to limit the window of vulnerability from any compromise. If keys are rotated on a monthly bases, for example, then the payoff from acquiring a key is limited to that month.

Basic Process

The basic key rotation process has three steps:

1. Create a new encrypter by cloning an existing encrypter.
2. Change the default encrypter reference to point to the new encrypter.
3. Update values stored in the database to use the new encrypter. This step may take some time, depending on the number of encrypted values in the database, but can be accomplished over time, since any old values will continue to be decryptable with the old encrypter and new values will employ the new encrypter.

Detailed Process

Begin by creating a new key.

1. Log in as the user running the servlet container.
2. Open a shell window and change directory to the home directory. We refer to this directory as ***user_home***.
3. You must select a new unique name for the new encrypter and key, and a new tag. These can all be the same. The tag should be short. We suggest these incorporate temporal information or sequence number. For example, “AESyymm”, where “yy” is the year and “mm” the month. You must also

determine the encryption algorithm: AES or DESede ("triple DES") are standard for business applications.

4. Locate the tool **cmgt-cryptography-tool.jar** in the file system. If you have installed Release 8.0 into the SDK, the JAR fileis in ***sdk_home*/releases/debs-8.0/image/install/**.
5. Copy this JAR file to ***user_home***.
6. The base configuration provides a standard selection of encrypters that are likely to meet most needs. Therefore, creating a new encrypter means cloning an existing encrypter in most cases. To see the configured encrypters for your system, execute:

   ```
   java -jar cmgt-cryptography-tool.jar list
   ```
7. To create a new encrypter, execute, for example:

   ```
   java -jar cmgt-cryptography-tool.jar clone-encrypter JCE_AES
   AES0805
   ```
   The above command will clone the JCE_AES encrypter and name it AES0805. The tag and key names will also be set to AES0805 in this example, but if a different tag or key name is desired it can be set on the command line.
8. You can verify the new encrypter by executing:

   ```
   java -jar cmgt-cryptography-tool.jar info AES0805
   ```
   You will see something like this:
   ```
   AES0805: Name=AES0805 Tag=AES0805 initialized=true algorithm=AES
   keymanager=JCEKeyManager keyname=AES0805 provider=SunJCE
   ```

Changing the Default Encrypter

9. To change the default encrypter, we need set the alias to the new encrypter:

   ```
   java -jar cmgt-cryptography-tool.jar alias DefaultEncrypter
   AES0805
   ```
   If you are working with a clustered configuration with no shared filesystem, then the keystore and configuration files must be manually synchronized.
10. To do this, you must first activate the key. Do this by encrypting something:

    ```
    echo "hello" | java -jar cmgt-cryptography-tool.jar encrypt
    ```
11. You must copy the keystore and configuration file to each of the cluster members. The cryptography service configuration file is colocated with Sterling preferences and is named **CryptographyService.xml**. By default, this

is in ***user_home*/cmgt/debs/conf/**, where ***user_home*** is the home directory of the user that starts the Sterling Multi-Channel Selling Solution Web application. The location of the keystore can be identified with the following command:

```
java -jar cmgt-cryptography-tool.jar info JCEKeyManager
```

Note that the location of the keystore may be specified by using the KeyStorePath attribute of the appropriate KeyManager element in the **CryptographyService.xml** configuration file. For example:

```
<KeyManager Class="com.comergent.cryptography.JCEKeyManager"
    Name="JCEKeyManager" KeyStorePath="cmgt/debs/conf/.keyStore">
</KeyManager>
```

The path may be specified as a path relative to the ***user_home*** directory or as an absolute path.

12. In order for this change to take effect it is currently necessary to cycle the servlet container. Use techniques appropriate to your servlet container to restart it.

Updating Existing Database Ciphers.

At this point, all new values for encrypted columns in the database will be encrypted with the new encrypter, but values stored with the old encrypter are unmodified. We need to update these values. In particular, passwords protected by encryption (Encryption="2-Way" in the schema file) will no longer work.

13. First identify all primary beans that have encrypted properties. This can be accomplished by identifying Elements in DsDataElements with Encryption="2-Way", then searching the data object schema files for <DataElement> references to those elements. More than one bean may refer to the same encrypted column. For example, the UserContactBean and the UserBean both refer to the CMGT_USER_CONTACTS table and the PASSWORD column within it. In this case, we obviously need only update one of these bean types.
14. Locate the cipher-update script in your installation: **cipher-update.sh** or **cipher-update.bat**, depending on your shell. If you have installed Release 8.0 into the SDK and built your project, then you will find the script in ***sdk_home*/workspaces/*project*/WEB-INF/scripts/**.
15. Copy the script to the appropriate runtime location:
    a. If you are working in the SDK, then copy the script to ***sdk_home*/builds/*project*/**.

   b. If you are working in a running deployment environment, then copy the script to ***debs_home*/Sterling/**.

16. For each primary bean that has encrypted properties, execute the cipher-update script. You must provide the name of the bean and the name of the key field used to retrieve each instance of the bean. For example:

    ```
    cypher-update OrderBean ShoppingCartKey
    ```

    or

    ```
    cypher-update UserContactBean Userkey
    ```

    This may take a long time if the database is very large, so this step is best executed during a period of low Web activity.

Release 6.0 and Earlier Releases

To perform encryption, the Sterling Multi-Channel Selling Solution makes use of an encryption algorithm provided by the crysec Java package. The algorithm uses a random key to seed the encryption. The package looks for a file called **dcmsKey.ser** in the home directory of the user being used to run the servlet container. If the file exists, then the value it contains is used as the seed. If the file does not exist, then the algorithm creates a random seed and writes it to a new file called **dcmsKey.ser** in the home directory of the user running the servlet container.

This file is used to encrypt and decrypt all data fields whose Encryption attributes are set to “1-way” or “2-way”. Bear in mind, that if you change this file for any reason, then you will *not* be able to run the Sterling Multi-Channel Selling Solution if any data is encrypted. Similarly, if you decide to change the user used to run the servlet container, then make sure that you copy the **dcmsKey.ser** file over to the user’s home directory *before* starting the servlet container or accessing the Sterling Multi-Channel Selling Solution.

- In a Windows 2000 installation of the Sterling Multi-Channel Selling Solution, the user’s home directory is typically **C:\Documents and Settings\<*username*>**.
- In a UNIX installation of the Sterling Multi-Channel Selling Solution, the user’s home directory is typically **/home/<*username*>/**.

Clustering

In a clustered environment, you must make sure that every machine in the cluster uses the same copy of **dcmsKey.ser**. You should start one machine in the cluster first, make sure that the **dcmsKey.ser** file is created, and then copy it over to the home directories on the other cluster machines before starting each of these up and adding them to the cluster.

Storing Passwords

To store user passwords in the Knowledgebase as encrypted strings, follow these instructions. You must perform Step 1 *before* loading the data. Note that you cannot mix unencrypted and encrypted passwords: you must decide in advance whether you will encrypt passwords, and this choice applies to all users.

1. Set the Encryption attribute of the UserAuthenticator DataElement to “1-way” or “2-way”. The DataElements are located in the file specified by the DsDataElements element of the **DataServices.xml** file.
2. Load the minimal data or reference data.
3. Use one of the created users to log in to the Sterling Multi-Channel Selling Solution to create new users. All users created from now on will have encrypted passwords.

Password Policies

The Sterling Multi-Channel Selling Solution supports the ability to specify password policies. These are used to determine how passwords are created, criteria that passwords must satisfy (such as minimum lengths), and how often passwords must be changed. See the *Sterling Multi-Channel Selling Solution Developer Guide* for more information.

Cross-Site Request Forgery Filter

The Sterling Multi-Channel Selling Solution supports the ability to prevent Cross-Site Request Forgery (CSRF) by configuring a servlet filter that invalidates the current session when it detects a potential CSRF request. You can configure the filter using the following init-parameters in the **web.xml** file located in the **WEB-INF** directory.

- noop: set the value of this parameter to true to disable the filter. The default value of this parameter is false.
- checkOnly: set the value of this parameter to true to log a potential CSRF request but not invalidate the current session. The default value of this parameter is false.

- noCheckList: set the value of this parameter to a comma-separated list of messageTypes you want to skip while checking.

Note:	Ensure that the JSPs associated with the "white-list" commands do not contain any known holes, particularly those related to cross-site scripting.

CHAPTER 13

Integrating with External Systems

This chapter provides an overview on integrating your Sterling Multi-Channel Selling Solution with other systems.

Implementing Punchout to an External System

In Release 7.1 of the Sterling Multi-Channel Selling Solution, you can integrate other external systems with the enterprise server to add functionality to your e-commerce system. In general, we support:

- the ability of users to “punch out” from the Sterling Multi-Channel Selling Solution to an external system.
- the ability of users to “punch in” from an external system into the Sterling Multi-Channel Selling Solution.

Punching Out

The following illustration is one example of a punchout process from the Sterling Multi-Channel Selling Solution to a configuration system.

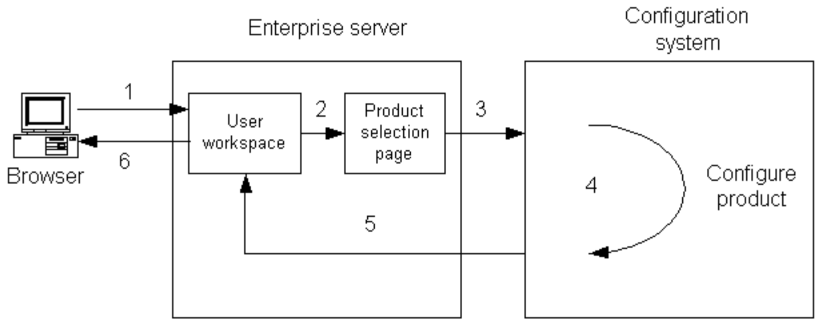


FIGURE 11. **Punchout Process**

Typically, users proceed as follows:

1. Log in to the Sterling Multi-Channel Selling Solution.
2. From the user home page or from a product inquiry list page, click a link to get to a product selection page.
3. Click a product link to punch out to the configuration system.
4. Complete the configuration process.
5. At the end of the configuration process, click a link to return (“punch in”) to the Sterling Multi-Channel Selling Solution.
6. The newly selected and configured product is added to a product inquiry list.
   - If the punchout process is initiated from a product inquiry list, then the product inquiry list detail page is redisplayed with the selected product in it.
   - If the punchout process is initiated from the user home page, then the user is prompted either to select one of their existing product inquiry lists or to create a new product inquiry list.

To Implement the Punchout Mechanism

1. Create a product catalog JSP page.

   Design this template so that when it is used to display a page of configurable products, each product link contains a ReturnURL parameter as well as any information that the product configurator requires. The ReturnURL is used by the product configurator to return session information to the Sterling Multi-Channel Selling Solution once configuration is complete.

   For example, suppose that your product catalog page contains the following text:

   ```
   href="http://myconfigurator.mycompany.com/configurator?
   Model=MX250&ReturnURL=DYNAMIC_URL/ConfiguratorPunchInDisplay&
   SESSION_ID&CART_KEY"
   ```

   Then, at runtime the generated HTML must include the substitution of the DYNAMIC_URL to point to the standard Sterling Multi-Channel Selling Solution port. Similarly, the SESSION_ID and CART_KEY must be substituted for the session ID and cart key (if it exists). Thus, the dynamically-generated HTML might include:

   ```
   href="http://myconfigurator.mycompany.com/configurator?
   Model=MX250&
   ReturnURL=http://mydebs.mycompany.com:9991/-
   ConfiguratorPunchInDisplay&
   SessionID=hgkjg435jk&
   CartKey=247"
   ```

2. Make sure that the configurator is able to preserve the ReturnURL and other parameters as it performs the configuration process.

   At the end of the configuration process, the “Add to cart” button must be capable of:

   - *Either*, generating a Redirect directive that uses the ReturnURL to post the configured product back to the Sterling Multi-Channel Selling Solution together with the session ID and cart key (if it exists).
   - *Or*, constructing the configured product using Javascript and then posting the configured product together with the session ID and cart key (if it exists) directly to the Sterling Multi-Channel Selling Solution.

   The configured product data is returned in the form of an XML document whose data and structure must convey the information required by the Sterling Multi-Channel Selling Solution.

3. Specify a DTD in order to validate that the XML document corresponds to the correct structure.

   A sample XML document is presented here:

```
<?xml version='1.0'?>
<!DOCTYPE Config2BOM SYSTEM 'src/data/messages/Config2BOM.DTD'>
<Config2BOM>
    <LineItem>
        <SKU>MXDS-7500</SKU>
        <Description></Description>
        <Quantity>1</Quantity>
    </LineItem>
    <LineItem>
        <SKU>MX-IN500</SKU>
        <Description>500 MHz Processor</Description>
        <Quantity>1</Quantity>
    </LineItem>
    <LineItem>
        <SKU>MX-LNXA</SKU>
        <Description>Linux Office Suite</Description>
        <Quantity>1</Quantity>
    </LineItem>
</Config2BOM>
```

   The following fragment of Javascript and HTML adds this product to a form before the form is posted to the Sterling Multi-Channel Selling Solution:

```
<HTML>
<script language="JavaScript">
function addConfiguredProduct()
{
    document.Config2Form.XML.value="<?xml version='1.0'?>
    <!DOCTYPE Config2BOM SYSTEM 'src/data/msgs/Config2BOM.DTD'>
    <Config2BOM>\
    <LineItem>\
    <SKU>MXDS-7500</SKU>\
    <Description>\
    (null)\
    </Description>\
    <Quantity>1</Quantity>\
    </LineItem>\
    <LineItem>\
    <SKU>MX-IN500</SKU>\
    <Description>\
    500 MHz Processor\
    </Description>\
    <Quantity>1</Quantity>\
    </LineItem>\
    <LineItem>\
    <SKU>MX-LNXA</SKU>\
    <Description>\
    Linux Office Suite\
    </Description>\
    <Quantity>1</Quantity>\
    </LineItem>\
    </Config2BOM>";
}
</script>

<BODY onLoad="addConfiguredProduct()">
<FORM NAME="Config2Form"
    ACTION="http://mydebs.mycompany.com:9991/-
        ConfiguratorPunchInDisplay"
    METHOD="POST">
<INPUT TYPE="HIDDEN" NAME="SessionID" Value="hgkjg435jk">
<INPUT TYPE="HIDDEN" NAME="CartKey" Value="247">
<INPUT TYPE="HIDDEN" NAME="XML" Value="">
<INPUT TYPE="SUBMIT" VALUE="Add Product">
</FORM>
</BODY>
</HTML>
```

4. Modify the Sterling Multi-Channel Selling Solution configuration and write appropriate controllers, BLCs, and JSP pages to process the punch in request.

   Typically, this includes writing a combination of controllers and BLCs that insert the configured product into a product inquiry list. If a product inquiry list key is not provided, then the controller and BLC must allow for a page that offers the user the choice of adding the product to an open product inquiry list or adding it to a new product inquiry list created by the BLC.

Punching In

You can write an entry point into the Sterling Multi-Channel Selling Solution from an external Web application or from one Sterling Multi-Channel Selling Solution application to another simply by specifying a URL that points to the system.

Authentication

When a user accesses a URL within the Sterling Multi-Channel Selling Solution and if no valid session information is available, then the browser is re-directed to the appropriate login page. The original URL is not stored, so after logging in the user must navigate to the intended page.

Specifying the Sterling Advisor Starting State

The Sterling Advisor application supports the ability to specify additional state information in the URL accessing Sterling Advisor for the first time. By creating a URL with additional state information, you can enable users to punch in to a Sterling Advisor session at a predetermined questionnaire page and with predetermined answers. See the *Sterling Multi-Channel Selling Solution Administration Guide* for further information about Sterling Advisor concepts.

The Sterling Advisor punch in URL can take an optional parameter called startstate. The URL may specify more than one value for the startstate parameter. You can specify a value of a startstate parameter as follows:

- startstate=page_<*name of questionnaire page*>
- startstate=answer_<*question name*>_<*answer name*>

For example, suppose the following is a URL used to punch in to Sterling Advisor:

```
http://<server:port>/Sterling/debs/matrix?cmd=advisor&
    startstate=page_International&
    startstate=answer_Enterprise_Networking&
    startstate=answer_UserType_Expert
```

This URL provides the Sterling Advisor application with a specified start page: International; and provides two answers that will be used in firing rules: Networking and Expert associated with the questions Enterprise and UserType respectively.

This URL could be created by means of the scriptlet:

```
<A HREF="<%= link ("catalog", "advisor",
    "startstate=page_International&
    startstate=answer_Enterprise_Networking&
    startstate=answer_UserType_Expert") %>">
```

Note that the start page is identified by its *name*: when you create Sterling Advisor questionnaire pages, each questionnaire page is given a name. If there is no questionnaire page with the specified name, then the parameter is ignored. If there is no question whose name corresponds to the question name, then the startstate parameter is ignored. Similarly, if there is no answer whose name matches the answer name specified in a startstate parameter, then it is ignored.

If there are spaces or special characters in the names of pages, questions, or answers, then you need to encode the parameters before including them in the *link()* method. For example:

```
<% String temp_ParameterString = "startstate=" +
    java.net.URLEncoder.encode("Start Page"); %>
```

```
<A HREF="<%= link ("catalog", "advisor",
    temp_ParameterString) %>">
```

Note the following for the behavior of Sterling Advisor when startstate parameters are provided.

TABLE 15. Sterling Advisor Startstate Behavior

startstate	startstate answers specified	startstate answers not specified
startstate page specified	The answers are set in the state. Sterling Advisor fires its rules until a rule is fired that specifies a questionnaire page. At this point, the Sterling Advisor displays the questionnaire page specified by the start state parameter.	The Sterling Advisor displays the questionnaire page specified by the start state parameter.
startstate page not specified	The answers are set in the state. Sterling Advisor fires its rules until a rule is fired that specifies a questionnaire page. At this point, the Sterling Advisor displays the questionnaire page specified by the rule.	Sterling Advisor displays its standard start page.

CHAPTER 14

Sterling Configurator Integration

This chapter describes how to integrate Sterling Configurator with other e-commerce applications. These applications may be other applications in the Sterling Multi-Channel Selling Solution or third-party e-commerce applications already installed on your e-commerce site. It covers:

Overview

Punching in to the Sterling Configurator

You can use the Sterling Configurator as a stand-alone configurator application. You can integrate it into your existing e-commerce system so that users can select products, configure them, and add the configured products to their cart as a seamless experience. All you have to do is to ensure that the correct information is passed to the Sterling Configurator as the user begins to configure a product, and that the right information is passed back into your e-commerce application.

Follow these steps:

1. Determine the page(s) in your e-commerce application from which you want users to initiate their configuration session. Typically, these will be the pages in your product catalog or shopping cart application on which customers select products or view their selection of products.
2. On each such Web page, add a button or link with the information set in the URL as described in "Punch into Sterling Configurator URL Definition" on page 224. In particular, this URL must include the URL that the Sterling Configurator will use to return the user to your e-commerce application once they have completed their configuration session.
3. Ensure that your e-commerce application can process the returned post from the Sterling Configurator. The configured model is returned in the form of an XML message that conforms to the DTD described in "DTD for Punchin and Punchout Messages" on page 237.
4. Create the models used by the Sterling Configurator to render the models and determine what selections are available during the configuration session. These models can be created using the Sterling Multi-Channel Selling Solution Visual Modeler or they may be created by an other application. If you use an external application to create models, then you must import them as XML files into the Sterling Multi-Channel Selling Solution using the Visual Modeler. You then use the Visual Modeler to translate the models ready for the Sterling Configurator to use. The imported XML files must conform to the DTD described in "DTD for Model Import" on page 227. An example model file for import is given in "Model Export and Import XML Example" on page 243.
5. Determine whatever logic is required to ensure that the correct model is used for each configuration session. The Sterling Multi-Channel Selling Solution uses a mapping from each configurable product to its corresponding model, but you can use other mechanisms too.

Punch into Sterling Configurator URL Definition

In this section, we provide the form of the URL used to punch in to the Sterling Configurator and some example URLs for punching into Sterling Configurator from Sterling Advisor and from Sterling Order Manager. If you are not using the Sterling Multi-Channel Selling Solution, then your e-commerce application must generate the appropriate URLs to punch in from the Web pages you use.

Standard Punchin URL

To perform the punch-in to the Sterling Configurator, you invoke the following message type with a URL of this form:

```
http://<Machine name:port>/Sterling/en/US/direct/matrix?
    cmd=configure
```

The following information must be part of the inbound post into the Sterling Configurator:

1. The name of the model that the Sterling Configurator should invoke. This parameter is mandatory.
2. The return URL is a mandatory parameter. Depending on your e-commerce application, you may need to include a session ID so that the user can continue their work in the e-commerce application.
3. Parameters passed into Sterling Configurator which may used for rules that set default selections or which generate messages. In a standard implementation of Sterling Advisor and Sterling Configurator, answers selected in the Sterling Advisor questionnaire are passed into the Sterling Configurator. In general, these parameters are optional.

To invoke this message type, you must specify the following parameters in the HTTP request:

- Model: the model name to uniquely identify the model used by the Sterling Configurator.
- ReturnURL: used to determine how to return the configured product to the e-commerce application. Depending on your e-commerce application, you may need to include a session ID in order that the user can continue their work in the e-commerce application.
- picks: an array of strings such as picks made during a Sterling Advisor session.
- ConfigXML: information about a current configuration in XML. This must conform to the DTD described in "DTD for Punchin and Punchout Messages" on page 237.

Punching out from a Product Detail Page

For example, this is the form of a URL used to initiate a configuration from a product detail page:

```
http://<Machine name:port>/Sterling/en/US/direct/matrix?
```

```
    cmd=parametersToConfig&productID=MXDS-7500
```

The following parameters are set in the HTTP request:

- productID: when the request is received by the Sterling Multi-Channel Selling Solution, it maps the specified product to the model used to configure the product.
- advisor.picks: these optional parameters set information about picks made during a Sterling Advisor.

When the Sterling Multi-Channel Selling Solution receives this request, it redirects the request to the URL described in "Standard Punchin URL" on page 225 populating the parameters as necessary.

Punching out from an Inquiry List Page

From an inquiry list page, when the users click on **Configure** for a particular product, the request is posted to the following URL:

```
http://localhost:8080/Sterling/en/US/direct/matrix?
    cmd=ConfiguratorPunchOutOrders
```

The following parameters are set in the HTTP request:

- shoppingCartKey: this key is used to identify the inquiry list to which the the product has been added.
- lineKey: this is the line key within the inquiry list.
- returnURL: used to determine how to return the configured product to the inquiry list. Depending on your e-commerce application, you may need to include a session ID in order that the user can continue their work in the e-commerce application.

When the Sterling Multi-Channel Selling Solution receives this request, it redirects the request to the URL described in "Standard Punchin URL" on page 225 populating the parameters as necessary.

DTDs for the Sterling Configurator

This section provides the document type definitions (DTDs) that specify the form of XML documents used by the Configurator. It covers:

- "DTD for Model Import" on page 227
- "DTD for Punchin and Punchout Messages" on page 237

- "DTD for Runtime XML Model File" on page 237

DTD for Model Import

This DTD describes the XML documents used to import models into the Configurator Visual Modeler. Models exported from the Visual Modeler are exported in this form too.

```
<!--
    -- Types:
    --      integer an integer value
    --      number  a floating point number
    --      string  a string value
    --      datestr a date encoded as a YYYY-MM-DD HH:mm:ss.hh string
    --
-->

<!--
    --
    -- Root node of a modeler import/export file
    --
    -->
<!ELEMENT MODELER (#PCDATA | MODELGROUP)*>
<!ATTLIST MODELER version CDATA #IMPLIED>

<!--
    --
    -- LOCALE nodes store text specific to a models usage within one
    -- locale. A description,
    -- property value (for some property types, as determined by the
    -- modeler), error (warn|suggest too) message or tab name can all
    -- be localized
    --
-->
<!ELEMENT LOCALE (#PCDATA | DESCRIPTION | VALUE | MESSAGE | ERRORMES-
SAGE | NAME)*>
<!ATTLIST LOCALE xml:lang CDATA #IMPLIED>

<!--
    --
    -- LOCALES nodes store zero or more LOCALE nodes
    --
-->
<!ELEMENT LOCALES (#PCDATA | LOCALE)*>


<!--
    --
    -- One of the values within a list
    --
-->
<!ELEMENT LISTVAL (#PCDATA)>

<!--
    --
    -- The tabs within a model (used during configurator presentation
    -- to break the user interface into tabbed sections) are stored as
    -- a MODELTAB node, the sequence number determines the
    -- relationship of a tab with its siblings. Each tab contains
    -- ITEMS which are the names of top level option classes within
    -- the model
    --
    -- SEQ:integer the sequence number of the tab
    --
-->
<!ELEMENT MODELTAB (#PCDATA | LOCALES | ITEM)*>
<!ATTLIST MODELTAB SEQ CDATA #IMPLIED>

<!--
    --
    -- LIST's are stored as with this element type
    --
    -- DESCRIPTION:string the description of the list
    -- NAME:stromgthe name of the list
    --
    -->
<!ELEMENT LIST (#PCDATA | LISTVAL)*>
<!ATTLIST LIST DESCRIPTION CDATA #IMPLIED>
<!ATTLIST LIST NAME CDATA #IMPLIED>

<!--
    --
    -- Expansion actions are stored with this
    -- node type
    --
    -- SEQ:integer the sequence number of this expansion action
    -- item:string the path name of the expansion item, . and *.
    --             have special meaning at the start of this value
    -- max:number  the max value this expansion action applies to
    -- min:number  the min value this expansion action applies to
    -- qty:integer the amount of "item" that should be picked, this
    -- value can be a number of a formula
    --
-->
<!ELEMENT ACTIONITEM (#PCDATA)>
<!ATTLIST ACTIONITEM SEQ CDATA #IMPLIED>
<!ATTLIST ACTIONITEM item CDATA #IMPLIED>
<!ATTLIST ACTIONITEM max CDATA #IMPLIED>
```

```
<!ATTLIST ACTIONITEM min CDATA #IMPLIED>
<!ATTLIST ACTIONITEM qty CDATA #IMPLIED>

<!--
-- Rules are attached to items within a model by name, but defined
-- elsewhere in a model
--
    -- BEGINDATE:datestrthe starting date on which this rule will
    -- begin to fire
    -- CHECKPOINT:integer0=no 1=yes, a checkpoint rule will stop all
    -- rule firing if it fires
    -- ENDDATE:datestrthe ending date after which this rule will no
    -- longer fire
    -- NAME:stringthe name of the rule attached at this point
    -- SEQ:integerthe sequence number of this rule with regard to the
    -- other itemrule's attached at this same spot
-->
<!ELEMENT ITEMRULE (#PCDATA)>
<!ATTLIST ITEMRULE BEGINDATE CDATA #IMPLIED>
<!ATTLIST ITEMRULE CHECKPOINT CDATA #IMPLIED>
<!ATTLIST ITEMRULE ENDDATE CDATA #IMPLIED>
<!ATTLIST ITEMRULE NAME CDATA #IMPLIED>
<!ATTLIST ITEMRULE SEQ CDATA #IMPLIED>

<!--
    -- ACTIONS are subnodes of a rule. There are several types of
    -- actions and a single rule can have any combinations of actions.
    --
    -- MSGTYPE:integer 0=error, 1=warn, 2=suggest, empty for
    -- non-message actions
    -- FORMULA:stringused only for expansion actions, this formula is
    -- evaluated to determine which ACTIONITEM nodes should be picked
    -- for the user
    -- PROPNAME:stringused only for assignment rules specifies the
    -- name of a property to set, can be a path name relative to the
    -- location where the rule is attached or relative to the model
    -- itself, or simply property name
    -- PROPVALUE:stringused only for assignment rules to specify the
    -- formula that should be used to calculate the value for the
    -- property
    -- SEQ:integerthe sequence number of this action
    -- TYPE:integerthe type of action
    --
    -- TYPE='0' message rule, FORMULA PROPNAME and PROPVALUE should
    -- be empty
    -- TYPE='1' expansion rule FORMULA should be a formula PROPVALUE
    -- and PROPNAME are empty
    -- TYPE='2' assignment rule PROPNAME should be a property name,
    -- PROPVALUE a formula to assign and FORMULA should be empty
    -- string
    --
-->
<!ELEMENT ACTION (#PCDATA | LOCALES | ACTIONITEM)*>
<!ATTLIST ACTION MSGTYPE CDATA #IMPLIED>
<!ATTLIST ACTION FORMULA CDATA #IMPLIED>
<!ATTLIST ACTION PROPNAME CDATA #IMPLIED>
<!ATTLIST ACTION PROPVALUE CDATA #IMPLIED>
<!ATTLIST ACTION SEQ CDATA #IMPLIED>
<!ATTLIST ACTION TYPE CDATA #IMPLIED>

<!--
    --
    -- A rule is a condition and actions. The condition is specified
    -- as a boolean operation and the actions are previously
    -- described.
    --
    -- DESCRIPTION:stringthe description of the rule
    -- NAME:stringthe name of the rule
    -- TRIGGER:integercurrently the visual modeler only supports rules
    -- whose trigger is the constant '1'
    --
-->
<!ELEMENT RULE (#PCDATA | BOOLOP | ACTION)*>
<!ATTLIST RULE DESCRIPTION CDATA #IMPLIED>
<!ATTLIST RULE NAME CDATA #IMPLIED>
<!ATTLIST RULE TRIGGER CDATA #IMPLIED>

<!--
    --
    -- The root node of a model
    --
    -- BEGINDATE:datestrthe starting date
    -- DESCRIPTION:stringthe description of the item
    -- ENDDATE:datestrthe ending date
    -- ID:stringinternal id of an item, usually the key from the
    -- system that created the item
    -- NAME:stringname of the item
    -- PATH:stringpath within the model group heirarchy to this model
    -- SKU:stringsku associated with the model
    -- SUBASSEMBLY_TYPE:string MODEL, OPTION_CLASS or OPTION_ITEM
-->
<!ELEMENT MODEL (#PCDATA | PROPERTIES | RULE | LOCALES | PROPVAL |
CLASS | CLASS_SUBASSEMBLY | ITEMRULE | TABLES | MODELTABS | ITEM |
LIST)*>
<!ATTLIST MODEL BEGINDATE CDATA #IMPLIED>
<!ATTLIST MODEL DESCRIPTION CDATA #IMPLIED>
<!ATTLIST MODEL ENDDATE CDATA #IMPLIED>
<!ATTLIST MODEL ID CDATA #IMPLIED>
```

```
<!ATTLIST MODEL NAME CDATA #IMPLIED>
<!ATTLIST MODEL PATH CDATA #IMPLIED>
<!ATTLIST MODEL SKU CDATA #IMPLIED>
<!ATTLIST MODEL SUBASSEMBLY_TYPE CDATA #IMPLIED>

<!--
    -- Used for error, suggest, or warning message value
    -->
<!ELEMENT MESSAGE (#PCDATA)>

<!--
    --
    -- An item in a model
    --
    -- BEGINDATE:datestrthe starting date
    -- ENDDATE:datestrthe ending date
    -- ID:stringinternal id of an item, usually the key from the
    -- system that created the item
    -- SKU:stringsku associated with the model
    -- NAME:stringname of the item
    -- SEQ:integerthe sequence number of this item
-->
<!ELEMENT ITEM (#PCDATA | LOCALES | PROPVAL | ITEMRULE)*>
<!ATTLIST ITEM BEGINDATE CDATA #IMPLIED>
<!ATTLIST ITEM ENDDATE CDATA #IMPLIED>
<!ATTLIST ITEM ID CDATA #IMPLIED>
<!ATTLIST ITEM SKU CDATA #IMPLIED>
<!ATTLIST ITEM NAME CDATA #IMPLIED>
<!ATTLIST ITEM SEQ CDATA #IMPLIED>

<!--
    --
    -- A model group
    --
    -- DESCRIPTION:stringthe description of the item
    -- ID:stringinternal id of an item, usually the key from the
    -- system that created the item
    -- NAME:stringname of the item
    -- PATH:stringpath within the model group heirarchy to this model
    -- ROOTMG:stringtrue or false that the item was the root model
    -- group of the export set
-->
<!ELEMENT MODELGROUP (#PCDATA | PROPERTIES | MODEL | MODELGROUP |
RULE)*>
<!ATTLIST MODELGROUP DESCRIPTION CDATA #IMPLIED>
<!ATTLIST MODELGROUP ID CDATA #IMPLIED>
<!ATTLIST MODELGROUP NAME CDATA #IMPLIED>
<!ATTLIST MODELGROUP PATH CDATA #IMPLIED>
<!ATTLIST MODELGROUP ROOTMG CDATA #IMPLIED>
```

```
<!--
    --
    -- An option class in a model
    --
    -- BEGINDATE:datestrthe starting date
    -- ENDDATE:datestrthe ending date
    -- ID:stringinternal id of an item, usually the key from the
    -- system that created the item
    -- NAME:stringname of the item
    -- RATIO:numberthe ratio of this items quantity to it's parent,
    -- for example in a bicycle model, the tire option class may have
    -- a ratio of 2 to it's parents (the bicycle)
    -- 1
    -- SKU:stringsku associated with the item
-->
<!ELEMENT CLASS (#PCDATA | LOCALES | PROPVAL | ITEM | ITEMRULE | CLASS
| ITEM_SUBASSEMBLY)*>
<!ATTLIST CLASS BEGINDATE CDATA #IMPLIED>
<!ATTLIST CLASS ENDDATE CDATA #IMPLIED>
<!ATTLIST CLASS ID CDATA #IMPLIED>
<!ATTLIST CLASS NAME CDATA #IMPLIED>
<!ATTLIST CLASS RATIO CDATA #IMPLIED>
<!ATTLIST CLASS SKU CDATA #IMPLIED>

<!--
    --
    -- A reference to a subassembly causes that subassembly
    -- to appear (to the end user of configurator) as if that
    -- subassembly were part of a larger model
    --
    -- BEGINDATE:datestr   the starting date
    -- ENDDATE:datestr     the ending date
    -- ID:string   internal id of an item, usually the key from the
    -- system that created the item
    -- NAME:string name of the item
    -- SKU:string  sku associated with the item
    -- SUBASSEMBLY_PATH:string the path to the subassembly from the
    -- root model group down
    -->
<!ELEMENT CLASS_SUBASSEMBLY (#PCDATA | LOCALES)*>
<!ATTLIST CLASS_SUBASSEMBLY BEGINDATE CDATA #IMPLIED>
<!ATTLIST CLASS_SUBASSEMBLY ENDDATE CDATA #IMPLIED>
<!ATTLIST CLASS_SUBASSEMBLY ID CDATA #IMPLIED>
<!ATTLIST CLASS_SUBASSEMBLY NAME CDATA #IMPLIED>
<!ATTLIST CLASS_SUBASSEMBLY SKU CDATA #IMPLIED>
<!ATTLIST CLASS_SUBASSEMBLY SUBASSEMBLY_PATH CDATA #IMPLIED>
```

```
<!--
    --
    -- The message for a constraint table. Not necessarily an "error"
    -- in the strict sense of the word, MESSAGE would be a better name
    -- since the MSGTYPE attribute of the TABLE node determines
    -- whether this is an an error, warning, or suggestion.
    --
-->
<!ELEMENT ERRORMESSAGE (#PCDATA)>

<!--
    --
    -- A reference to an optoin item subbassembly
    --
    -- BEGINDATE:datestr   the starting date
    -- ENDDATE:datestr     the ending date
    -- ID:stringinternal id of an item, usually the key from the
    -- system that created the item
    -- NAME:stringname of the item
    -- SKU:stringsku associated with the item
    -- SUBASSEMBLY_PATH:string the path to the subassembly from the
    -- root model group down
-->
<!ELEMENT ITEM_SUBASSEMBLY (#PCDATA | LOCALES)*>
<!ATTLIST ITEM_SUBASSEMBLY BEGINDATE CDATA #IMPLIED>
<!ATTLIST ITEM_SUBASSEMBLY ENDDATE CDATA #IMPLIED>
<!ATTLIST ITEM_SUBASSEMBLY ID CDATA #IMPLIED>
<!ATTLIST ITEM_SUBASSEMBLY NAME CDATA #IMPLIED>
<!ATTLIST ITEM_SUBASSEMBLY SKU CDATA #IMPLIED>
<!ATTLIST ITEM_SUBASSEMBLY SUBASSEMBLY_PATH CDATA #IMPLIED>

<!--
    --
    -- A boolean operation is part of the condition of a rule
    --
    -- BOOLOP:string   and|or
    -- SEQ:integer     relative position of this boolop amongst its
    -- siblings
    -- TYPE:integer    always 0
    --
-->
<!ELEMENT BOOLOP (#PCDATA | FRAGMENT | BOOLOP)*>
<!ATTLIST BOOLOP BOOLOP CDATA #IMPLIED>
<!ATTLIST BOOLOP SEQ CDATA #IMPLIED>
<!ATTLIST BOOLOP TYPE CDATA #IMPLIED>

<!--
    --
    -- A relational fragment: func1(val1) op func2(val2)
    -- FUNC1:stringname of a rule engine
    -- function (sum, count, min, max, value)
    -- FUNC2:stringname of a rule engine
    -- function (sum, count, min, max, value)
    -- NULLACTION:string "fragment is true", "fragment is false",
    -- "rule is true", or "rule is false"; the outcome if either side
    -- of the rule is undefined
    -- OP:string <, >, <=, >=, =, !=, "in", or "not in"
    -- PROP1:stringname of prop1
    -- PROP2:stringname of prop2
    -- SEQ:integerrelative position of this boolop amongst its
    -- siblings
    -- TYPE:integeralways 1
-->
<!ELEMENT FRAGMENT (#PCDATA)>
<!ATTLIST FRAGMENT FUNC1 CDATA #IMPLIED>
<!ATTLIST FRAGMENT FUNC2 CDATA #IMPLIED>
<!ATTLIST FRAGMENT NULLACTION CDATA #IMPLIED>
<!ATTLIST FRAGMENT OP CDATA #IMPLIED>
<!ATTLIST FRAGMENT PROP1 CDATA #IMPLIED>
<!ATTLIST FRAGMENT PROP2 CDATA #IMPLIED>
<!ATTLIST FRAGMENT SEQ CDATA #IMPLIED>
<!ATTLIST FRAGMENT TYPE CDATA #IMPLIED>

<!--
    --
    -- Definition of a property, all properties must
    -- be defined before they are used
    --
    -- DEFAULTVALUE:string the default value for a property
    -- LOCALIZABLE:stringtrue or false whether the values of the
        property can vary by locale, only appropriate for string
        properties
    -- NAME:stringname of the property
    -- TYPE:stringnumber, string, or list
-->
<!ELEMENT PROPERTIES (#PCDATA)>
<!ATTLIST PROPERTIES DEFAULTVALUE CDATA #IMPLIED>
<!ATTLIST PROPERTIES LOCALIZABLE CDATA #IMPLIED>
<!ATTLIST PROPERTIES NAME CDATA #IMPLIED>
<!ATTLIST PROPERTIES TYPE CDATA #IMPLIED>

<!--
    --
    -- A value for a property
    --
    -- LOCALIZABLE:stringtrue or false
    -- NAME:stringthe name of the property
    -- VALUE:stringthe value for a property unless it is localizable
    -- in which case it would be a child VALUE node under
    -- LOCALES and LOCALE nodes
    --
-->
<!ELEMENT PROPVAL (#PCDATA | LOCALES)*>
<!ATTLIST PROPVAL LOCALIZABLE CDATA #IMPLIED>
<!ATTLIST PROPVAL NAME CDATA #IMPLIED>
<!ATTLIST PROPVAL VALUE CDATA #IMPLIED>


<!--
    -- For properties whose values are localizable, these nodes
    -- contain those values, if the property is not localized, then
    -- the value attribute of the PROPVAL element is used
    -- instead
-->
<!ELEMENT VALUE (#PCDATA)>

<!--
    --
    -- Container for the definitions of all the tabs within a model,
    -- tabs are used to group items in the end user presentation of a
    -- model
    --
-->
<!ELEMENT MODELTABS (#PCDATA | MODELTAB)*>

<!--
    --
    -- The name of a tab
    --
    -->
<!ELEMENT NAME (#PCDATA)>

<!--
    --
    -- Description nodes are used to hold the localized values for the
    -- description attribute from nearly any of the above other
    -- elements that have a description attribute
    --
-->
<!ELEMENT DESCRIPTION (#PCDATA)>


<!--
    --
    -- Holds the table rules (constraint tables) for a model
    --
    -->
```

```
<!ELEMENT TABLES (#PCDATA | TABLE)*>

<!--
    --
    -- A constraint table
    --
    -- BEGINDATE:datestrthe starting date
    -- DESCRIPTION:stringthe description of the item
    -- ENDDATE:datestrthe ending date
    -- MSGTYPE:number0=error, 1=warning, 2=suggest
    -- NAME:stringname of the table
-->
<!ELEMENT TABLE (#PCDATA | LOCALES | ROW)*>
<!ATTLIST TABLE BEGINDATE CDATA #IMPLIED>
<!ATTLIST TABLE DESCRIPTION CDATA #IMPLIED>
<!ATTLIST TABLE ENDDATE CDATA #IMPLIED>
<!ATTLIST TABLE MSGTYPE CDATA #IMPLIED>
<!ATTLIST TABLE NAME CDATA #IMPLIED>

<!--
    --
    -- A row within the constraint table
    --
    -- COMPATIBLE:stringtrue or false whether the row expresses
    -- items that work together 'true' or items
    -- that do not work together 'false'
    -- SEQ:integerrelative position of this item amongst its siblings
    --
-->
<!ELEMENT ROW (#PCDATA | COLUMN)*>
<!ATTLIST ROW COMPATIBLE CDATA #IMPLIED>
<!ATTLIST ROW SEQ CDATA #IMPLIED>

<!--
    --
    -- A column within a row of a constraint table
    --
    -- ITEMPATH:stringthe path to an option class that contains the
    -- items
    -- NAME:stringthe name of the option class
    -- SEQ:integerrelative position of this column amongst its
    -- siblings
-->
<!ELEMENT COLUMN (#PCDATA | CELLDATA)*>
<!ATTLIST COLUMN ITEMPATH CDATA #IMPLIED>
<!ATTLIST COLUMN NAME CDATA #IMPLIED>
<!ATTLIST COLUMN SEQ CDATA #IMPLIED>
<!--
    --
    -- The data for a cell within a constraint table column
    --
    -- ITEMPATH:stringthe full path and item name of the constraint
    -- item
    -- NAME:stringthe name of the item
-->
<!ELEMENT CELLDATA (#PCDATA)>
<!ATTLIST CELLDATA ITEMPATH CDATA #IMPLIED>
<!ATTLIST CELLDATA NAME CDATA #IMPLIED>
>
```

DTD for Punchin and Punchout Messages

This DTD describes the XML documents used to specify models and selections as they are included in punchin posts to Configurator and punchout posts from Configurator.

```
<!ELEMENT ConfiguratorBOM (LineItem)>
<!ATTLIST ConfiguratorBOM ModelName  CDATA  #REQUIRED
                          Version    CDATA  #REQUIRED
>

<!ELEMENT LineItem (LineItem*)>
<!ATTLIST LineItem        DescriptionCDATA  #REQUIRED
                          ItemID     CDATA  #REQUIRED
                          ItemKey    CDATA  #REQUIRED
                          Name       CDATA  #REQUIRED
                          Quantity   CDATA  #REQUIRED
                          SKU        CDATA  #REQUIRED
                          Visible (true | false)
                                            #REQUIRED
>
```

Each model passed in a message has a top-level ConfiguratorBOM element that defines as attributes the model name and version number of the model. It contains one LineItem element which in turn can contain many LineItem elements: one for each node of the configurable model. The Quantity attribute is used to record whether the line item is picked and it takes integer values.

DTD for Runtime XML Model File

This DTD describes the runtime version of XML model files that the Configurator uses in order to render the end-user experience. When models are created by the Configurator Visual Modeler or when they are imported from as XML files, the models are translated into this form and saved to the file system as XML files whose name encodes the model name.

```
<!ELEMENT model (property*, list*, propval*, rule*, class*)>
<!ATTLIST modelid CDATA #REQUIRED
```

```
               name CDATA #REQUIRED
               description CDATA #IMPLIED
               partid CDATA #IMPLIED
>
<!ELEMENT property EMPTY>
<!ATTLIST propertyid          ID                 #REQUIRED
                   name       CDATA      #REQUIRED
                   type    (string | list | number)#REQUIRED
                   defaultvalue CDATA            #IMPLIED
>

<!ELEMENT list (#PCDATA)> <!-- put the list values as parsed character
data -->
<!ATTLIST list name CDATA                          #REQUIRED>

<!ELEMENT propval EMPTY>
<!ATTLIST propval id IDREF                         #REQUIRED
    value CDATA                                        #REQUIRED
>

<!ELEMENT rule (boolop, action)>
<!ATTLIST  rule name CDATA                         #REQUIRED
           firingphase preexpansion | expansion | postexpansion |
               inference)                          #REQUIRED
           trigger (true|false)                    #REQUIRED
>

<!ELEMENT boolop (boolop| fragment)*>
<!ATTLIST boolop boolop (or | and | ornot | andnot)
                                                   #REQUIRED>

<!ELEMENT fragment EMPTY>
<!ATTLIST fragment func1 CDATA #REQUIRED
                   prop1 CDATA #REQUIRED
                   op CDATA #REQUIRED
                   func2 CDATA #REQUIRED
                   prop2 CDATA #REQUIRED
                   nullaction (fragfalse | fragtrue |
                              condtrue | condfalse)  #REQUIRED
>

<!ELEMENT action (expansionaction?, (erroraction | warningaction |
    suggestionaction | assignmentaction | addfilteraction |
    performfilteraction |stopaction)*)>

<!ELEMENT expansionaction (expansion)+>
<!ATTLIST expansionaction formula CDATA             #REQUIRED>

<!ELEMENT expansion EMPTY>
```

```
<!ATTLIST expansion min CDATA #REQUIRED
                   max CDATA #REQUIRED
                   qty CDATA #REQUIRED
                   item CDATA #REQUIRED
>

<!ELEMENT addfilteraction EMPTY>
<!ATTLIST addfilteraction filtername CDATA #REQUIRED>
<!ATTLIST addfilteraction property CDATA #REQUIRED>

<!ELEMENT performfilteraction EMPTY>
<!ATTLIST performfilteraction filtername CDATA #REQUIRED>
<!ATTLIST performfilteraction domain CDATA #REQUIRED>

<!ELEMENT erroraction EMPTY>
<!ATTLIST erroraction message CDATA #REQUIRED>

<!ELEMENT warningaction EMPTY>
<!ATTLIST warningaction message CDATA #REQUIRED>

<!ELEMENT suggestionaction EMPTY>
<!ATTLIST suggestionaction message CDATA #REQUIRED>

<!ELEMENT assignmentaction EMPTY>
<!ATTLIST assignmentaction propname CDATA #REQUIRED
                          formula CDATA #REQUIRED
>
<!ELEMENT stopaction EMPTY>

<!ELEMENT class (propval*,rule*,(item | class | subassembly)*)>
<!ATTLIST class name CDATA #REQUIRED
               id CDATA #REQUIRED
               description CDATA #IMPLIED
               partid DATA #IMPLIED
               begindate CDATA #IMPLIED
               enddate CDATA ratio CDATA #IMPLIED
>

<!ELEMENT item (propval*,rule*)>
<!ATTLIST item name CDATA #REQUIRED
               id CDATA #REQUIRED
               description CDATA #IMPLIED
               partid CDATA #IMPLIED
               begindate CDATA #IMPLIED
               enddate CDATA #IMPLIED
>
<!ELEMENT subassembly EMPTY>
<!ATTLIST subassembly name CDATA#REQUIRED
               description CDATA#IMPLIED
               type (model|class|item) #REQUIRED
               partid CDATA#IMPLIED
               include CDATA#REQUIRED
>
```

Sample XML for Configurator

This example below shows the resulting XML message that can be used to punch back from the Configurator into an e-commerce system after configuration. The punchback message is transferred to the receiving system as XML over HTTP.

```
<ConfiguratorBOM ModelName="Symbol_0020Models/Mobile_0020Computing/
VRC694X" Version="1.0">
    <LineItem Description="VRC694X" ItemID="0" ItemKey="206610"
Name="VRC694X" Quantity="1" SKU="VRC694X" Visible="true">
    <LineItem Description="Radio Frequency" ItemID="1"
ItemKey="206611" Name="RF" Quantity="1" SKU="" Visible="false">
    <LineItem Description="Spectrum 24 (11Mbps)" ItemID="2"
ItemKey="206612" Name="Spec24(11Mbps)" Quantity="0" SKU="" Visi-
ble="false"/>
    <LineItem Description="Spectrum 24 (1Mbps)" ItemID="3"
ItemKey="206613" Name="Spec24(1Mbps)" Quantity="1" SKU="" Visi-
ble="true"/>
    </LineItem>
    <LineItem Description="Antenna" ItemID="4" ItemKey="206614"
Name="Antenna" Quantity="1" SKU="" Visible="false">
    <LineItem Description="One Antenna" ItemID="5" ItemKey="206615"
Name="OneAnt" Quantity="0" SKU="" Visible="false"/>
    <LineItem Description="Two Antenna" ItemID="6" ItemKey="206616"
Name="TwoAnt" Quantity="1" SKU="" Visible="true"/>
    </LineItem>
    <LineItem Description="Heater" ItemID="7" ItemKey="206617"
Name="Heater" Quantity="1" SKU="" Visible="false">
    <LineItem Description="Heater" ItemID="8" ItemKey="206618"
Name="Heater" Quantity="0" SKU="" Visible="false"/>
    <LineItem Description="No Heater" ItemID="9" ItemKey="206619"
Name="None" Quantity="1" SKU="" Visible="true"/>
    </LineItem>
    <LineItem Description="Display" ItemID="10" ItemKey="206620"
Name="Display" Quantity="1" SKU="" Visible="false">
    <LineItem Description="8 Line x 40 Character" ItemID="11"
ItemKey="206621" Name="8Ln40Ch" Quantity="1" SKU="" Visible="true"/>
    </LineItem>
    <LineItem Description="RAM Memory" ItemID="12" ItemKey="206622"
Name="RAM" Quantity="1" SKU="" Visible="false">
    <LineItem Description="640KB" ItemID="13" ItemKey="206623"
Name="640KB" Quantity="1" SKU="" Visible="true"/>
    </LineItem>
    <LineItem Description="Keyboard" ItemID="14" ItemKey="206624"
Name="Keybrd" Quantity="1" SKU="" Visible="false">
    <LineItem Description="54 Key (Alphanumeric)" ItemID="15"
ItemKey="206625" Name="54Key" Quantity="1" SKU="" Visible="true"/>
    </LineItem>
    <LineItem Description="Flash Memory Expansion" ItemID="16"
ItemKey="206626" Name="FlashMem" Quantity="1" SKU="" Visible="false">
    <LineItem Description="1.2 MB Flash" ItemID="17" ItemKey="206627"
Name="1_2MB" Quantity="1" SKU="" Visible="true"/>
    </LineItem>
    <LineItem Description="Country Code" ItemID="18" ItemKey="206628"
Name="CountryCode" Quantity="1" SKU="" Visible="true">
    <LineItem Description="Europe (E1) (Australia, Austria, Croatia,
Denmark, Finland, Greece, Iceland, Ireland, Liechtenstein, Norway,
Switzerland and UK)" ItemID="19" ItemKey="206629" Name="E1" Quan-
tity="0" SKU="" Visible="false"/>
    <LineItem Description="European Union" ItemID="20"
ItemKey="206630" Name="EU" Quantity="0" SKU="" Visible="false"/>
    <LineItem Description="United States and Canada" ItemID="21"
ItemKey="206631" Name="US" Quantity="1" SKU="" Visible="true"/>
    </LineItem>
    <LineItem Description="Standard Configurations" ItemID="22"
ItemKey="206632" Name="Standard Configurations" Quantity="1" SKU=""
Visible="true">
    <LineItem Description="VRC6940-00V651US" ItemID="23"
ItemKey="206633" Name="VRC6940-00V651US" Quantity="0" SKU="VRC6940-
00V651US" Visible="false"/>
    <LineItem Description="VRC6940-0HV651E1" ItemID="24"
ItemKey="206634" Name="VRC6940-0HV651E1" Quantity="0" SKU="VRC6940-
0HV651E1" Visible="false"/>
    <LineItem Description="VRC6940-0HV651US" ItemID="25"
ItemKey="206635" Name="VRC6940-0HV651US" Quantity="0" SKU="VRC6940-
0HV651US" Visible="false"/>
    <LineItem Description="VRC6940-20V651US" ItemID="26"
ItemKey="206636" Name="VRC6940-20V651US" Quantity="1" SKU="VRC6940-
20V651US" Visible="true"/>
    <LineItem Description="VRC6940-2HV651E1" ItemID="27"
ItemKey="206637" Name="VRC6940-2HV651E1" Quantity="0" SKU="VRC6940-
2HV651E1" Visible="false"/>
    <LineItem Description="VRC6940-2HV651US" ItemID="28"
ItemKey="206638" Name="VRC6940-2HV651US" Quantity="0" SKU="VRC6940-
2HV651US" Visible="false"/>
    <LineItem Description="VRC6946-20V651EU" ItemID="29"
ItemKey="206639" Name="VRC6946-20V651EU" Quantity="0" SKU="VRC6946-
20V651EU" Visible="false"/>
    <LineItem Description="VRC6946-20V651US" ItemID="30"
ItemKey="206640" Name="VRC6946-20V651US" Quantity="0" SKU="VRC6946-
20V651US" Visible="false"/>
    <LineItem Description="VRC6946-2HV651US" ItemID="31"
```

```
ItemKey="206641" Name="VRC6946-2HV651US" Quantity="0" SKU="VRC6946-
2HV651US" Visible="false"/>
    </LineItem>
    <LineItem Description="Required Accessories" ItemID="32"
ItemKey="206642" Name="RequiredAccessories" Quantity="1" SKU="" Visi-
ble="false"/>
    <LineItem Description="Would you like to view compatible accesso-
ries?" ItemID="33" ItemKey="206643" Name="Q:Accessories" Quan-
tity="1" SKU="" Visible="false">
    <LineItem Description="No" ItemID="34" ItemKey="206644" Name="No"
Quantity="1" SKU="" Visible="true"/>
    <LineItem Description="Yes" ItemID="35" ItemKey="206645"
Name="Yes" Quantity="0" SKU="" Visible="false"/>
    </LineItem>
    <LineItem Description="Antenna (Spectrum 24)" ItemID="36"
ItemKey="206646" Name="S24Antenna" Quantity="1" SKU="" Visi-
ble="false">
    <LineItem Description="ML-2499-APA1-00 - (S24) 6 Inch Rubber Duck
antenna" ItemID="37" ItemKey="206647" Name="6inch" Quantity="0"
SKU="ML-2499-APA1-00" Visible="false"/>
    <LineItem Description="ML-2499-PTA1-01 - (S24) Patch Antenna"
ItemID="38" ItemKey="206648" Name="Patch" Quantity="0" SKU="ML-2499-
PTA1-01" Visible="false"/>
    <LineItem Description="ML-2499-VRA1-00 - (S24) 3 1/2 Inch Rugged
Rubber Duck Antenna" ItemID="39" ItemKey="206649" Name="3HalfInch"
Quantity="0" SKU="ML-2499-VRA1-00" Visible="false"/>
    </LineItem>
    <LineItem Description="Power Cables" ItemID="40" ItemKey="206650"
Name="PowerCables" Quantity="1" SKU="" Visible="false">
    <LineItem Description="25-39385-01 - AC Power supply output cable
to VRC69XX" ItemID="41" ItemKey="206651" Name="AC Ps" Quantity="0"
SKU="25-39385-01" Visible="false"/>
    <LineItem Description="50-14001-006 - AC Power Supply (for fixed
mount)" ItemID="42" ItemKey="206652" Name="AC Ps fixed" Quantity="0"
SKU="50-14001-006" Visible="false"/>
    <LineItem Description="23844-00-00 - AC Line Cord (for fixed
mount)" ItemID="43" ItemKey="206653" Name="AC Lc fixed" Quantity="0"
SKU="23844-00-00" Visible="false"/>
    </LineItem>
    <LineItem Description="Documentation" ItemID="44"
ItemKey="206654" Name="Doc" Quantity="1" SKU="" Visible="false">
    <LineItem Description="72-37641-01 - Product Reference Guide"
ItemID="45" ItemKey="206655" Name="PRG" Quantity="0" SKU="72-37641-
01" Visible="false"/>
    </LineItem>
    <LineItem Description="Miscellaneous" ItemID="46"
ItemKey="206656" Name="Misc" Quantity="1" SKU="" Visible="false">
    <LineItem Description="C163H07 - Additional Bracket Assembly (One
bracket assembly is already included per device)" ItemID="47"
```

```
ItemKey="206657" Name="BAssem" Quantity="0" SKU="C163H07" Visi-
ble="false"/>
    <LineItem Description="C163016 - Spare Isolation Mounts (Included
in Bracket Assembly)" ItemID="48" ItemKey="206658" Name="Is Mounts"
Quantity="0" SKU="C163016" Visible="false"/>
    <LineItem Description="C163019 - Spare Knob Kit (Included in
Bracket Assembly)" ItemID="49" ItemKey="206659" Name="Knob K" Quan-
tity="0" SKU="C163019" Visible="false"/>
    </LineItem>
    <LineItem Description="Cables" ItemID="50" ItemKey="206660"
Name="Cables" Quantity="1" SKU="" Visible="false">
    <LineItem Description="25-38407-01 - LS/KS -3XXX Scanner cable"
ItemID="51" ItemKey="206661" Name="LS/KS Scnr C" Quantity="0" SKU="25-
38407-01" Visible="false"/>
    <LineItem Description="25-41308-01 - P300std scanner cable"
ItemID="52" ItemKey="206662" Name="P300std Scnr C" Quantity="0"
SKU="25-41308-01" Visible="false"/>
    <LineItem Description="25-38408-01 - Spare DC Power Cable 10
(VRC6940 to Forklift) (One DC Power Cable is already included per
device)" ItemID="53" ItemKey="206663" Name="DC Pwr C" Quantity="0"
SKU="25-38408-01" Visible="false"/>
    <LineItem Description="25-38411-01 - RS232 Cable" ItemID="54"
ItemKey="206664" Name="RS232 C" Quantity="0" SKU="25-38411-01" Visi-
ble="false"/>
    </LineItem>
    <LineItem Description="EPOG Model Group" ItemID="55"
ItemKey="206665" Name="EPOG Model Group" Quantity="1" SKU="" Visi-
ble="true">
    <LineItem Description="Terminals" ItemID="56" ItemKey="206666"
Name="Terminals" Quantity="0" SKU="" Visible="false"/>
    </LineItem>
    </LineItem>
</ConfiguratorBOM>
```

Model Export and Import XML Example

The XML format described below is used to import models into the modeling environment of the Sterling Configurator called Visual Modeler.

```
<MODELER version="2">
    <MODEL BEGINDATE="2001-12-12 14:16:52.0" DESCRIPTION="VRC694X"
ENDDATE="2101-11-18 14:16:52.0" ID="I_200020" NAME="VRC694X"
SUBASSEMBLY_TYPE="MODEL">
    <LOCALES>
        <LOCALE xml:lang="en-US">
            <DESCRIPTION>VRC694X</DESCRIPTION>
        </LOCALE>
    </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="UI: ICON GRAPHIC" VALUE="../
images/Symbol/models/69beauty.jpg"/>
    <PROPVAL LOCALIZABLE="false" NAME="MoreInfo" VALUE="http://
www.symbol.com/products/mobile_computers/
mobile_stationaryvmt_vrc_6900.html"/>
    <CLASS BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206802" NAME="RF" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Radio Frequency</DESCRIPTION>
            </LOCALE>
        </LOCALES>
        <PROPVAL LOCALIZABLE="false" NAME="UI: PRICING STYLE"
VALUE="NONE"/>
        <PROPVAL LOCALIZABLE="false" NAME="UI: IGNORE IN QUOTE"
VALUE="yes"/>
        <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206803" NAME="Spec24(11Mbps)" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Spectrum 24 (11Mbps)</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
    <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206804" NAME="Spec24(1Mbps)" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Spectrum 24 (1Mbps)</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
    </CLASS>
    <CLASS BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206805" NAME="Antenna" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Antenna</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="UI: PRICING STYLE"
VALUE="NONE"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: IGNORE IN QUOTE"
VALUE="yes"/>
    <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206806" NAME="OneAnt" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>One Antenna</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
    <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206807" NAME="TwoAnt" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Two Antenna</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
    </CLASS>
    <CLASS BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206808" NAME="Heater" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Heater</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="UI: IGNORE IN QUOTE"
VALUE="yes"/>
    <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206809" NAME="Heater" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Heater</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
    <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206810" NAME="None" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>No Heater</DESCRIPTION>
            </LOCALE>
        </LOCALES>
        </ITEM>
    </CLASS>
    <CLASS BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206811" NAME="Display" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Display</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="UI: PRICING STYLE"
VALUE="NONE"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: IGNORE IN QUOTE"
VALUE="yes"/>
```

```
    <PROPVAL LOCALIZABLE="false" NAME="UI: REQUIRED" VALUE="true"/>
    <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206812" NAME="8Ln40Ch" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>8 Line x 40 Character</DESCRIPTION>
            </LOCALE>
        </LOCALES>
        </ITEM>
    </CLASS>
    <CLASS BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206813" NAME="RAM" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>RAM Memory</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="UI: PRICING STYLE"
VALUE="NONE"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: IGNORE IN QUOTE"
VALUE="yes"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: REQUIRED" VALUE="true"/>
    <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206814" NAME="640KB" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>640KB</DESCRIPTION>
            </LOCALE>
        </LOCALES>
        </ITEM>
    </CLASS>
    <CLASS BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206815" NAME="Keybrd" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Keyboard</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="UI: PRICING STYLE"
VALUE="NONE"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: IGNORE IN QUOTE"
VALUE="yes"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: REQUIRED" VALUE="true"/>
    <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206816" NAME="54Key" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>54 Key (Alphanumeric)</DESCRIPTION>
            </LOCALE>
        </LOCALES>
        </ITEM>
    </CLASS>
    <CLASS BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206817" NAME="FlashMem" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Flash Memory Expansion</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="UI: PRICING STYLE"
VALUE="NONE"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: IGNORE IN QUOTE"
VALUE="yes"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: REQUIRED" VALUE="true"/>
    <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206818" NAME="1_2MB" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>1.2 MB Flash</DESCRIPTION>
            </LOCALE>
        </LOCALES>
        </ITEM>
    </CLASS>
    <CLASS BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206819" NAME="CountryCode" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Country Code</DESCRIPTION>
            </LOCALE>
        </LOCALES>
        <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206820" NAME="E1" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Europe (E1) (Australia, Austria, Croatia,
Denmark, Finland, Greece, Iceland, Ireland, Liechtenstein, Norway,
Switzerland and UK)</DESCRIPTION>
            </LOCALE>
        </LOCALES>
        </ITEM>
        <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206821" NAME="EU" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>European Union</DESCRIPTION>
            </LOCALE>
        </LOCALES>
        </ITEM>
        <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206822" NAME="US" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>United States and Canada</DESCRIPTION>
            </LOCALE>
        </LOCALES>
        </ITEM>
    </CLASS>
    <CLASS BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206823" NAME="Standard Configurations" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Standard Configurations</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="SYM_ITEM_TYPE" VALUE="STDCON-
FIG"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: OPTION CLASS VIEW"
VALUE="INVISIBLEPOPUP"/>
    <PROPVAL LOCALIZABLE="true" NAME="UI: POST PICK GUIDING TEXT">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <VALUE>Standard Configuration</VALUE>
            </LOCALE>
        </LOCALES>
    </PROPVAL>
    <PROPVAL LOCALIZABLE="false" NAME="MoreInfo" VALUE="http://
www.symbol.com/products/mobile_computers/
mobile_stationaryvmt_vrc_6900.html"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: ICON GRAPHIC" VALUE="../
images/Symbol/models/69beauty.jpg"/>
        <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206824" NAME="VRC6940-00V651US" SKU="VRC6940-
00V651US">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>VRC6940-00V651US</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="StdConfigSelected"
VALUE="TRUE"/>
        </ITEM>
        <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206825" NAME="VRC6940-0HV651E1" SKU="VRC6940-
0HV651E1">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>VRC6940-0HV651E1</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="StdConfigSelected"
VALUE="TRUE"/>
        </ITEM>
        <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206826" NAME="VRC6940-0HV651US" SKU="VRC6940-
0HV651US">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>VRC6940-0HV651US</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="StdConfigSelected"
VALUE="TRUE"/>
        </ITEM>
        <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206827" NAME="VRC6940-20V651US" SKU="VRC6940-
20V651US">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>VRC6940-20V651US</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="StdConfigSelected"
VALUE="TRUE"/>
        </ITEM>
        <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206828" NAME="VRC6940-2HV651E1" SKU="VRC6940-
2HV651E1">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>VRC6940-2HV651E1</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="StdConfigSelected"
VALUE="TRUE"/>
        </ITEM>
        <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206829" NAME="VRC6940-2HV651US" SKU="VRC6940-
2HV651US">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>VRC6940-2HV651US</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="StdConfigSelected"
VALUE="TRUE"/>
    </ITEM>
    <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206830" NAME="VRC6946-20V651EU" SKU="VRC6946-
20V651EU">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>VRC6946-20V651EU</DESCRIPTION>
            </LOCALE>
        </LOCALES>
        <PROPVAL LOCALIZABLE="false" NAME="StdConfigSelected"
VALUE="TRUE"/>
    </ITEM>
    <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206831" NAME="VRC6946-20V651US" SKU="VRC6946-
20V651US">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>VRC6946-20V651US</DESCRIPTION>
            </LOCALE>
        </LOCALES>
        <PROPVAL LOCALIZABLE="false" NAME="StdConfigSelected"
VALUE="TRUE"/>
    </ITEM>
    <ITEM BEGINDATE="2001-12-12 14:16:52.0" ENDDATE="2101-11-18
14:16:52.0" ID="I_206832" NAME="VRC6946-2HV651US" SKU="VRC6946-
2HV651US">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>VRC6946-2HV651US</DESCRIPTION>
            </LOCALE>
        </LOCALES>
        <PROPVAL LOCALIZABLE="false" NAME="StdConfigSelected"
VALUE="TRUE"/>
    </ITEM>
    </CLASS>
    <CLASS BEGINDATE="2001-12-18 12:39:09.0" ENDDATE="2101-12-18
12:39:09.0" ID="I_206833" NAME="RequiredAccessories" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
            <DESCRIPTION>Required Accessories</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="true" NAME="UI: CONSTANT GUIDING TEXT">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <VALUE>The above Standard Configuration does not ship
with the items listed below. Selecting each of these items will ensure
a complete hardware solution.</VALUE>
            </LOCALE>
        </LOCALES>
    </PROPVAL>
    <PROPVAL LOCALIZABLE="false" NAME="UI: CONTROL" VALUE="CHECKBOX"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: OPTION CLASS VIEW"
VALUE="INVISIBLE"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: IGNORE IN QUOTE"
VALUE="yes"/>
</CLASS>
    <CLASS BEGINDATE="2001-12-18 12:40:36.0" ENDDATE="2101-12-18
12:40:36.0" ID="I_206834" NAME="Q:Accessories" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Would you like to view compatible accesso-
ries?</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="UI: OPTION CLASS VIEW"
VALUE="INVISIBLE"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: PRICING STYLE"
VALUE="NONE"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: IGNORE IN QUOTE"
VALUE="yes"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: REQUIRED" VALUE="true"/>
    <ITEM BEGINDATE="2001-12-18 12:41:15.0" ENDDATE="2101-12-18
12:41:15.0" ID="I_206835" NAME="No" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>No</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
    <ITEM BEGINDATE="2001-12-18 12:41:20.0" ENDDATE="2101-12-18
12:41:20.0" ID="I_206836" NAME="Yes" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Yes</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="Q:Accessories" VALUE="Yes"/>
    </ITEM>
</CLASS>
<CLASS BEGINDATE="2001-12-18 12:02:08.0" ENDDATE="2101-12-18
12:02:08.0" ID="I_206837" NAME="S24Antenna" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Antenna (Spectrum 24)</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="UI: CONTROL" VALUE="CHECKBOX"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: OPTION CLASS VIEW"
```

```
VALUE="INVISIBLE"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: IGNORE IN QUOTE"
VALUE="yes"/>
    <ITEMRULE BEGINDATE="2001-12-18 12:42:56.0" CHECKPOINT="0" END-
DATE="2101-12-18 12:42:56.0" NAME="ExpandAccessories" SEQ="1"/>
    <ITEM BEGINDATE="2001-12-18 12:48:33.0" ENDDATE="2101-12-18
12:48:33.0" ID="I_206838" NAME="6inch" SKU="ML-2499-APA1-00">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>ML-2499-APA1-00 - (S24) 6 Inch Rubber Duck
antenna</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
    <ITEM BEGINDATE="2001-12-18 12:48:44.0" ENDDATE="2101-12-18
12:48:44.0" ID="I_206839" NAME="Patch" SKU="ML-2499-PTA1-01">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>ML-2499-PTA1-01 - (S24) Patch Antenna</
DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
    <ITEM BEGINDATE="2001-12-18 12:48:51.0" ENDDATE="2101-12-18
12:48:51.0" ID="I_206840" NAME="3HalfInch" SKU="ML-2499-VRA1-00">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>ML-2499-VRA1-00 - (S24) 3 1/2 Inch Rugged
Rubber Duck Antenna</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
</CLASS>
<CLASS BEGINDATE="2001-12-18 12:02:23.0" ENDDATE="2101-12-18
12:02:23.0" ID="I_206841" NAME="PowerCables" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Power Cables</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="UI: CONTROL" VALUE="CHECKBOX"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: OPTION CLASS VIEW"
VALUE="INVISIBLE"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: IGNORE IN QUOTE"
VALUE="yes"/>
    <ITEMRULE BEGINDATE="2001-12-18 12:46:40.0" CHECKPOINT="0" END-
DATE="2101-12-18 12:46:40.0" NAME="ExpandAccessories" SEQ="1"/>
    <ITEM BEGINDATE="2001-12-18 12:49:09.0" ENDDATE="2101-12-18
12:49:09.0" ID="I_206842" NAME="AC Ps" SKU="25-39385-01">
```

```
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>25-39385-01 - AC Power supply output cable 
to VRC69XX</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
    <ITEM BEGINDATE="2001-12-18 12:49:19.0" ENDDATE="2101-12-18 
12:49:19.0" ID="I_206843" NAME="AC Ps fixed" SKU="50-14001-006">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>50-14001-006 - AC Power Supply (for fixed 
mount)</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
    <ITEM BEGINDATE="2001-12-18 12:49:26.0" ENDDATE="2101-12-18 
12:49:26.0" ID="I_206844" NAME="AC Lc fixed" SKU="23844-00-00">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>23844-00-00 - AC Line Cord (for fixed 
mount)</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
    </CLASS>
    <CLASS BEGINDATE="2001-12-18 12:02:30.0" ENDDATE="2101-12-18 
12:02:30.0" ID="I_206845" NAME="Doc" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Documentation</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="UI: CONTROL" VALUE="CHECKBOX"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: OPTION CLASS VIEW" 
VALUE="INVISIBLE"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: IGNORE IN QUOTE" 
VALUE="yes"/>
    <ITEMRULE BEGINDATE="2001-12-18 12:47:07.0" CHECKPOINT="0" END-
DATE="2101-12-18 12:47:07.0" NAME="ExpandAccessories" SEQ="1"/>
    <ITEM BEGINDATE="2001-12-18 12:49:42.0" ENDDATE="2101-12-18 
12:49:42.0" ID="I_206846" NAME="PRG" SKU="72-37641-01">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>72-37641-01 - Product Reference Guide</
DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
```

```
</CLASS>
<CLASS BEGINDATE="2001-12-18 12:02:38.0" ENDDATE="2101-12-18
12:02:38.0" ID="I_206847" NAME="Misc" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Miscellaneous</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="UI: CONTROL" VALUE="CHECKBOX"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: OPTION CLASS VIEW"
VALUE="INVISIBLE"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: IGNORE IN QUOTE"
VALUE="yes"/>
    <ITEMRULE BEGINDATE="2001-12-18 12:47:30.0" CHECKPOINT="0" END-
DATE="2101-12-18 12:47:30.0" NAME="ExpandAccessories" SEQ="1"/>
    <ITEM BEGINDATE="2001-12-18 12:49:59.0" ENDDATE="2101-12-18
12:49:59.0" ID="I_206848" NAME="BAssem" SKU="C163H07">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>C163H07 - Additional Bracket Assembly
(One bracket assembly is already included per device)</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
    <ITEM BEGINDATE="2001-12-18 12:50:06.0" ENDDATE="2101-12-18
12:50:06.0" ID="I_206849" NAME="Is Mounts" SKU="C163016">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>C163016 - Spare Isolation Mounts
(Included in Bracket Assembly)</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
    <ITEM BEGINDATE="2001-12-18 12:50:12.0" ENDDATE="2101-12-18
12:50:12.0" ID="I_206850" NAME="Knob K" SKU="C163019">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>C163019 - Spare Knob Kit (Included in
Bracket Assembly)</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
</CLASS>
<CLASS BEGINDATE="2001-12-18 12:02:45.0" ENDDATE="2101-12-18
12:02:45.0" ID="I_206851" NAME="Cables" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Cables</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="UI: CONTROL" VALUE="CHECKBOX"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: OPTION CLASS VIEW"
VALUE="INVISIBLE"/>
    <PROPVAL LOCALIZABLE="false" NAME="UI: IGNORE IN QUOTE"
VALUE="yes"/>
    <ITEMRULE BEGINDATE="2001-12-18 12:47:50.0" CHECKPOINT="0" END-
DATE="2101-12-18 12:47:50.0" NAME="ExpandAccessories" SEQ="1"/>
    <ITEM BEGINDATE="2001-12-18 12:50:31.0" ENDDATE="2101-12-18
12:50:31.0" ID="I_206852" NAME="LS/KS Scnr C" SKU="25-38407-01">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>25-38407-01 - LS/KS -3XXX Scanner cable</
DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
    <ITEM BEGINDATE="2001-12-18 12:50:37.0" ENDDATE="2101-12-18
12:50:37.0" ID="I_206853" NAME="P300std Scnr C" SKU="25-41308-01">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>25-41308-01 - P300std scanner cable</
DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
    <ITEM BEGINDATE="2001-12-18 12:50:43.0" ENDDATE="2101-12-18
12:50:43.0" ID="I_206854" NAME="DC Pwr C" SKU="25-38408-01">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>25-38408-01 - Spare DC Power Cable 10
(VRC6940 to Forklift) (One DC Power Cable is already included per
device)</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
    <ITEM BEGINDATE="2001-12-18 12:50:49.0" ENDDATE="2101-12-18
12:50:49.0" ID="I_206855" NAME="RS232 C" SKU="25-38411-01">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>25-38411-01 - RS232 Cable</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    </ITEM>
</CLASS>
<CLASS BEGINDATE="2002-02-12 12:40:05.0" ENDDATE="2102-02-12
12:40:05.0" ID="I_206856" NAME="EPOG Model Group" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
```

```
                <DESCRIPTION>EPOG Model Group</DESCRIPTION>
            </LOCALE>
        </LOCALES>
    <PROPVAL LOCALIZABLE="false" NAME="UI: OPTION CLASS VIEW"
VALUE="INVISIBLE"/>
    <ITEM BEGINDATE="2002-02-12 12:40:46.0" ENDDATE="2102-02-12
12:40:46.0" ID="I_206857" NAME="Terminals" SKU="">
        <LOCALES>
            <LOCALE xml:lang="en-US">
                <DESCRIPTION>Terminals</DESCRIPTION>
            </LOCALE   <LOCALE>
        </LOCALES>
    </ITEM>
    </CLASS>
    </MODEL>
</MODELER>
```

CHAPTER 15

Implementing a Payment Gateway

You can integrate your order management processing with external payment gateways to manage credit card and gift card processing. This chapter describes the setup steps required to manage your payment gateways. It covers the following topics:

- "Default Payment Gateways" on page 257
- "Split Payments and the Payment Charge Sequence" on page 258
- "Gateway Implementation Considerations" on page 259
- "Credit Card Gateway Set-Up" on page 259
- "Gift Card Gateway Set-Up" on page 260
- "Customizing the Payment Gateway" on page 261
- "Credit Card Number Checking" on page 263
- "Troubleshooting" on page 263

Default Payment Gateways

Out of the box, the Sterling Multi-Channel Selling Solution provides the following credit card gateway options:

- None (no credit card processing)

- Cybersource

Out of the box, the Sterling Multi-Channel Selling Solution provides the following gift card gateway options:

- None (no gift card processing)
- Test (a stub for testing gift card processing functions)

Storefront administrators enable or disable credit card and gift card processing from the **Commerce** tab of their Organization Profile page.

Split Payments and the Payment Charge Sequence

When credit card and gift card payment methods are enabled, consumers can split their payments among one or more credit cards, accounts, and gift cards. Consumers can specify the amounts to charge against credit cards and accounts, but the system uses up to the full value of the gift card. Consumers can specify a total of seven payment methods.

The charge sequence is:

1. Gift cards in the order entered, using the available balance of each gift card
2. Credit cards and accounts in the order entered, using the specified maximum charge amount

Note that the maximum charge amount is different from the credit limit on a credit card.

The charge sequence continues until the full value of the order is charged. The last non-gift card specified receives the charge for the balance of the order, even if the balance exceeds the specified maximum charge amount. If the Sterling Multi-Channel Selling Solution handles the credit card authorization and the amount charged on a credit card exceeds the credit card limit, credit card authorization fails.

For example, suppose that a consumer’s order total is $150 and their default payment type is CC1 (Visa). The charge sequence is as follows:

1. The consumer specifies a charge amount of $50 to their default credit card, CC1 (Visa).
2. The consumer enters gift card GC1 which has a $30 credit
3. The consumer enters gift card GC2 which has a $20 credit

4. The consumer enters a second credit card, CC2 (MasterCard), with a charge amount of $30

Since the system charges purchase amounts against gift cards first, the following is the actual charge sequence:

1. GC1: $30: the remaining balance is $120
2. GC2: $20: the remaining balance is $100
3. CC1: $50. the remaining balance is $50
4. CC2: $50, the remaining balance is $0.

Gateway Implementation Considerations

As part of payment gateway implementation, consider the order state machines used by the enterprise and storefront partners (see CHAPTER 20, "Order and Return Management" for more details). In particular, we recommend making the Cancelled, Rejected, and Shipped states in the storefront order state machine end states, or consider carefully what should happen to transactions if there are transitions out of these states.

Credit Card Gateway Set-Up

To Set Up the Cybersource Credit Card Payment Gateway

1. Identify or create a directory on the servlet container machine in which you will store merchant certificates and the logging files. We refer to this directory as ***cc_gateway***. We recommend that you create a sub-directory ***cc_gateway*/ logging/** for the log files.
2. Obtain a merchant certificate from the gateway provider, for example CyberSource. The merchant certificate is a file that is used to authenticate your transactions with the gateway, and it must be associated with your merchant id. A typical name for the file is **cmgt.p12**.
3. Copy the merchant certificate to ***cc_gateway***.
4. Log into the Sterling Multi-Channel Selling Solution as an enterprise administrator, and navigate to the organization profile.
5. Click the **Commerce** tab.
6. Click **CC Payment Gateway**.

7. Select CyberSource (or the name for the supported gateway) from the Credit Card Payment Processor Gateway Type drop-down list.
8. Enable CV Number Check: Set this to True if you require end-users to enter their credit card CV code before placing their credit card order.
9. Processing Type for Orders: Set this to Authorization Only to handle only credit card authorization. Set this to Authorization and Settlement to handle credit card authorization and settlement.
10. Merchant ID: Enter your Merchant ID. Note that the Merchant ID string must be the basename of the merchant certificate file. For example, if the merchant certificate file is **matrix.p12**, then the Merchant ID must be set to “matrix”.
11. Key Directory: The path to ***cmgt_gateway***.
12. Target API Version: currently, you should enter 1.7.
13. You can leave Send to Production and Enable Logging as False or change them to True as appropriate.
14. Log Directory: The path to the logging directory.
15. Log Maximum Size (MB): We suggest setting a value of 10.
16. Click **Save**.

Gift Card Gateway Set-Up

To Set Up a Gift Card Payment Gateway

1. Navigate to your organization profile.
2. Click the **Commerce** tab.
3. Click **GC Payment Gateway.** The Gift Card Payment Gateway Setup page displays.
4. Choose the Gift Card Payment Processor Gateway Type from the drop-down list.

   See the "Customizing the Payment Gateway" on page 261 for information about setting up payment gateways.
5. Enter the required information for the gift card payment gateway, if any, then click **Save**.
6. Click **Back** to return to the **Commerce** tab.

7. Check the Gift Card check box.
8. Click **Save**.

Customizing the Payment Gateway

The most common form of customization is to add a payment gateway. To add a credit card gateway, see "To Add a Credit Card Payment Gateway" on page 261. To add a gift card gateway, see "To Add a Gift Card Payment Gateway" on page 262.

To Add a Credit Card Payment Gateway

To be able to communicate with the new credit card gateway type, follow these steps:

1. Select a new integer code to represent the new credit card gateway type (current codes being used can be found in com.comergent.api.appservices.payment.IPaymentGateway or by looking in the CMGT_LOOKUPS database table for the lookup type “PaymentGatewayType”).
2. Write a handler class that implements the com.comergent.api.appservices.payment.ICreditCardPaymentHandler interface to handle interaction with the new gateway. Be sure that the code writes a transaction history record to the database with the results of each requested transaction.
3. Create an entry in **ObjectMap.xml** configuration file for your new handler using the new integer code by following the instructions in the com.comergent.api.appservices.payment.ICreditCardPaymentHandlerFactory description. Subclassing the object factory should generally not be necessary.

To be able to configure the new credit card gateway type through the UI:

4. Add an entry for the new integer code to the CMGT_LOOKUPS database table for the lookup type “PaymentGatewayType”.
5. Modify the **WEB-INF/web/en/US/payment/PGSetup.jsp** file *updateFieldVisibility()* method by adding a block to the “// Hide or display the individual fields.” section for the new gateway type to control which data fields should be hidden and which should be shown when a gateway of this type is selected. (If additional payment gateway fields are added to the screen, add the necessary hideID calls to the other gateway blocks.)

6. Modify the PaymentSetupController class to validity check and save the fields applicable to the new gateway type (the branch-by-gateway type logic is in the *validateAndSaveGatewayInformation()* method).

To Add a Gift Card Payment Gateway

To be able to communicate with the new gift card gateway type:

1. Select a new integer code to represent the new gift card gateway type (current codes being used can be found in com.comergent.api.appservices.payment.giftCard.IGiftCardGateway or by looking in the CMGT_LOOKUPS database table for lookup_type "GiftCardPaymentGatewayType").
2. Write a handler class that implements the com.comergent.api.appservices.payment.giftCard.IGiftCardPaymentHandler interface to handle interaction with the new gateway.
3. Create an entry in ObjectMap.xml for your new handler using the new integer code by following the instructions in the com.comergent.api.appservices.payment.giftCard.IGiftCardPaymentHandlerFactory description. Subclassing the object factory should generally not be necessary.

To be able to configure the new gateway type through the UI:

1. Add an entry for the new integer code to the CMGT_LOOKUPS database table for lookup_type "GiftCardPaymentGatewayType".
2. If the new gateway requires additional configuration fields, we recommend that you modify the **WEB-INF/web/en/US/payment/giftCard/GiftCardGatewaySetup.jsp** file to have an updateFieldVisibility method, to be called from the gatewayTypeChanged method, which will show or hide the fields based on the gateway type selected. See the **WEB-INF/web/en/US/payment/PGSetup.jsp** file to see an example of how this was done for credit card gateways.
3. If the new gateway requires additional configuration fields, you must also modify the com.comergent.reference.appservices.payment.giftCard.controller.GiftCardPaymentSetupController controller to perform validity checking and save the fields applicable to the new gateway type.

Credit Card Number Checking

Credit card numbers are validated for basic correctness of form when users enter them. This validation uses the industry-standard LUHN algorithm. Please see the following URL for more information:

http://www.ee.unb.ca/tervo/ee4253/luhn.html.

Troubleshooting

This section describes known issues and their solution.

TABLE 16. Troubleshooting

Issue	Solution
Users see an error like this: * For card XXXXXXXXXXXX1111 - A system level processing critical error has occurred ("[Security:090549]The certificate chain received from ics2wstest.ic3.com - 66.185.180.11 contained a V3 CA certificate which did not indicate it really is a CA."). Do not attempt to retry this transaction before contacting Customer Service with the details of this message, or duplicate transactions may result.	This is a known problem when WebLogic is run in development mode. You can work around this by specifying the following system properties to the WebLogic server(s): -Dweblogic.security.SSL.enforceConstraints=off

CHAPTER 16

Implementing Portal Support

This chapter describes how to expose functionality from the Sterling Multi-Channel Selling Solution as portlets. These portlets can be deployed into any portlet container that supports the JSR-168 portlet specification.

Overview

The Sterling Multi-Channel Selling Solution provides support for Web services, and provides some Web services as out-of-the-box functionality: see the *Sterling Multi-Channel Selling Solution Developer Guide* for the details of these services. You can use these Web services to work with portlets that can be deployed to your portlet container.

The Sterling Multi-Channel Selling Solution provides some out-of-the-box portlets which can be deployed immediately. See "Implementation" on page 267 for details.

The Selling System WS Proxy can be used as a Web services invocation API and not necessarily as part of a portlets implementation. In other words, it can be made part of any JVM process and support the dynamic invocation of Web services. The Selling System WS Proxy is fully compliant with WS-I standards and recommends the usage of the document/literal wrapped pattern.

Framework

The basic framework of the portal implementation comprises three layers of entities: portlets, proxies, and Web services.

Portlets

These are the entities declared in the Selling System Portal Web application **portlet.xml** configuration file. They are the named elements that you can refer to from the Portal Container (such as Pluto). Each portlet element in the **portlet.xml** configuration file declares a portlet and specifies the proxy which it uses to invoke an underlying Web service.

Proxies

These are the entities used to invoke the Web services: the **WSProxyConfigurations.xml** configuration file specifies which Web service a proxy invokes, and which InputBuilder Java class and ViewControl JSP page is to be used to submit the request to the service and to render the result.

Web Services

The Web services are not part of the Selling System Portal Web application, but are provided by the Sterling Multi-Channel Selling Solution. The **WSProxyConfigurations.xml** configuration file specifies the location of the Web service for each proxy.

Naming Conventions

For each object such as order, invoice, promotion, and so on, the following naming conventions specify the associated portlets.

TABLE 17. **Portlet Naming Conventions**

Portlet	**Proxy**	**Web Service**
*Object*Get	*Object*Get	*Object*Get
*Object*Search	*Object*Search	*Object*Search

The *Object*Get portlet retrieves single instances of an object, and uses the corresponding *Object*Get Web service to retrieve an object identified by a unique key. The *Object*Search portlet displays a list of objects: it uses the corresponding *Object*Search Web service to retrieve a list of objects.

Implementation

The following steps explain how to deploy the out-of-the-box portlets provided by the Sterling Multi-Channel Selling Solution. More detailed instructions for specific Portal Containers are provided in "Pluto Implementation" on page 270

General Implementation Steps

1. In the Sterling Multi-Channel Selling Solution, navigate to System Services -> XML Messages and ensure that the Message URL for the SOAP Messages system property is set correctly. Its value should be something like:

   ```
   http://debsserver:port/Sterling/msg/matrix/soap
   ```

2. Obtain the Selling System Portal Web application WAR file: it is called something like **SterlingPortal-def-RC1.war**.
3. Deploy the Selling System Portal Web application into your portal container. We will refer to the location of the deployed Web application as ***sterling_portal_home*/**.
4. Edit the ***sterling_portal_home*/WEB-INF/WSProxyConfigurations.xml** configuration file by declaring each proxy in a Proxy element and setting the server and port. The following is a sample Proxy element that provides an order search portlet:

   ```
   <Proxy name="OrderSearch" bindingType="SOAP">
       <ServiceDefinition>
           <URL>
           http://server:port/Sterling/dXML/5.0/
   GetWSDL.jsp?sfName=matrix&amp;fileName=OrderInterface.wsdl
           </URL>
           <Service>OrderSearch_Services</Service>
           <Operation>OrderSearchRequest</Operation>
       </ServiceDefinition>
       <InputBuilder>
           <Class>
               com.comergent.portal.data.input.OrderSearchBuilder
           </Class>
           <Parameter name="param1">value1</Parameter>
       </InputBuilder>
       <ViewControl>
           <Class>
               com.comergent.portal.data.view.PortletViewControl
           </Class>
           <Parameter name="jspFile">OrderSearchView.jsp</Parameter>
       </ViewControl>
   ```

```
</Proxy>
```

Note the following items:

- The name attribute of the Proxy element refers to the proxy in the ***sterling_portal_home*/WEB-INF/portlet.xml** configuration file.
- The ServiceDefinition element must point to the URL used to retrieve the WSDL from the Sterling Multi-Channel Selling Solution deployment. The URL has the following format:

  ```
  http://server:port/Sterling/dXML/5.0/
  GetWSDL.jsp?sfName=matrix&amp;fileName=OrderInterface.wsdl
  ```

  sfName is the name of the storefront, for example, matrix. Specify the ampersand separator using &amp.
- The Service element and Operation element specify the precise Web service operation that you want the proxy to invoke.

Use the InputBuilder element to build the service request that you want to generate. Typically this class is used to marshall the beans used to make the request.

Use the ViewControl element to process the result of invoking the Web service. Typically, this is a JSP page that displays the response object in a user-friendly way.

5. Edit the ***sterling_portal_home*/WEB-INF/portlet.xml** configuration file to declare the portlets that you want to make available to the portal container. The following is a sample portlet element:

```
<portlet>
    <description>Sterling WS Proxy Portlet</description>
    <portlet-name>CmgtWSProxy_OrderSearch</portlet-name>
    <display-name>OrderSearch</display-name>
    <portlet-class>
        com.comergent.portal.portlet.WSProxyPortlet
    </portlet-class>
    <init-param>
        <name>ProxyName</name>
        <value>OrderSearch</value>
    </init-param>
    <expiration-cache>-1</expiration-cache>
    <supports>
        <mime-type>text/html</mime-type>
        <portlet-mode>VIEW</portlet-mode>
        <portlet-mode>EDIT</portlet-mode>
        <portlet-mode>HELP</portlet-mode>
    </supports>
```

```
    <supported-locale>en</supported-locale>
    <supported-locale>de</supported-locale>
    <portlet-info>
        <title>WS Proxy</title>
        <short-title>WS Proxy</short-title>
        <keywords>WS, Proxy</keywords>
    </portlet-info>
    <portlet-preferences>
        <preference>
            <name>dummyName</name>
            <value>dummyValue</value>
            <read-only>false</read-only>
        </preference>
        <preference>
            <name>dummyName2</name>
            <value>dummyValue2</value>
        </preference>
        <preference>
            <name>readonly</name>
            <value>readonly</value>
            <read-only>true</read-only>
        </preference>
        <preferences-validator>
            org.apache.pluto.core.impl.PreferencesValidatorImpl
        </preferences-validator>
    </portlet-preferences>
    <security-role-ref>
        <role-name>plutoTestRole</role-name>
        <role-link>tomcat</role-link>
    </security-role-ref>
</portlet>
```

It uses the mandatory ProxyName init-param element to specify which proxy it uses. The value of this parameter must be the same as the name attribute of a proxy declared in the **WSProxyConfigurations.xml** configuration file.

6. Configure the portlet container so that it can access the portlets provided the Selling System Portal Web application. Configuration for a Pluto portal container implementation is described in detail in "Pluto Implementation" on page 270.

7. Restart the portlet container.

If you now access the portlet container through your browser, then you should see that the Sterling Multi-Channel Selling Solution portlets are now available to portlet container users.

Pluto Implementation

Apache Pluto is an open source portal container implementation of the portlet specification. This section describes the implementation steps required to deploy the Sterling Portal Web application within a Pluto version 1.1.4 installation. The steps in this section enable the OrderGet portlet. You can enable other portlets in a similar way.

This section refers to the Pluto installation directory as ***pluto_home***. The instructions assume that you name the Sterling Portal Web application WAR file **Sterling.war** so that the resulting application can be referred to as Sterling.

1. Install Pluto version 1.1.4 onto your portal server machine.
2. Check that the port numbers that the Pluto implementation uses do not conflict with other port numbers used by other servlet containers or Web servers on the portal server machine. To check the port numbers, edit the ***pluto_home*\conf\server.xml** file, check the <Server port> and <Connector port> elements, and change the port values if necessary.
3. Start up Pluto.
4. Navigate to **http://*server:port*** to verify that the Pluto server is up and running.
5. Navigate to **http://*server:port*/pluto/portal**. Log in with user id **pluto** and password **pluto**. The pluto home page displays, showing several tabs: "About Apache Pluto", "Test Page", "Secondary Page" and "Pluto Admin".
6. Deploy the Selling System Portal Web application to Pluto by copying the **Sterling.war** file to the ***pluto_home*/webapps/** directory. If Pluto does not automatically pick up the application, restart Pluto.
7. Edit the ***pluto_home*/webapps/pluto/WEB-INF/Pluto-portal-driver-config.xml** configuration file. You declare the new application and its portlets by adding a page element to the <render-config> element, similar to the following:

```
<page name="Sterling" uri="/WEB-INF/themes/pluto-default-theme.jsp">
   <portlet context="/Sterling" name="CmgtWSProxy_OrderGet"/>
</page>
```

   This step adds a new tab, "Sterling", in the web page of the Pluto application with a reference to Order Get. You can add multiple portlet contexts in this page (Sterling) or create a new page for another portlet context.

8. In the deployed Selling System Portal Web application, edit the ***pluto_home*/webapps/Sterling/WEB-INF/web.xml** configuration file. You must add an element for each portlet similar to the following:

```
<servlet>
   <servlet-name>CmgtWSProxy_OrderGet</servlet-name>
   <display-name>CmgtWSProxy_OrderGet</display-name>
   <description>Automated generated Portlet Wrapper</description>
   <servlet-class>org.apache.pluto.core.PortletServlet</servlet-
class>
<init-param>
   <param-name>portlet-name</param-name>
   <param-value>CmgtWSProxy_OrderGet</param-value>
</init-param>
<init-param>
    <param-name>portlet-guid</param-name>
    <param-value>Sterling.CmgtWSProxy_OrderGet</param-value>
 </init-param>
 <init-param>
    <param-name>portlet-class</param-name>
    <param-value>com.comergent.portal.portlet.WSProxyPortlet</
param-value>
  </init-param>
  <security-role-ref>
     <role-name>plutoTestRole</role-name>
     <role-link>tomcat</role-link>
   </security-role-ref>
 </servlet>
```

The value of the initialization parameter portlet-name in this definition must match one of the portlets defined in the **portlet.xml** configuration file.

Use the corresponding servlet mapping element to map URLs to the named servlet. For simplicity, keep the servlet-name and url-mapping values as indicated.

```
<servlet-mapping>
        <servlet-name>CmgtWSProxy_OrderGet</servlet-name>
        <url-pattern>/PlutoInvoker/CmgtWSProxy_OrderGet/*</url-
pattern>
    </servlet-mapping>
```

This completes the mapping from a URL to the target portlet.

Next, you configure the Selling System Portal Web application.

9. Edit the ***pluto_home*/webapps/Sterling/WEB-INF/portlet.xml** configuration file. Each portlet must be declared in an element similar to the following:

```
<portlet>
        <description>Order Get Portlet</description>
        <portlet-name>CmgtWSProxy_OrderGet</portlet-name>
        <display-name>OrderGet</display-name>

        <portlet-class>com.comergent.portal.portlet.WSProxyPort-
let</portlet-class>

        <init-param>
    <name>ProxyName</name>
    <value>OrderGet</value>
</init-param>

        <init-param>
    <name>InvokeOnLoad</name>
    <value>true</value>
</init-param>

        <expiration-cache>-1</expiration-cache>

        <supports>
     <mime-type>text/html</mime-type>
            <portlet-mode>VIEW</portlet-mode>
            <portlet-mode>EDIT</portlet-mode>
            <portlet-mode>HELP</portlet-mode>
</supports>
<supported-locale>en</supported-locale>
        <supported-locale>de</supported-locale>

<portlet-info>
    <title>Order Get</title>
    <short-title>Order Get</short-title>
    <keywords>WS, Proxy</keywords>
        </portlet-info>
    <portlet-preferences>
            <preference>
    <name>dummyName</name>
    <value>dummyValue</value>
    <read-only>false</read-only>
    </preference>
            <preference>
    <name>dummyName2</name>
    <value>dummyValue2</value>
    </preference>
            <preference>
                <name>readonly</name>
                <value>readonly</value>
                <read-only>true</read-only>
            </preference>
            <preferences-valida-
tor>org.apache.pluto.core.impl.PreferencesValidatorImpl</prefer-
ences-validator>
        </portlet-preferences>
        <security-role-ref>
            <role-name>plutoTestRole</role-name>
            <role-link>tomcat</role-link>
        </security-role-ref>
</portlet>
```

The portlet-name element value must correspond to the name that you defined in the **Pluto-portal-driver-config.xml** file and in the <servlet> element of the **web.xml** file..

10. Edit the ***pluto_home*/webapps/Sterling/WEB-INF/WSProxyConfigurations.xml** configuration file. Note that in the following example, you enter the literal text "matrix&amp".

```
<Proxy name="OrderGet" bindingType="SOAP">
    <ServiceDefinition>
    <URL>http://[server]:[port]/Sterling/dXML/5.1/
GetWSDL.jsp?sfName=matrix&amp;fileName=OrderInterface.wsdl</URL>
    <Service>OrderGetService</Service>
    <Operation>OrderGetRequest</Operation>
    <PortType>OrderGetPortType</PortType>
    </ServiceDefinition>
    <InputBuilder>
        <Class>com.comergent.portal.data.input.OrderGetBuilder</
Class>
        <Parameter name="RemoteUser.UserLogin">ajones</Parameter>
        <Parameter name="RemoteUser.UserAuthenticator">ajones</
Parameter>
        <Parameter name="MessageHeader.MessageID">ID123</Parame-
ter>
        <Parameter name="UserFullName">ABC</Parameter>
    <Parameter name="Order.OrderNumber">1923611486</Parameter>
    </InputBuilder>
    <ViewControl>
    <Class>com.comergent.portal.data.view.PortletViewControl</
Class>
    <Parameter name="jspFile">OrderGetView.jsp</Parameter>
    </ViewControl>
    </Proxy>
```

The name attribute of each Proxy element must match the ProxyName init-param element of a portlet that you defined in the <portlet> element of the **portlet.xml** file in step 8. You must change the URL used to access the WSDL for the Web Service so that it points to the correct server and port for the Sterling Multi-Channel Selling Solution.

Each Proxy configuration encapsulates the necessary parameters for invoking a remote Web service. It contains the following elements:

- ServiceDefinition element: mandatory configuration fragment that contains the details of the Web service to invoke. It includes the WSDL URL, the service to use, and the operation of that service to invoke.
- InputBuilder element: The marshalling logic implementation handler for wrapping any request parameters. Optionally, you can specify parameters that provide additional data to the InputBuilder. You can also use fully-qualified names as the names of parameters. For example:

```
<Parameter name="RemoteUser.UserLogin">ajones</Parameter>
<Parameter name="RemoteUser.UserAuthenticator">ajones</Parameter>
<Parameter name="MessageHeader.MessageID">ID123</Parameter>
<Parameter name="UserFullName">ABC</Parameter>
<Parameter name="OrderSearchCriterion.UserName">cchen</Parameter>
<Parameter name="OrderSearchCriterion.SortedBy">
ShoppingCartKey</Parameter>
```

  This means that the InputBuilder class sets the value of the RemoteUser.UserLogin parameter to “ajones”.

- ViewControl element: The presentation control implementation handler that generates the necessary presentation fragments.
- The Proxy name attribute serves as the handle on a specific proxy and should be unique. As part of any navigation in presentation, any action URL should map to a valid proxy name.

For example, in the JSP page presenting the portlet, make sure that the value of the action attribute corresponds to the proxy name attribute:

```
<portlet:actionURL
    secure='<%=renderRequest.isSecure()?"True":"False"%>'
    var="orderURL">
    <portlet:param name="orderNumber" value='<%= orderNumber%>'/>
    <portlet:param name="action" value="OrderGet"/>
```

```
</portlet:actionURL>
```

In the above example, action=OrderGet maps to the proxy OrderGet fragment.

11. Restart the Pluto portal server.
12. Point your browser to http://server:port/pluto/portal. After logging in, you should see a link: **Sterling**.
13. Click the **Sterling** link: you should now see the portlets that you have defined.

CHAPTER 17

Implementing Sterling MarketLink

Sterling MarketLink enables users of external systems such as Ariba Buyer or Commerce One to “punch in” to the Sterling Multi-Channel Selling Solution. This chapter covers:

- Sterling MarketLink for Ariba Buyer
- Sterling MarketLink for Commerce One

Sterling MarketLink for Ariba Buyer

Overview

Using Sterling MarketLink, Release 7.1 of the Sterling Multi-Channel Selling Solution supports the ability for users of Release 7.0 and later Ariba Buyer systems to punch out from their Buyer session and punch into your implementation of the Sterling Multi-Channel Selling Solution. While users are punched into the Sterling Multi-Channel Selling Solution they can select products, navigate through the product category hierarchy, search the product catalog, conduct a guided selling session, configure complex products, and obtain prices before returning to their Buyer session.

Companies that have installed Ariba Buyer use it to enable their procurement agents to buy products from external partners through their browser. The Ariba Commerce Services Network (ACSN) is set up to enable buyers and suppliers to communicate seamlessly through a common network. Each buying partner and supplier is set up on the ACSN with a profile that includes an Ariba Network ID (ANID) and a password. Suppliers can set up PunchOut and PurchaseOrder URLs as part of their profile.

When a procurement user starts work in an Ariba Buyer session, they are authenticated when they log in. They can browse product catalogs provided in the form of cXML documents by their suppliers and select items from one or more catalogs to purchase. A procurement user can “punch out” from their Ariba session and, through the ACSN, “punch in” to a Sterling Multi-Channel Selling Solution set up to support this functionality.

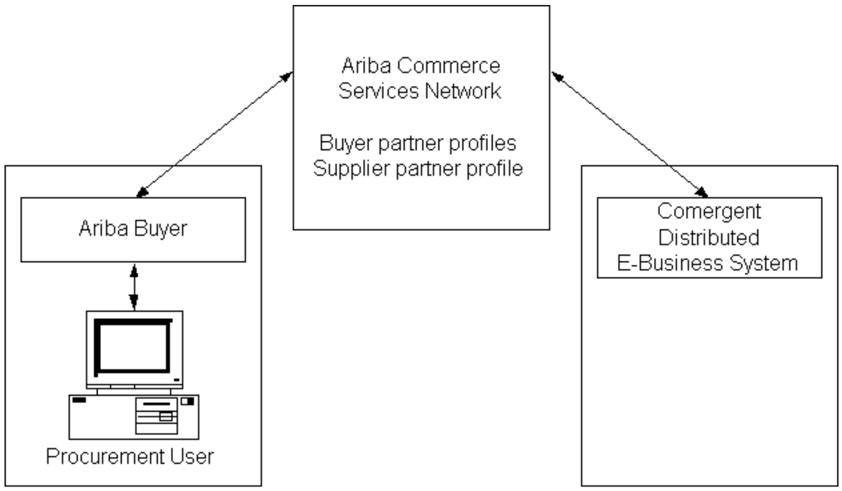


FIGURE 12. Ariba Commerce Services Network

The ACSN uses the partner profiles to provide authentication information and to identify the URLs to use to punch in to the Sterling Multi-Channel Selling Solution.

Setting Up

To set up this functionality, follow these steps:

1. Set up a profile for your organization at the ACSN. The ACSN provides the following:
    a. The ANID for your organization.
    b. The password for your ANID.

2. Set up the configuration parameters in **web.xml**:
   a. In **web.xml**, set up the punchin URL: this is the standard XML message URL. By default, it is “msg/matrix”, but you will almost certainly have changed this as part of your modification of the **web.xml** file described in CHAPTER 9, "Customizing Your Sterling Multi-Channel Selling Solution".
3. Set the values of the General elements of the **Comergent.xml** configuration file as follows:
   a. The value of the UserAgent element must be “DEBS”.
4. Set the values of the MarketLink elements of the **BusinessRule.xml** configuration file as follows:
   a. The value of the AribaIdentity element must be your organization’s ANID provided by Ariba.
5. In your ACSN profile, set the PunchOut and PurchaseOrder URLs. These must be the same, and depend on the values you set up in Step 2a.
6. In your Sterling Multi-Channel Selling Solution, set up a partner profile and partner user for each of your partners who will be punching in from their Ariba Buyer system.
   a. Each partner must have their own partner profile in the Sterling Multi-Channel Selling Solution. Make sure that you create the partner profile as a Direct Commerce-enabled partner. If the partner has an existing Direct Commerce-enabled profile already, then you do not have to create a new one.
   b. Create a partner user for this partner: their username must be the ANID for the *partner* and their password must be *your* ANID password.
   c. Assign the ProcurementUser role to this user.
   d. Create a price list that provides the list of products and prices that you want to make available to this partner through the Ariba Buyer. You can use an existing price list if one meets your needs. However, make sure that the currency of the price list is compatible with the Ariba Buyer system at the partner site. See the *Sterling Multi-Channel Selling Solution Administration Guide* for more information about setting up price lists.
7. For each partner, export your product catalog to them:

a. Create a Catalog Export Set for the partner: take care that you use the price list assigned to the partner in Step 6d, and that you specify the export format to be cXML 1.1. See the *Sterling Multi-Channel Selling Solution Administration Guide* for further information. Note that the AribaIdentity element set in Step Step 4 is used to identify your organization in the generated cXML document.

b. You must produce the product catalog extract as a cXML document, and send it to the partner. You can produce the extract as a one-off task or schedule it so that the catalog information is extracted at regular intervals. See the *Sterling Multi-Channel Selling Solution Administration Guide* for more information.

c. The partner must incorporate the product catalog extract into their installation of Ariba Buyer.

8. Test the system. In particular, make sure that when a user is logged in to the Sterling Multi-Channel Selling Solution as a procurement user they see the buttons that enable them to return to their Ariba Buyer session rather than the buttons that enable a user to place a product inquiry list as an order.

Sterling MarketLink for Commerce One

Overview

Using Sterling MarketLink, Release 7.1 of the Sterling Multi-Channel Selling Solution supports the ability of users of Commerce One Enterprise Buyer systems to punch out from their Enterprise Buyer session and punch into your implementation of the Sterling Multi-Channel Selling Solution. While users are punched into the Sterling Multi-Channel Selling Solution they can select products, navigate through the product category hierarchy, search the product catalog, conduct a guided selling session, configure complex products, and obtain prices before returning to their Enterprise Buyer session.

Companies that have installed Enterprise Buyer use it to enable their procurement agents to buy products from external partners through their browser. The Commerce One MarketSite is set up to enable buyers and suppliers to communicate through a common network seamlessly. Each buying partner and supplier is set up on the MarketSite with a profile that includes a supplier ID. Suppliers can set up PunchIn URLs as part of their profile.

When a procurement user starts work in an Enterprise Buyer session, they are authenticated when they log in. They can browse a list of suppliers and by clicking on the link associated with a supplier, be directed to the supplier's Web site. In particular, if a procurement user wants to, they can "punch out" from their Enterprise Buyer session and, through MarketSite, "punch in" to a Sterling Multi-Channel Selling Solution which has been set up to support this functionality.

Communication between the Enterprise Buyer and the Sterling Multi-Channel Selling Solution uses xCBL 3.0. Make sure that your installations of Enterprise Buyer and MarketSite use this version of xCBL.

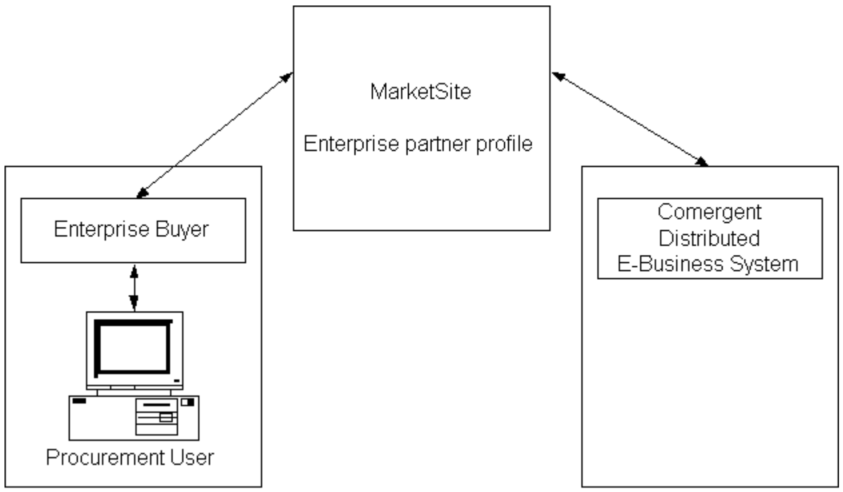

MarketSite
Enterprise partner profile
Enterprise Buyer
Procurement User
Comergent Distributed E-Business System

FIGURE 13. Enterprise Buyer to Sterling Multi-Channel Selling Solution Integration

Setting Up

To set up this functionality, follow these steps:

1. In your Sterling Multi-Channel Selling Solution, create a partner profile for the partner who will be using Enterprise Buyer to punch in.
2. Create a partner user for this partner. You must assign them the MarketProcurementUser role.
3. Create appropriate price lists for the partner and assign them to the partner.
4. Establish a Supplier profile for the enterprise on the MarketSite installation used by the partner. The Supplier profile must include:

- the URL to be used to punch in to the Sterling Multi-Channel Selling Solution. It must be of the form:

  ```
  http://<debsserver>:<port>/Sterling/debs/matrix?
      Username=username&Password=password
  ```

  where the username and password values are for the user created in Step 2.

- The MarketSite Trading Partner ID (TPID).

5. In your Sterling Multi-Channel Selling Solution, **BusinessRule.xml** configuration file, set the CommerceOneIdentity element value to the TPID.

CHAPTER 18

Message Conversion

This chapter presents an overview of how the Sterling Multi-Channel Selling Solution manages to support different external message families such as dXML, RosettaNet, and cXML. Readers of this chapter should be familiar with Java, XML, XSLT, and the message families used by their implementations of the Sterling Multi-Channel Selling Solution.

Overview

When the Sterling Multi-Channel Selling Solution receives a request that is posted in the form of an XML message, it must process the request appropriately. It does this by converting the message into an internal XML message that conforms to the Sterling XML standard, and then processes the message using the message type to determine which controller, BLC, and JSP pages are used.

For example, if an inbound XML message is a dXML PriceAvailabilityRequest, then prior to passing the message to be processed by a controller, then the Sterling Multi-Channel Selling Solution converts the message to a Comergent XML PriceAvailabilityRequest.

Conversely, when the Sterling Multi-Channel Selling Solution posts a message such as a price and availability request to an external system, it must compose the message in the format recognized by the external system. If the external system requires RosettaNet, then the Sterling Multi-Channel Selling Solution must convert the internal representation of the message in Sterling XML to an XML document that conforms to the appropriate RosettaNet DTD.

The Sterling Multi-Channel Selling Solution uses the XSLT (XSL Transformation) technology to perform the conversion. To do so, it invokes a number of classes described in "Inbound Message Processing" on page 288.

Message Families

The Sterling Multi-Channel Selling Solution uses the following terminology:

- Message family: a family of messages is typically used to define a standard such as RosettaNet, Commerce One's xCBL, Ariba's cXML, or Sterling Commerce's dXML.
- Message version: each family can support more than one version: this is the set of message types that comprise a particular release of the message family: the current dXML version is 3.0.
- Message category: a specific message family and message version together define a set of messages: this is referred to as the message category. It must be possible to list all of the message types supported in particular message category.
- Message type: each message has a message type. Typically, it expresses the purpose of the message such as PriceAvailabilityGet or OrderCancelRequest.

Supported Conversions

Currently, the Sterling Multi-Channel Selling Solution supports the following conversions between message types:

TABLE 18. Supported Message Type Conversions

To / From	Comergent 1.0	Comergent 2.0	cXML	dXML 4.1.1	RosettaNet	xCBL
Comergent 1.0						
Comergent 2.0				X		
cXML				X		
dXML 4.1.1	X	X	X	X	X	X

TABLE 18. **Supported Message Type Conversions (Continued)**

To / From	Comergent 1.0	Comergent 2.0	cXML	dXML 4.1.1	RosettaNet	xCBL
RosettaNet				X		
xCBL				X		

To implement a new conversion standard, see "Implementing a New Mapping" on page 285.

Implementing a New Mapping

Suppose that you must support a new messaging standard called MSML. You will communicate with a partner whose partner server is set up to receive MSML messages and it will return MSML messages in response that you must process.

For each MSML message type that you want to create or process, you must map its content to a corresponding Comergent XML 3.0 message type. For example, suppose that you must send a MSML PriceAvailabilityGet message out to the partner server, and must process the returned MSML PriceAvailabilityPut message.

You must do the following:

1. Create a new MessageCracker class for the MSML message family. You can extend the abstract class, AbstractMessageCracker. Overwrite the *getMessageFamily()*, *getMessageVersion()*, *getMessageCategory()* and *getMessageType()* methods with methods appropriate to the MSML message family.
2. Create a new entry in the **MessageCrackerMap.xml** file for this new message family. It should be of the form:

```
<URLExt Name="msml">
    <ContentType Name="text/msml">
        <MessageCrackerImpl>
            com.comergent.dcm.messaging.MSMLMessageCracker
        </MessageCrackerImpl>
        <ControllerImpl>
            com.comergent.dcm.messaging.MessagingController
        </ControllerImpl>
        <Request>
            com.comergent.dcm.messaging.XMLRequest
        </Request>
        <Response>
            com.comergent.dcm.messaging.XMLResponse
        </Response>
    </ContentType>
</URLExt>
```

The value of the URLExt Name attribute is used so that this element is used if an inbound message is posted to a URL that looks like this:
`http://server:port/Sterling/msg/matrix/msml`

Note that you may not need to declare a separate URLExt element if the content type of the MSML message is set to be unique amongst the content types processed by the Sterling Multi-Channel Selling Solution.

3. In the ***debs_home*/Sterling/WEB-INF/converters/ConverterMap.xml** file, add the following element to the OutboundConverters element:

```
<Converter>
    <MessageCategory>Comergent_3.0</MessageCategory>
    <MessageMap>
        WEB-INF/converters/Comergent_3_0_to_MSML.xml
    </MessageMap>
    <ConvertedMessageCategory>
        MSML
    </ConvertedMessageCategory>
</Converter>
```

4. In the ***debs_home*/Sterling/WEB-INF/converters/ConverterMap.xml** file, add the following element to the InboundConverters element:

```
<Converter>
    <MessageCategory>MSML</MessageCategory>
    <MessageMap>
        WEB-INF/converters/MSML_to_Comergent_3_0.xml
    </MessageMap>
    <ConvertedMessageCategory>
        Comergent_3.0
    </ConvertedMessageCategory>
</Converter>
```

5. Create two files: **Comergent_3_0_to_MSML.xml** and **MSML_to_Comergent_3_0.xml**. These files follow this format:

```
<?xml version="1.0" encoding="UTF-8"?>
<MessageMap>
    <MessageCategory>Comergent_3.0</MessageCategory>
    <ConvertedMessageCategory>MSML</ConvertedMessageCategory>
    <AllowNullConversion>0</AllowNullConversion>
    <MessageType Name="PriceAvailabilityRequest">
        <ConvertedMessageType>
```

```
                PriceAvailabilityGet
            </ConvertedMessageType>
            <ConverterImpl>
                com.comergent.dcm.msgconverter.ConverterImpl
            </ConverterImpl>
            <Stylesheet>
                WEB-INF/stylesheets/cmgt30tomsml_pa_req.xsl
            </Stylesheet>
        </MessageType>
</MessageMap>
```

and

```
<?xml version="1.0" encoding="UTF-8"?>
<MessageMap>
    <MessageCategory>MSML</MessageCategory>
    <ConvertedMessageCategory>
        Comergent_3.0
    </ConvertedMessageCategory>
    <AllowNullConversion>1</AllowNullConversion>
    <MessageType Name="PriceAvailabilityPut">
        <ConvertedMessageType>
            PriceAvailabilityReply
        </ConvertedMessageType>
        <ConverterImpl>
            com.comergent.dcm.msgconverter.ConverterImpl
        </ConverterImpl>
        <Stylesheet>
            WEB-INF/stylesheets/msmltocmgt30_pa_rep.xsl
        </Stylesheet>
    </MessageType>
</MessageMap>
```

Here we assume that the existing Converter class, ConverterImpl, will suffice to perform the conversion.

The AllowNullConversion element is used to specify whether a message can be passed through untransformed if there is no MessageType listed for a particular message.

- “0” means that messages are not passed through if there is no MessageType element corresponding to the message type of a particular message.
- “1” means that a message may be passed through untransformed if its MessageType is not included in this file.

The default value for this element is “1”.

6. Create XSLT mapping files, **cmgt30tomsml_pa_req.xsl** and **msmltocmgt30_pa_rep.xsl**, that map elements to elements along these lines:

```
<xsl:template match="PriceAvailability">
<OrderType>
 <xsl:value-of select="/Comergent/PriceAvailability/OrderType"/>
</OrderType>
<CurrencyCode>
 <xsl:value-of select="/Comergent/PriceAvailability/CurrencyCode"/>
</CurrencyCode>
<SellerKey>
 <xsl:value-of select="/Comergent/PriceAvailability/SellerKey"/>
</SellerKey>
<SellerName>
 <xsl:value-of select="/Comergent/PriceAvailability/SellerName"/>
</SellerName>
<SellerLocation>
 <xsl:value-of select="/Comergent/PriceAvailability/SellerLocation"/>
</SellerLocation>
<StatusCode>
 <xsl:value-of select="/Comergent/PriceAvailability/StatusCode"/>
</StatusCode>
<StatusMessage>
 <xsl:value-of select="/Comergent/PriceAvailability/StatusMessage"/>
</StatusMessage>
<InResponseToID>
 <xsl:value-of select="/Comergent/PriceAvailability/InResponseToID"/>
</InResponseToID>
</xsl:template>
```

You create the exact mapping of element to element by examining the two DTDs to identify corresponding elements.

Inbound Message Processing

This section provides a brief description of the processing of an inbound message as it is received by the Sterling Multi-Channel Selling Solution.

1. Typically, messages posted to the Sterling Multi-Channel Selling Solution are sent using a URL that is mapped to the MessagingServlet.

   For example, suppose that you want the Messaging Servlet to process requests sent to http://<server>:<port>/Sterling/msg/matrix. Set the following lines in the **web.xml** file:

```
<servlet>
    <servlet-name>MessagingServlet</servlet-name>
    <servlet-class>
        com.comergent.dcm.messaging.MessagingServlet
    </servlet-class>
    <init-param>
        <param-name>browserCache</param-name>
        <param-value>true</param-value>
        <description>
            This can disable client-side caching of pages
        </description>
    </init-param>
</servlet>
```

and

```
<servlet-mapping>
    <servlet-name>MessagingServlet</servlet-name>
    <url-pattern>/msg/matrix/*</url-pattern>
</servlet-mapping>
```

2. The MessagingServlet class is a sub-class of the DispatchServlet class. It overrides the *createController()*, *createRequest()*, and *createResponse()* methods. The MessagingServlet first creates ComergentRequest and ComergentResponse objects to wrap the request and response objects received from the servlet container. To do so, it uses a MessageCrackerEntry and a MessageCracker to read the inbound message and identify its message family, message version, and message type.

   Both the MessageCrackerEntry and MessageCracker are created by the MessageCrackerFactory. This class inspects the URL and content type of the request to determine which MessageCrackerEntry and MessageCracker must be created. The MessageCrackerFactory class uses the **MessageCrackerMap.xml** file to map each specific combination of URL extension and content type to target request types and response types. It also specifies which controller should first be used to process the request.

   A typical entry in **MessageCrackerMap.xml** looks like this:

```
<URLExt Name="">
<ContentType Name="application/x-icc-xml">
    <MessageCrackerImpl>
        com.comergent.dcm.messaging.ComergentMessageCracker
    </MessageCrackerImpl>
    <ControllerImpl>
        com.comergent.dcm.core.MessagingController
    </ControllerImpl>
    <Request>com.comergent.dcm.messaging.XMLRequest</Request>
    <Response>com.comergent.dcm.messaging.XMLResponse</Response>
</ContentType>
</URLExt>
```

In this example, any request whose content type is “application/x-icc-xml” is cracked using the ComergentMessageCracker and the classes used for the controller, request, and response are MessagingController, XMLRequest, and XMLResponse respectively.

Note that the XMLRequest and XMLResponse classes provide a XMLRequestAccessor and XMLResponseGenerator class respectively. These classes provide convenience methods to get and set data elements.

3. Once the appropriate controller and ComergentRequest and ComergentResponse classes are created, the *execute()* method of the DispatchServlet is invoked and the processing of the request proceeds using the standard architecture of the Sterling Multi-Channel Selling Solution.

Generating an Outbound Message

This section describes what happens when a Sterling Multi-Channel Selling Solution application sends a message to an external system. See "Sending XML Messages to an External System" on page 331 for a more detailed example.

1. The application invokes the *restore()* method on a business object such as a shopping cart transfer request whose recipes defines its datasource as “MESSAGE”.
2. The MsgService element of the **DataServices.xml** file determines which class is used to restore the business object: by default it is com.comergent.dcm.dataservices.MsgService. The DataManager calls this class.
3. The DataManager identifies the URL to which the message should be posted. Typically, this is a URL specific to the intended partner recipient. In addition, the DataManager identifies the message category to use to send the message.
4. The OutboundConverters section of the **ConverterMap.xml** file determines which MessageMap file is to be used to perform the conversion from the internal Comergent format (Comergent 3.0) to the required external format.
5. The MessageMap file determines the following:
   - The outbound message type
   - The class used to perform the conversion
   - The XSL stylesheet to be used to perform the conversion

6. The MsgService class invokes the conversion class to generate the XML message and posts it to the specified URL.

Extrinsic Elements

You can use Extrinsic elements to extend the message definitions for both inbound and outbound messages that conform to the dXML 4.1 message family. Each Extrinsic element that you add to a message provides a name-value pair: this can be used to provide additional information in the message.

Each Extrinsic element must conform to the **Extrinsic.dtd** DTD whose basic form is:

```
<!ELEMENT Extrinsic (#PCDATA)>
<!ATTLIST Extrinsic
    name CDATA #REQUIRED
>
```

An example of using Extrinsic elements is provided in the handling of catalog exports. See the *Sterling Multi-Channel Selling Solution Developer Guide* for more information.

Inbound Processing

When an inbound dXML message is received, it is transformed into a Comergent XML document. If the inbound message contains an Extrinsic element, then it is turned into an element whose name is the Name attribute of the Extrinsic element and whose value is the value of the Extrinsic element. For example:

```
<Extrinsic Name="PackingUnit">Crate</Extrinsic>
```

is transformed into:

```
<PackingUnit>Crate</PackingUnit>
```

Outbound Processing

You can mark a field of a data object as Extrinsic by invoking the *setExtrinsic(true)* method of the MetaData class. When the data object is turned into a dXML message, the corresponding field is declared as an Extrinsic element. For example, suppose that there is a PackingUnit field of the Product data object., then the following code would result in the value of the PackingUnit element being declared as an Extrinsic element:

```
m_productBean.setPackingUnit("Crate");
MetaData temp_MetaData = m_productBean.getMetaData("PackingUnit");
temp_MetaData.setExtrinsic(true);
```

If the product data bean is used to generate a dXML message, then the resulting dXML message will include:

```
<Extrinsic Name="PackingUnit">Crate</Extrinsic>
```

CHAPTER 19

Implementing Order Management Integration

This chapter describes the steps required to integrate the Sterling Multi-Channel Selling Solution order management system with an ERP system.

Simple ERP Integration

When users place orders in the Sterling Multi-Channel Selling Solution, you may want to immediately send these orders to an ERP system or some other external system so that they can be processed. The Sterling Multi-Channel Selling Solution can be configured so that orders are HTTP posted to a specified URL as XML messages. The receiving HTTP server can process the messages as appropriate. For example:

- The receiving HTTP server could be an instance of the Sterling Message Broker server. The Sterling Message Broker provides a powerful and flexible message-based processor that can transform and dispatch XML documents using a variety of protocols, including XSLT and EDI.
- The receiving HTTP server could be a server running on the external ERP system.

The setup steps to set up the integration with an external HTTP server are:

1. Log in to the Sterling Multi-Channel Selling Solution as an enterprise administrator.

2. Using the Profile Manager, create a partner to represent the ERP system. Make a note of its partner key. Set its message URL to the URL used to post the XML messages from the Sterling Multi-Channel Selling Solution to the ERP system and set the XML Message Version to “Native”.
3. Using the System Administration System Properties page (accessed by clicking the System Services link in the enterprise administrator’s System Administration pane), set the Sterling Integrator **ERPIntegrationUrl** system administration property to the partner key of the partner created in Step 2.
4. Using the System Administration System Properties page, set the Sterling Orders **Send Orders XML msgs to ERP** system administration property to “true”.
5. Set the Sterling Orders **DEBSAdminForERP_User** and **DEBSAdminForERP_Password** properties to the username and password of an enterprise administrator. These values are used to initiate the cron job that checks for unsent orders and is used to authenticate the messages to the ERP system.

As direct commerce users place orders, their orders should be posted to the ERP system URL in the form of an ERPCreateOrderRequest message.

- If the message is successfully posted, then the integration status of the order is set to “Y”, but the order status field is left at Order Submitted until the ERP system responds to accept (or reject) the order.
- If for any reason the message is not successfully posted, then the integration status of the order is set to “N”.

6. A cron job can be set up to regularly check for orders that have not yet been successfully posted to the ERP system, and to re-try posting them. The cron job should be set up as a system cron job and its class set to “com.comergent.apps.orderMgmt.orders.bizAPI.OrdersERPCron”. This class is provided as part of the Sterling Multi-Channel Selling Solution.

CHAPTER 20

Order and Return Management

As your customers create and place orders on your Sterling Multi-Channel Selling Solution, the state of each order is managed by the Sterling Orders application. This chapter describes how the state of an order and its order lines change in response to activities initiated by the customer, customer service representatives, and external systems.

In addition, returns are also managed using states to track return requests as they are processed. Customers may request to return items once they have received them and customer service representatives may also initiate return requests in response to customer requests. Requests to return items must be accepted by an external system before a customer can actually return the item.

Overview

Order States

An order starts in the “Open” state when it is created. Orders complete their movement through the system when either all of the order line items have shipped (the “Shipped” state) or when the order has been cancelled (the “Cancelled” state). All of the transitions to these states and other intermediate states are determined by the actions of users or by messages received from external systems.

Order states primarily reflect an aggregation of the states of the order lines that make up the order. For example, an order is in the “In process” state if all of its order lines have state “In process”. If an order line is added, then the order line is in the “Order submitted” and the order state reverts to “Order submitted” until the new order line state changes to “In process”. Similarly, when one or more order lines are in the “Partially shipped” state, the whole order is in the “Partially shipped” state. Only when all the order lines are in the “Shipped” state does the order move to this state.

For our reference implementation of the Sterling Multi-Channel Selling Solution, the state transition diagram is presented in Figure 14 on page 300. For example:

- If the order is in the “In process” state and the customer who created the order submits a request to cancel the order, then the state of the order is changed to “Cancel submitted”.
- If the order is in the “Partially shipped” state and the external system sends a ShipmentUpdate message, and if the order shipment is completed, then the state is updated to “Shipped”.

In general, only certain actions may be performed on an order when an order is in a particular state. For example, once a customer has created and submitted an order, the order is in the “Order submitted” state. Until the ERP system has responded with a StatusUpdate message to accept the order, the customer may not change their order. When the StatusUpdate message is received and processed, then the order state is changed to “In process”.

Order Line States

Each order has header information (such address and billing information) and one or more order lines. Each order line has a product ID, a quantity, shipping information, and an order line state. In general, possible order line states are a subset of the possible order states. As for orders, order lines may transition from one state to another in response to customer activity, actions by a customer service representatives, or in response to messages from an external system. For example:

- If the order line is in the “In process” state and the customer who created the order submits a request to delete the order line, then the state of the order is changed to “Cancel submitted”.
- If the order line is in the “Partially shipped” state and the external system sends a ShipmentUpdate message, and if the order line shipment is completed, then the state is updated to “Shipped”.

In general, only certain actions may be performed on an order line when an order line is in a particular state. For example, once a customer has created and submitted an order, the order line is in the "Order submitted" state. Until the ERP system has responded with a StatusUpdate message to accept the order and the particular order line, the customer may not change their order line. When the StatusUpdate message is received and processed, then the order line state is changed to "In process".

Return States

Each return request has header information (such address and billing information) and one or more return lines. Each return line has a product ID, a quantity, shipping information. Each return has a state, but individual return lines do not. Like order states, return states are maintained in the CMGT_LOOKUPS table.

As for orders, returns may transition from one state to another in response to customer activity, actions by a customer service representatives, or in response to messages from an external system. For example:

- If the return is in the "Open" state and the returns rules engine rejects the return request, then the state of the return is changed to "Denied by RulesEngine".
- If the return is in the "Submitted to ERP" state and the external system sends a StatusUpdate message to accept the return, then the state is updated to "Approved by ERP".

In general, only certain actions may be performed on a return when a return is in a particular state. For example, once a customer has created and submitted a return request, the return may be automatically processed by a return rules engine or it may be manually processed by a customer service representative.

Order Process Modeler

The Order Process Modeler is a mechanism to manage the business logic of the order management cycle. The Order Process Modeler manages the state transitions of an order as it proceeds through the stages after it has been placed by a user.

The Order Process Modeler makes use of the state machine architecture introduced in Release 6.4. This is described in greater detail in the *Sterling Multi-Channel Selling Solution Developer Guide*.

The state transitions for an order are defined in the **EnterpriseOrderStateMachine.xml** configuration file: this file determines what business logic is performed when an order is to be moved from one state to another, and specifies what actions should be performed if the transition is successful. For example, the following fragment from the **EnterpriseOrderStateMachine.xml** configuration file specifies what should happen to an order in the “Open” state when one of the valid inputs is received, and what state the order should be moved to if the transition is successful:

```
<State Name="Open" Start="true">
    <Description>This is the initial State in the Ordering flow.
    </Description>
    <InputList>
    <Input Name="ORDER INPUT USER PLACE"
        Roles="Partner.DirectCommerceUser;Registered.User;
            Enterprise.CustomerServiceRepresentative">
        <Description>This is the "Place" action.</Description>
        <NextState>Order Submitted</NextState>
        <ActionHandlerList>
            <ActionHandler>
    com.comergent.apps.orderMgmt.orders.bizAPI.OrderPlaceHandler
            </ActionHandler>
            <ActionHandler>
    com.comergent.apps.orderMgmt.orders.bizAPI.OrderPersistHandler
            </ActionHandler>
            <ActionHandler>
    com.comergent.apps.orderMgmt.orders.bizAPI.SaveDiscounts
            </ActionHandler>
            <ActionHandler>
    com.comergent.apps.orderMgmt.orders.bizAPI.WriteHistoryHandler
            </ActionHandler>
        </ActionHandlerList>
        <ActionEventList>
            <ActionEvent>OrderPlaceEmailEvent</ActionEvent>
        </ActionEventList>
    </Input>
    <Input Name="ORDER INPUT XML PLACE"
        Roles="Partner.DirectCommerceUser;Registered.User">
        <Description>This is the "Place" action.</Description>
        <NextState>Order Submitted</NextState>
        <ActionHandlerList>
            <ActionHandler>
com.comergent.apps.orderMgmt.orders.bizAPI.PreProcessXMLPlaceHandler
            </ActionHandler>
            <ActionHandler>
    com.comergent.apps.orderMgmt.orders.bizAPI.OrderPlaceHandler
            </ActionHandler>
            <ActionHandler>
    com.comergent.apps.orderMgmt.orders.bizAPI.OrderPersistHandler
            </ActionHandler>
            <ActionHandler>
    com.comergent.apps.orderMgmt.orders.bizAPI.SaveDiscounts
```

```
            </ActionHandler>
            <ActionHandler>
    com.comergent.apps.orderMgmt.orders.bizAPI.WriteHistoryHandler
            </ActionHandler>
        </ActionHandlerList>
            <ActionEventList>
            <ActionEvent>OrderPlaceEmailEvent</ActionEvent>
        </ActionEventList>
    </Input>
    </InputList>
</State>
```

The way to read this file is to start with the State element. In this example, the Name attribute of the State is "Open". If an order is in the "Open" state, and a request is received to process this order with the "ORDER INPUT USER PLACE" input, then the request is processed as follows:

1. The state machine verifies that "ORDER INPUT USER PLACE" is a valid input for an order in this state.
2. The Roles attribute of the Input element is checked to see that the user attempting the state change is entitled to do this. Note that the list of roles is delimited by semi-colons (";"), rather than commas.
3. The handler classes are invoked one after the other, in this order:
    a. com.comergent.apps.orderMgmt.orders.bizAPI.OrderPlaceHandler
    b. com.comergent.apps.orderMgmt.orders.bizAPI.OrderPersistHandlerc
    c. om.comergent.apps.orderMgmt.orders.bizAPI.SaveDiscounts
    d. com.comergent.apps.orderMgmt.orders.bizAPI.WriteHistoryHandler
4. If one of the handler classes throws an InputFailedException, then processing is halted, and the order remains in its current state. If all of the handlers process their *performInputAction()* method successfully, then the order is moved into the "Order Submitted" state, and an ActionEvent, OrderPlaceEmailEvent, is fired. This is processed by the EventBus: see the *Sterling Multi-Channel Selling Solution Developer Guide* for further information.

Reference Implementation Transitions

The following diagrams illustrate the state transitions supported by the reference implementation of the Sterling Multi-Channel Selling Solution.

Order States

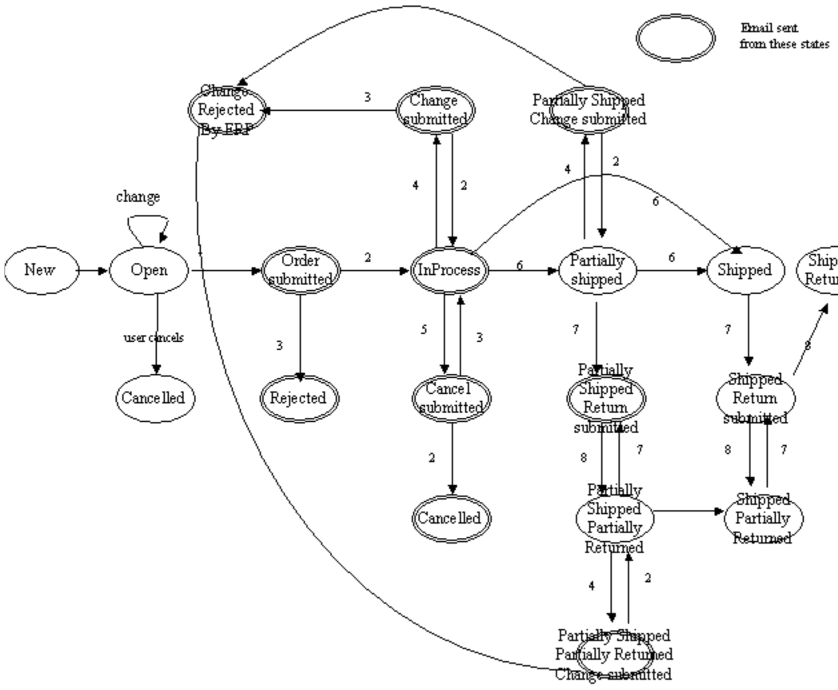

Email sent from these states
Change Rejected By ERP
Change submitted
Partially Shipped Change submitted
change
New
Open
Order submitted
InProcess
Partially shipped
Shipped
Shipped Returned
user cancels
Cancelled
Rejected
Cancel submitted
Cancelled
Partially Shipped Return submitted
Shipped Return submitted
Partially Shipped Partially Returned
Shipped Partially Returned
Partially Shipped Partially Returned Change submitted
1
2
3
4
5
6
7
8

FIGURE 14. **Order State Transitions**

The numbers in the diagram refer to the messages that are sent to an external system as the order moves from one state to another, or which are received from an external system and which cause a change in order state:

1. OrderCreate message sent to external system
2. StatusUpdate(accepted) message received from external system
3. StatusUpdate(rejected) message received from external system
4. OrderChange message sent to external system
5. OrderCancel message sent to external system
6. ShipmentUpdate message received from external system

7. ReturnCreate message sent to external system
8. RMAUpdate message received from external system

Order Line States

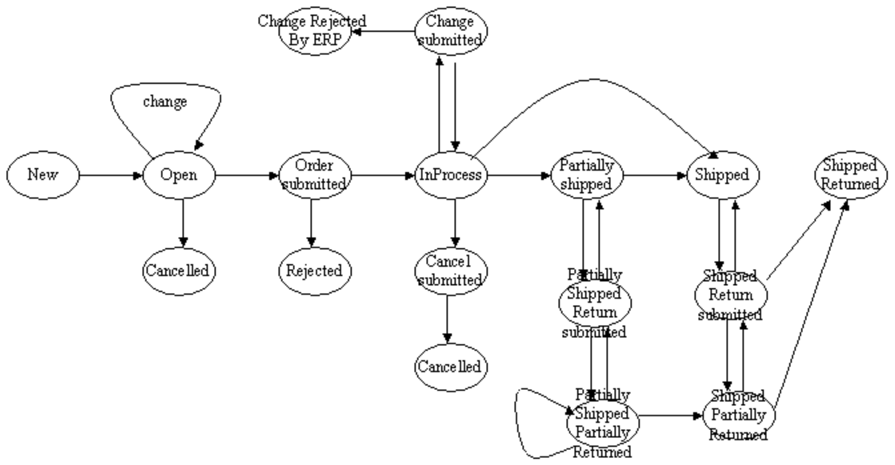


FIGURE 15. Order Line State Transitions

Return States

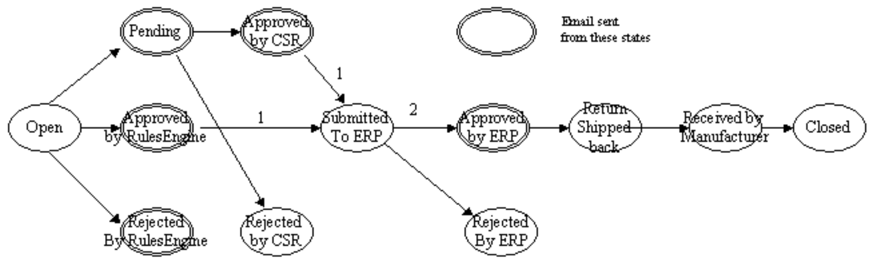


FIGURE 16. Return State Transitions

The numbers in the diagram refer to the messages that are sent to an external system as the return moves from one state to another, or which are received from an external system and which cause a change in return state:

1. Message sent to external system
2. RMAUpdate message received from external system

Setting Up Order and Return States

Order and return states are defined in the CMGT_LOOKUPS table. As orders and returns move from one state to another, the states must be present in this table. Each order state is identified by the string “order_status” in the LOOKUP_TYPE column. The DESCRIPTION column of the table identifies the value of the state: “Open”, “Cancel submitted”, and so on.

Similarly, each return state is identified by the string “return_status” in the LOOKUP_TYPE column and the value of the state is contained in the DESCRIPTION column: “Open”, “Pending”, etc. In addition, return reasons and return criteria are also defined in the CMGT_LOOKUPS table.

Order states are loaded into the CMGT_LOOKUPS table using the XMLLoader utility. See "Populating the Knowledgebase" on page 109 for more information regarding the use of this utility.

Each order state is created with a LightWeightLookup element in the **LightWeightLookupList** file of the form:

```
<LightWeightLookup state="INSERTED">
    <LookupType state="INSERTED">order_status</LookupType>
    <LookupCode state="INSERTED">1</LookupCode>
    <Description state="INSERTED">Shipped</Description>
    <Locale state="INSERTED">en_US</Locale>
    <Flag state="INSERTED">1</Flag>
</LightWeightLookup>
```

Controllers and BLCs

The business logic to manage orders and returns is implemented through the Sterling Orders and Sterling Returns applications. These applications include sets of controllers and business logic classes (BLCs). See the *Sterling Multi-Channel Selling Solution Developer Guide* for more information about the general architecture of the Sterling Multi-Channel Selling Solution and how controllers and BLCs are used.

Sterling Orders

Controllers

All the Sterling Orders controller classes extend the ForwardController class. In particular, the SimpleController class extends the ForwardController and many of the Sterling Orders controllers extend the SimpleController.

BLCs

All the Sterling Orders BLCs extend the basic BLC class. The BLCs used to respond to message requests from an external ERP system extend the CommonOrderXMLBLC class which is a subclass of the basic BLC class.

Sterling Returns

Controllers

All the Sterling Returns controller classes extend the SimpleController class.

BLCs

All the Sterling Returns BLCs extend the basic BLC class.

Messages to and from External Systems

All of the messages exchanged between the Sterling Multi-Channel Selling Solution and an external ERP system are defined in the dXML message family. See the *Sterling Multi-Channel Selling Solution Reference Guide* for more information about the dXML DTDs.

The following outbound messages from the Sterling Multi-Channel Selling Solution must be processed by the ERP system:

- ERPOrderCreate
- ERPOrderChange
- ERPOrderCancel
- OrderCancelResponse
- OrderChangeResponse
- OrderCreateResponse

The following inbound messages may be received from an external ERP system:

- OrderAcknowledgement
- OrderCancelRequest
- OrderChangeRequest
- OrderCreateRequest
- RMAUpdateRequestFromERP
- ShipmentUpdateFromERPRequest
- StatusUpdateFromERPRequest

Example Messages

This section provides examples of the message types listed above.

OrderCancelRequest

```
<Comergent>
    <MessageHeader>
        <MessageType>OrderCancelRequest</MessageType>
        <MessageVersion>3.0</MessageVersion>
        <MessageID/>
        <SessionID/>
    </MessageHeader>
    <RemoteUser>
        <UserLogin>mscott</UserLogin>
        <UserFullName/>
        <UserAuthenticator>mscott</UserAuthenticator>
    </RemoteUser>
    <Order type="BusinessObject">
        <OrderKey>249e11554246efbb</OrderKey>
        <ForwardToERP>Y</ForwardToERP>
    </Order>
</Comergent>
```

OrderChangeRequest

```
<Comergent>
    <MessageHeader>
        <MessageType>OrderChangeRequest</MessageType>
        <MessageVersion>3.0</MessageVersion>
        <MessageID/>
        <SessionID/>
    </MessageHeader>
    <RemoteUser>
        <UserLogin>mscott</UserLogin>
        <UserFullName/>
        <UserAuthenticator>mscott</UserAuthenticator>
    </RemoteUser>
    <Order type="BusinessObject">
        <OrderKey>24814acc546adc5f</OrderKey>
        <ForwardToERP>Y</ForwardToERP>
        <LineItemList>
            <LineItem>
                <ChangeOperation>INSERTED</ChangeOperation>
                <SKU>MX-LNXA</SKU>
                <Quantity>55</Quantity>
                <OrderLineItemShipOrderAddress>
                    <FirstName>Bill</FirstName>
                </OrderLineItemShipOrderAddress>
            </LineItem>
        </LineItemList>
    </Order>
</Comergent>
```

OrderCreateRequest

```
<Comergent>
    <MessageHeader>
        <MessageType>OrderCreateRequest</MessageType>
        <MessageVersion>3.0</MessageVersion>
        <MessageID/>
        <SessionID/>
    </MessageHeader>
    <RemoteUser>
        <UserLogin>mscott</UserLogin>
        <UserFullName/>
        <UserAuthenticator>mscott</UserAuthenticator>
    </RemoteUser>
    <Order state="INSERTED" type="BusinessObject">
        <ShoppingCartKey>21</ShoppingCartKey>
        <EmailAddress>sborra@comergent.com</EmailAddress>
        <PaymentType>2</PaymentType>
        <LastName>bbbbaaaaa</LastName>
        <PONumber>2</PONumber>
        <OrderDate>2001-06-18 15:20:12.0</OrderDate>
        <OrderCurrencyCode>0</OrderCurrencyCode>
        <CreditCardType/>
        <CreditCardHolder>cch</CreditCardHolder>
        <PaymentExpirationDate>
            2001-12-31 0:0:0.0
        </PaymentExpirationDate>
        <FirstName/>
        <PhoneNumber/>
        <ShippingMethod/>
        <PaymentNumber/>
        <ForwardToERP>Y</ForwardToERP>
        <ShipOrderAddress state="INSERTED">
            <AddressKey/>
            <AddressType/>
            <FirstName/>
            <LastName/>
            <Title/>
            <CompanyName/>
            <MailStop/>
            <Country/>
            <Address1/>
            <Address2/>
            <City/>
            <PostalCode/>
            <State/>
            <County/>
            <RefNum/>
        </ShipOrderAddress>
        <SoldOrderAddress state="INSERTED">
```

```
            <AddressKey/>
            <AddressType/>
            <FirstName/>
            <LastName/>
            <Title/>
            <CompanyName/>
            <MailStop/>
            <Country/>
            <Address1/>
            <Address2/>
            <City/>
            <PostalCode/>
            <State/>
            <County/>
            <RefNum/>
        </SoldOrderAddress>
        <BillOrderAddress state="INSERTED">
            <AddressKey/>
            <AddressType/>
            <FirstName/>
            <LastName/>
            <Title/>
            <CompanyName/>
            <MailStop/>
            <Country/>
            <Address1/>
            <Address2/>
            <City/>
            <PostalCode/>
            <State/>
            <County/>
            <RefNum/>
        </BillOrderAddress>
        <LineItemList>
            <LineItem state="INSERTED">
                <Description/>
                <Quantity>30</Quantity>
                <ConfigFlag/>
                <ConfigContainer/>
                <UnitOfMeasure/>
                <TransferredListPrice/>
                <TransferStatus/>
                <SKU>MX-LNXA</SKU>
                <SKUAuthority>10020</SKUAuthority>
            </LineItem>
        </LineItemList>
    </Order>
</Comergent>
```

ReturnCreateRequest

```
<?xml version="1.0" encoding="UTF-8"?>
<Comergent>
    <MessageHeader>
        <MessageType>ReturnCreateRequest</MessageType>
        <MessageVersion>2.0</MessageVersion>
        <MessageID/>
        <SessionID/>
    </MessageHeader>
    <RemoteUser>
        <UserLogin>mscott</UserLogin>
        <UserFullName/>
        <UserAuthenticator>mscott</UserAuthenticator>
    </RemoteUser>
    <OneReturn type="BusinessObject" state="INSERTED">
        <OrderKey>91598c1c53129067</OrderKey>
        <ReturnLineItemList state="INSERTED">
            <ReturnLineItem state="INSERTED">
            <ReturnLineKey>10047</ReturnLineKey>
            <QuantityReturned state="INSERTED">1</QuantityReturned>
            <ReturnReason state="INSERTED">1</ReturnReason>
            <ReturnCriterion state="INSERTED">1</ReturnCriterion>
            </ReturnLineItem>
        </ReturnLineItemList>
    </OneReturn>
</Comergent>
```

RMAUpdateRequestFromERP

```
<?xml version="1.0" encoding="UTF-8" ?>
<Comergent>
    <MessageHeader>
        <MessageType>RMAUpdateFromERPRequest</MessageType>
        <MessageVersion>2.0</MessageVersion>
        <MessageID/>
        <SessionID/>
    </MessageHeader>
    <RemoteUser>
        <UserLogin>ajones</UserLogin>
        <UserFullName/>
        <UserAuthenticator>ajones</UserAuthenticator>
    </RemoteUser>
    <RMAUpdate type="BusinessObject" state="INSERTED">
        <ReturnKey>201</ReturnKey>
        <RMANumber>2987</RMANumber>
        <Title>Mr</Title>
        <FirstName>fname</FirstName>
        <LastName>llname</LastName>
        <CompanyName>ccname</CompanyName>
        <Address1>llline1</Address1>
        <Address2>llline2</Address2>
        <City>boston</City>
        <PostalCode>98765</PostalCode>
        <State>MA</State>
        <Country>USA</Country>
        <ShippingMethod>Fedex</ShippingMethod>
    </RMAUpdate>
</Comergent>
```

ShipmentUpdateFromERPRequest

```
<?xml version="1.0" encoding="UTF-8" ?>
    <Comergent>
    <MessageHeader>
        <MessageType>ShipmentUpdateFromERPRequest</MessageType>
        <MessageVersion>2.0</MessageVersion>
        <MessageID/>
        <SessionID/>
    </MessageHeader>
    <RemoteUser>
        <UserLogin>ajones</UserLogin>
        <UserFullName/>
        <UserAuthenticator>ajones</UserAuthenticator>
    </RemoteUser>
    <ShipmentUpdate type="BusinessObject" state="INSERTED">
        <OrderKey>579e752d946aafb1</OrderKey>
            <ShipmentUpdateLineItemList>
                <ShipmentUpdateLineItem>
                    <LineKey>10075</LineKey>
                    <ShipmentUpdateSerialLineItemList>
                    <ShipmentUpdateSerialLineItem>
                    <ShippedSerialNumber>s11</ShippedSerialNumber>
                    </ShipmentUpdateSerialLineItem>
                    <ShipmentUpdateSerialLineItem>
                    <ShippedSerialNumber>s12</ShippedSerialNumber>
                    </ShipmentUpdateSerialLineItem>
                </ShipmentUpdateSerialLineItemList>
            </ShipmentUpdateLineItem>
        </ShipmentUpdateLineItemList>
    </ShipmentUpdate>
</Comergent>
```

StatusUpdateFromERPRequest

```
<Comergent>
    <MessageHeader>
        <MessageType>StatusUpdateFromERPRequest</MessageType>
        <MessageVersion>2.0</MessageVersion>
        <MessageID/>
        <SessionID/>
    </MessageHeader>
    <RemoteUser>
        <UserLogin>mscott</UserLogin>
        <UserFullName/>
        <UserAuthenticator>mscott</UserAuthenticator>
    </RemoteUser>
    <ERPStatusUpdate type="BusinessObject" state="INSERTED">
        <OrderKey>24814acc546adc5f</OrderKey>
        <ERPOrderNumber>876</ERPOrderNumber>
        <ERPResponse>Accepted</ERPResponse>
    </ERPStatusUpdate>
</Comergent>
```

CHAPTER 21

Integrating with External Data Sources

To integrate the Sterling Multi-Channel Selling Solution with your existing e-commerce systems, you must provide the Sterling Multi-Channel Selling Solution with the information necessary to retrieve information from external data sources. This can involve working with the following:

- Sterling Schema
- DataService Classes
- Sales Tax and Shipping Charges Calculation
- Credit Card Authorization
- Loading Invoices
- LDAP Authentication
- Implementing a Single Sign-On Solution
- RosettaNet Setup Steps
- Sending XML Messages to an External System
- Trading Partner Enrollment Form

Sterling Schema

The Sterling Multi-Channel Selling Solution uses the concept of *data object* to encapsulate the data that the business logic classes use. Data objects correspond to the business entities in the system such as partners, products, shopping carts, and so on. An introduction to data objects is provided in the *Sterling Multi-Channel Selling Solution Developer Guide* and is covered in more depth in the *Sterling Multi-Channel Selling Solution Reference Guide*.

With the installation of the Sterling Multi-Channel Selling Solution, standard data objects are created for your use. If you want to create additional data objects, or modify and extend the current data objects, then you also need to run the DTD and Bean generation targets. See "Data Objects" on page 151 for further information.

DataService Classes

The DataService attribute of the Primary and Alternate elements of a DataSource element determines the Java class that is invoked to execute the connection to the external service. For example, if DataService="JdbcService", then the JDBCService class is invoked to interact with the database whose connection string is defined by ConnectionString="*connectionString*".

The mapping between strings used to define data services and the classes invoked is maintained in the **DataServices.xml** configuration file. For example, the following elements ensure that when the DataService attribute is "JdbcService", then the JDBCService class is used to access the external data source.

```
<JdbcService controlType="text" runtimeDisplayed="true"
    ChangeOnlyAtBootTime="true" visible="true" boxsize="45"
    displayQuestion="JdbcService class name"
    defaultChoice="com.comergent.dcm.dataservices.JDBCService"
    help="Enter the DataService class that extends the DataService
    abstract class provided. This class is used to support accessing
    data sources that are JDBC-complaint databases.">
    com.comergent.dcm.dataservices.JDBCService
</JdbcService>
<MsgService controlType="text" runtimeDisplayed="true"
    ChangeOnlyAtBootTime="true" visible="true" boxsize="45"
    displayQuestion="MsgService class name"
    defaultChoice="com.comergent.dcm.dataservices.MsgService"
    help="Enter the DataService class that extends the DataService
    abstract class provided. This class is used to support accessing
    data sources that are NOT DBC-complaint(Messaging source).">
    com.comergent.dcm.dataservices.MsgService
```

```
</MsgService>
```

Similarly, when the DataService attribute is “MsgService”, then the MsgService class is used to access the external data source.

If the primary data service is unavailable, then the alternate data service is used. This data service might use a different Java class and a different means of connecting to the external source. For example, it might emulate a browser by connecting to a URL on an external system and retrieve information from the returned HTML page. See "HTMLService Class" on page 340 for more information on how to write a DataService class that retrieves data in this manner.

In general, the Java class used as the DataService is a sub-class of the abstract class DataService provided by the Sterling Multi-Channel Selling Solution. Two examples of such sub-classes are provided by JDBCService and MsgService that are used to handle JDBC database connections and messages respectively.

In general, the DataService classes use the methods *persist()* and *restore()* to save data from the Sterling Multi-Channel Selling Solution into a persistent data store and to retrieve data from the data store into the Sterling Multi-Channel Selling Solution respectively. See the *Sterling Multi-Channel Selling Solution Reference Guide* for further details on how to implement a DataService sub-class. See "Sending XML Messages to an External System" on page 331 for an example of using the message service.

Sales Tax and Shipping Charges Calculation

The Sterling Multi-Channel Selling Solution has basic support for calculating sales tax information and shipping information.

- If you do not set the useExternalTaxSystem element to “true”, then sales tax is calculated using the tax percentages set in the Knowledgebase CMGT_LOCAL_TAX table. See the *Sterling Multi-Channel Selling Solution Reference Guide* for further information.
- Shipping charges are retrieved from the Knowledgebase CMGT_LOCAL_SHIP_COST table. See the *Sterling Multi-Channel Selling Solution Reference Guide* for further information.

Shipping Tracking Integration

It is possible to use the Sterling Multi-Channel Selling Solution to track the shipping status of orders. This makes use of the ShipmentTrackingUrl properties for each shipping method. For each shipping method you can specify a URL: this URL is displayed on the end-user visible page as a link. The tracking number of the specific order line item is appended to this URL.

The URL is specified for each supported shipping method using the System Service -> Orders -> ShipmentTrackingUrl property accessible through the Web UI. In general, the form of each URL assumes that the tracking number of a specific line item can be appended to the URL after the last "=" of the URL.

Tracking numbers should be provided by the back-end system that sends OrderShipmentUpdateRequest XML messages into the Sterling Multi-Channel Selling Solution. Each such message uses the ShipCarrier element to specify the carrier, and the TrackingNumber element to specify the tracking number for the line item. For example:

```
<Shipment>
    <OrderNumber>3021598401</OrderNumber>
    <ShipmentUpdateLineItemList>
        <ShipmentUpdateLineItem>
            <LineKey>600865</LineKey>
            <OrderShipmentUpdateInfo>
                <ShipmentDate>2006-9-22 5:30:0.0</ShipmentDate>
                <TrackingNumber>943279857432</TrackingNumber>
                <InvoiceNumber>12345</InvoiceNumber>
                <ShipCarrier>UPS</ShipCarrier>
            </OrderShipmentUpdateInfo>
            <ShipmentUpdateQuantityLineItem>
                <Quantity>1</Quantity>
            </ShipmentUpdateQuantityLineItem>
        </ShipmentUpdateLineItem>
    </ShipmentUpdateLineItemList>
    <GenerateInvoice>Y</GenerateInvoice>
    <Tax>18.95</Tax>
    <ShippingCharges>28.00</ShippingCharges>
    <MiscAdjustments>38.00</MiscAdjustments>
</Shipment>
```

Shipping data for each order line item is stored in the CMGT_ORDER_LI_SHIP table: the string stored in the SHIP_CARRIER column is used to access the corresponding tracking URL.

Credit Card Authorization

Credit card transactions are supported by means of the payment gateway mechanism: see CHAPTER 15, "Implementing a Payment Gateway". This section is provided for legacy purposes.

Note that the use of the useExternalCreditCardAuthorization business rule is no longer supported. Earlier releases, used to have the following configuration setting:

- If you do not set the useExternalCreditCardAuthorization element to "true", then credit card authorization is not performed.

The Sterling Multi-Channel Selling Solution supports the use of an external software product, CyberSource, to authenticate credit cards and to retrieve sales tax information. You can implement either or both functions. Follow these steps:

1. Purchase a license from CyberSource: this license must support either or both of credit card authorization and sales tax calculation. You should receive a merchant ID and a merchant reference number as part of the license. In the following we refer to these as *merchant_id* and *merchant_reference_number*. You should also be told the server URL to use in Step 6.
2. Download the Internet Commerce Suite (ICS) Software Development Kit. It is packaged in the form of a ZIP file: currently it is **SDK_4.0.3_java.zip**, but the version may change.
3. Extract the following files: **ics.jar**, **Ecert.class**, and **ICSClient.props**.
4. Run the Ecert class to generate the certificates, keys, and password files for your implementation. Enter:

   ```
   java Ecert <merchant_id>
   ```

   You may need to set the classpath to pick up all the required classes. The following files are created: ***merchant_id*.crt**, ***merchant_id*.pvt**, ***merchant_id*.pwd**, and **CyberSource_SJC_US.crt**.
5. Copy the files generated in Step 4 to ***debs_home*/Sterling/Web-INF/ properties/orderManagement/**.
6. Edit the ***debs_home*/Sterling/Web-INF/properties/orderManagement/ ICSClients.prop** file to enter the following properties:

   ```
   merchantID=merchant_id
   serverName=CyberSource_SJC_US
   serverURL=<CyberSource Server URL>
   myPrivateKey=<Absolute path to location of the merchant_id.pvt
       file>
   myCert=<Absolute path to location of the merchant_id.crt file>
   serverCert=<Absolute path to location of the
   ```

`CyberSource_SJC_US.crt file>`

The server URL must be the one that CyberSource provides as described in Step 1.

If you have followed the instructions above, then the location of the merchant_id.pvt file is ***debs_home*/Sterling/WEB-INF/properties/orderManagement/merchant_id.pvt**.

If you have followed the instructions above, then the location of your certificate file is ***debs_home*/Sterling/WEB-INF/properties/orderManagement/merchant_id.crt**.

If you have followed the instructions above, then the location of the server certificate is ***debs_home*/Sterling/WEB-INF/properties/orderManagement/CyberSource_SJC_US.crt**.

7. Copy **ics.jar** to ***debs_home*/Sterling/WEB-INF/lib/**.
8. Edit the orderCCauthorizationPropsFile element in **Comergent.xml** to point to the location of the **ICSClient.props** file. If you have followed the steps above, then this location is "*debs_home*/Sterling/WEB-INF/properties/orderManagement/ICSClient.props".
9. In the **Comergent.xml** file, set the value of the merchantRefNumber element to the merchant reference number provided by CyberSource.
10. Enable credit card authorization and sales tax calculation as follows:
    a. To enable credit card authorization, set the following element in **BusinessRule.xml** to "true": useExternalCreditCardAuthorization.
    b. To enable sales tax calculation, set the following element in **BusinessRule.xml** to "true": useExternalTaxSystem.
11. Restart the servlet container.

Customization

The default implementation of credit card authorization and sales tax calculation may not meet all your needs. This section covers the main topics that you need to consider when performing customization.

Currently, the methods to perform credit card authorization and sales tax calculation are invoked in the OrderInquiryList class that implements the bizAPI for processing an order. In turn, this uses the OILCreditCard and OILTaxes classes to invoke the ICS classes.

The principal ICS classes used are:

- ICSClient: the main class used to perform the request. Its *ICSClient(String s)* constructor is used to pass in the location of the **ICSClient.props** file.
- ICSClientRequest: use this class to invoke the ICSClient *reply()* method. Its ICSClientRequest(ICSClient icsc) constructor is used to populate the request with configuration information from the **ICSClient.props** file. Use its accessor methods to set the data fields: for example, *setMerchantRefNo()* to set the merchant reference number.

  You must use its *addApplication(String s)* to specify whether the request is for credit card authorization ("ics_auth") or for a sales tax calculation ("ics_tax").
- ICSReply: this class provides the response to the request. Its accessor methods provide the information you need. In particular, if *getReplyCode()* returns an integer less than or equal to zero, then the request failed and its *getErrorMessage()* method provides a reason for the failure.

The required fields to execute a request are provided in "Required Data" on page 317. Consult the ICS documentation for further information relating to the ICS API and its classes.

Required Data

To perform a credit card authorization, you must supply the following data:

- Merchant Reference Number: obtained from **Comergent.xml**
- Merchant ID: obtained from **ICSClient.props** file
- Customer First Name
- Customer Last Name
- Customer Email Address
- Customer Phone
- Bill Address1
- Bill Address2
- Bill City
- Bill State
- Bill Zip

- Bill Country
- Amount: must be at least 0.1 and greater than zero
- Customer Credit Card Number
- Customer Credit Card Expiration Month
- Customer Credit Card Expiration Year
- Currency

To perform a sales tax calculation, you must supply the following data:

- Merchant Reference Number (obtained from **Comergent.xml**)
- Merchant ID (obtained from **ICSClient.props** file)
- Bill Address1
- Bill Address2
- Bill City
- Bill State
- Bill Zip
- Bill Country
- Amount: can be 0.0

Loading Invoices

Sterling Invoicing provides the ability for your enterprise and your partners to interactively manage invoices. Invoices generated by an existing ERP system can be imported into the Sterling Multi-Channel Selling Solution and then both enterprise users and partner users can view and update the invoices through their browser. See the *Sterling Multi-Channel Selling Solution Administration Guide* for more information on how users can manage invoices.

To load invoices into the Sterling Multi-Channel Selling Solution, you must post them into the system as XML messages. The following message types are supported:

- InvoiceCreateRequest
- InvoiceChangeRequest
- OrderShipmentUpdateRequest: set the GenerateInvoice element to “Y”.

The message types are defined as dXML 4.1 DTDs: these are provided in the ***debs_home*/Sterling/WEB-INF/dXML/4.1/** directory.

Example messages can be provided on request: please contact your Sterling Commerce representative for further information.

A sample tool, called **dship.bat**, to post messages into the Sterling Multi-Channel Selling Solution can also be provided. Edit the file by changing the CONTAINER_HOME and SERVER_URL variables to point to your installation of the Sterling Multi-Channel Selling Solution, and then run the script at the command line.

LDAP Authentication

The Sterling Multi-Channel Selling Solution supports the use of LDAP (Lightweight Directory Access Protocol) for user authentication.

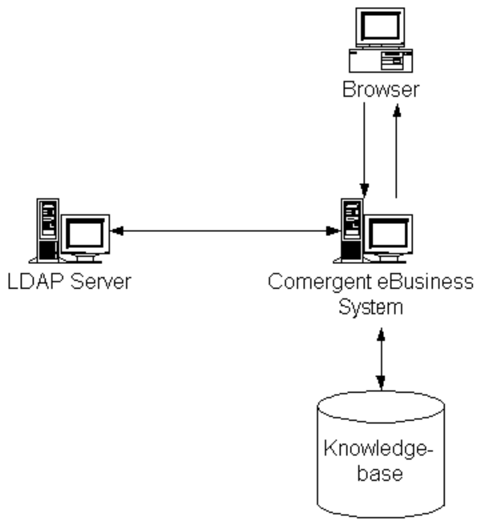


FIGURE 17. **LDAP Authentication**

When a user attempts to log into the Sterling Multi-Channel Selling Solution, they provide their username and password. If the Sterling Multi-Channel Selling Solution is implemented to use LDAP, then the authentication step is performed against the external LDAP server rather than against the Knowledgebase. For example, Sterling Multi-Channel Selling Solution UI functionality such as “forgot password" has no effect, since the authentication is performed against the LDAP server.

Only the tenant storefront (for example, Matrix Solutions) can authenticate users against the external LDAP server. Child storefronts authenticate users against the Knowledgebase.

To enable LDAP authentication, you log in as a tenant administrator user and set the LDAP properties on the System Services page. As a site administrator user, you also set a site-level property to identify the tenant storefront to the external LDAP server.

To set up LDAP Authentication

1. The Sterling Multi-Channel Selling Solution assumes that each username in the Sterling Multi-Channel Selling Solution system exists in the LDAP server. Ensure that, for each user created in the Sterling Multi-Channel Selling Solution, there is a corresponding user on the LDAP server with the same username and password as they use on the Sterling Multi-Channel Selling Solution.
2. Users must be set up using a base distinguished name (Base DN). Contact your LDAP server administrator to obtain the Base DN for your LDAP server.
3. As an administrator, set the following system properties:
    - LDAP BaseDN: set to the value that you obtained from your LDAP server administrator. For example, if your domain name is example.com:

      ```
      dc=example,dc=com
      ```
    - LDAP Authentication: set to "true".
    - LDAP Server URL: set to the URL used to access the LDAP server. Typically, this of the form:

      ```
          ldap://<ldapserver:port>
      ```
    - Security Authentication Type: set to "simple".
4. As a site administrator, log in to the Sterling Multi-Channel Selling Solution administration site. The URL is similar to:

   ```
   http://server:port/sterling/en/US/enterpriseMgr/admin
   ```
5. On the System Services page, set the Name of Tenant Namespace property to the storefront ID of the tenant. For example, the Matrix Solutions storefront ID is 100.

6. Restart the Sterling Multi-Channel Selling Solution servlet container.

Note the following:

- The out of the box LDAP implementation expects users to have Base DN’s that concatenate the following elements:
    - The Sterling user login (uid)
    - An organizational unit (ou), “People”
    - The BaseDN from the LDAP BaseDN property

  For example, the following is the DN for user ajones at matrix.com:
  ```
  uid=ajones,ou=People,dc=matrix,dc=com
  ```
  If the LDAP schema does not organize users in this way, then you must customize your implementation of the LDAP interface.
- You must use an existing base distinguished name (Base DN). Make sure that the Base DN element is set to the chosen name. As you create new users within the Sterling Multi-Channel Selling Solution, make sure that you create a user with the same username and password on the LDAP server.

Implementing a Single Sign-On Solution

In many implementations, the Sterling Multi-Channel Selling Solution is one of a number of systems that live together in the enterprise environment. Unless a single sign-on solution is implemented, users may have to log on to different systems as they need to access them, and may have to remember different usernames and passwords to authenticate themselves for each system. At the same time, enterprise system administrators have to maintain userids on each system, and so as users come and go and change their responsibilities, it becomes an administrative burden to maintain them all.

Consequently, implementing a single sign-on solution in the enterprise environment provides a way to manage users centrally in such a way that users can authenticate themselves once and then have access to all of the resources that they should. At the same time, it provides enterprise system administrators with a central location in which they can manage all their users and their entitlements. This section describes the basic framework for a single sign-on solution and how the Sterling Multi-Channel Selling Solution can be customized to participate in it.

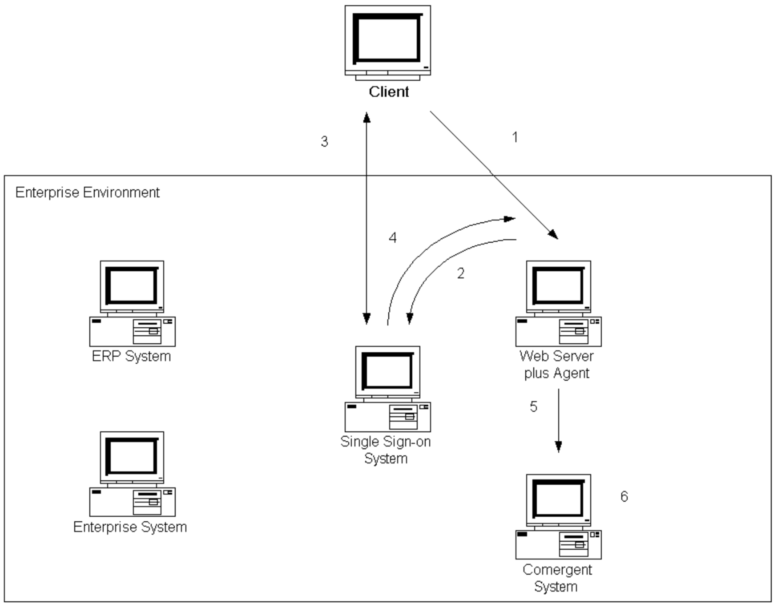


FIGURE 18. **Single Sign-On Environment**

Typically, an enterprise will implement a Single Sign-on solution by buying a single sign-on solution such as SiteMinder or Oblix, and delegate all authentication tasks to this server. Userids are maintained centrally on this system, and other systems (such as the Sterling Multi-Channel Selling Solution) must maintain their userid information to be consistent with this central repository. This process of synchronization is discussed more fully below: see "Profile Synchronization" on page 324.

The setup of the Single Sign-on system will vary with each single sign-on solution, but the basic framework is fairly similar. In most cases, agents of the Single Sign-on system are installed in front of each of the other resource systems, and these agents act as filters to ensure that only authenticated users can access the resource system. For example, you can install an agent (such an Apache module or IIS filter) on a Web server that is mounted in front of the Sterling Multi-Channel Selling Solution.

The typical flow of events when a user attempts to access one of the resource systems in a single sign-on environment goes like this:

1. The user points their browser to a URL that they should use to access a resource system such as the Sterling Multi-Channel Selling Solution. This URL should not point directly to the Sterling Multi-Channel Selling Solution, but rather to the Web server mounted in front of the Sterling Multi-Channel Selling Solution.
2. The Web server in front of the Sterling Multi-Channel Selling Solution filters the request and the installed agent checks to see if the user has already been authenticated by the Single Sign-on system and has a valid session token from the Single Sign-on system.
    a. If so, then the request is passed through to the Sterling Multi-Channel Selling Solution as an authenticated request.
        - If the request is already part of an authenticated Sterling Multi-Channel Selling Solution session, then the request is processed exactly like any other request.
        - If the request does not have a valid Sterling Multi-Channel Selling Solution session, then the request is passed to the appropriate login controller for processing, and the login controller makes use of special tokens provided in the request header to identify the user.

    b. If the request does not have a valid session token from the Single Sign-on system, then the browser's request is redirected to the sign-on page served up by the Single Sign-on system.
3. The user enters their authentication credentials such as a username and password as provided to them by the administrators of the Single Sign-on system, and submits these to the Single Sign-on system.
4. If the Single Sign-on system successfully authenticates the user, then a special token is added to the request, and the request is redirected back to the URL used to access the Sterling Multi-Channel Selling Solution through the Web server. The token signifies that the user has been authenticated and includes a special identifier (such as a username or user key) that can uniquely identify the user in the Sterling Multi-Channel Selling Solution.
5. The Web server in front of the Sterling Multi-Channel Selling Solution filters the request and now sees that the request has a valid session token from the Single Sign-on system, and so now passes the request to the Sterling Multi-Channel Selling Solution.
6. The Sterling Multi-Channel Selling Solution receives the request and detects that the request does not yet have a valid Sterling Multi-Channel Selling

Solution session. The request is passed to the appropriate login controller. The login controller extracts from the request the unique identifier created in Step 4, creates a session for the user, and restores the user from the identifier so that relevant information such as their roles and which partner they belong to can be retrieved. See "Login Controllers" on page 325 for an example of changes that can be made to a login controller in a single sign-on environment.

7. The user then goes about their Sterling Multi-Channel Selling Solution activities just as they would had they been able to log in directly to the Sterling Multi-Channel Selling Solution.
8. At the end of their work in the Sterling Multi-Channel Selling Solution, the user must log out. You must decide whether you want their act of logging out to apply only to the Sterling Multi-Channel Selling Solution or to indicate that they are logging out of the enterprise environment as a whole. See "Logout Controllers" on page 328 for an example of how to customize the Sterling Multi-Channel Selling Solution logout controller if you want the single sign-on session to be ended as well.

Profile Synchronization

For a single sign-on solution to work, it is critical to decide how profile information is to be maintained. In this section we describe some possible approaches.

Full Synchronization

In this approach, the enterprise administrators take on themselves the burden of maintaining a consistent view of users across the Single Sign-on system and the Sterling Multi-Channel Selling Solution. Each time a new user needs access to the Sterling Multi-Channel Selling Solution, a user profile must be added to the Single Sign-on system and a new user profile under the appropriate partner created on the Sterling Multi-Channel Selling Solution. The two systems must be maintained so that a unique identifier (such as username) can be passed from the Single Sign-on system to the Sterling Multi-Channel Selling Solution so that the user’s profile can be retrieved from it.

Full synchronization can be automated in a number of different ways, such as using a third “master system” to push user information to both systems as it is created in the master system, or by ensuring that the Single Sign-on system pushes data to the Sterling Multi-Channel Selling Solution as it is created on the Single Sign-on system. In these scenarios, it is common to think of the Sterling Multi-Channel Selling Solution as a slave to the Single Sign-on system, and so certain tasks such as user creation or password updates should be disabled in the Sterling Multi-Channel Selling Solution.

Partial Synchronization

In this approach, the user profile administration is maintained on the Single Sign-on system, and is pushed to the Sterling Multi-Channel Selling Solution when the user has successfully authenticated themselves against the Single Sign-on system. This approach requires that the Sterling Multi-Channel Selling Solution login controller can create user profiles, and possibly partner profiles, on the fly, but has the advantage that enterprise administrators do not have a double maintenance task whenever users must be added or changed.

Login Controllers

This section describes the changes that typically have to be made to login controllers in a single sign-on enterprise environment. The examples assume that once the Web server with the agent has checked the request for the special token that the request is passed through with the token in the request header, and the unique identifier for the user is the username of the user in the Sterling Multi-Channel Selling Solution.

Suppose that the DCUserLoginController is used to authenticate direct commerce users as they log in to the Sterling Multi-Channel Selling Solution. Create a new controller class, say SingleSignonLoginController which extends the DCUserLoginController. In the SingleSignonLoginController class, overwrite methods as follows:

```
/**
 * If working in a single sign-on environment, then retrieve the login
 * information from the request header and set the password to a dummy
 * value.

 * The boolean singleSignonBoolean indicates whether or not the
 * Sterling Multi-Channel Selling Solution is operating in a single
sign-on environment. This
 * can be set as a System property or in configuration file.
 *
 * The unique user identifier is retrieved from the request as the
 * value of the SS_LOGIN request header.
 *
 * The SSCredientials class implements the Credentials interface.
 */
protected Credentials getCredentials()
{
    if (!singleSignonBoolean)
    {
        login = request.getParameter(LOGIN);
        passwd = request.getParameter(PASSWD);
        userType = request.getParameter(USERTYPE);
    }
    else
    {
        login = request.getHeader(SS_LOGIN);
        userType = request.getHeader(USERTYPE);
    }
    if (login == null || login.equals(""))
    {
        login = DUMMY_VALUE;
    }
    if (passwd == null)
    {
        passwd = DUMMY_VALUE;
    }
    SSCredentials credentials =
        new SSCredentials(login, passwd, userType);
    return new CommerceCategoryCredentials(credentials,
        PartnerLoginController.CommerceCategory_Direct);
}

/**
 * Performs default post login processing and synchronizes the Single
 * Sign-on roles
 *
 * @param oldSession the ComergentSession object of the user
 * @exception ICCException; IOException
 */
protected String postLoginProcessing(ComergentSession oldSession)
    throws ICCException, IOException
{
    //Perform any processing logic in the parent controller
    super.postLoginProcessing(oldSession);
    if (singleSignonBoolean)
    {
        /*
        Perform logic to match up roles from Single Sign-on system
        with Sterling Multi-Channel Selling Solution System roles.
        */
    }
}

protected boolean doLogin()
{
    return true;
}
```

Credentials Class

The login controller makes use of a Credentials class to retrieve user information from the Sterling Multi-Channel Selling Solution Knowledgebase. Here is an example Credentials class to illustrate the main function this class performs:

```
public class SSCredentials extends UserPasswordCredentials
{
    String m_userType = null;
    /**
    * Constructor
    * @param userName the username of the user
    * @param password the associate password of the user
    * @param userType the type of user for the given username
    */
    public SSCredentials(String userName, String password,
        String userType)
    {
        super(userName, password);
        this.m_userType = userType;
    }

    /**
    * Create the user object for the username, password, and usertype
    * @return the user object
    */
    public User getUser()
    {
        if (m_user != null)
        return m_user;
        try
        {
            SSUserBean userBean = (SSUserBean)
            OMWrapper.getObject("com.comergent.bean.simple.UserBean");
            DsQuery query = QueryHelper.newWhereClause("UserLogin",
                DsQueryOperators.EQUALS, m_userName);
            query = QueryHelper.addWhereClause(query,
                DsQueryOperators.AND, "UserType",
                DsQueryOperators.EQUALS, m_userType);
            if ( !m_userType.equals("internal"))
            {
                query = QueryHelper.addWhereClause(query,
                    DsQueryOperators.AND, "ContactStatus",
                    DsQueryOperators.EQUALS, "A");
            }
            DataContext dc = new DataContext();
            dc.disableAccessCheck();
            userBean.restore(dc, query);
            m_user = new User(userBean);
        }
        catch(ICCException e)
        {
            m_user = null;
        }
        return m_user;
    }

    /**
     * Validate the user in with the information in the member
     * variables
     * @param session the ComergentSession the user is associated
     * @param messageType the message type
     * @return the validation status USER_OK means user is validated;
     * INVALID_USER otherwise
     */
     public int verify(ComergentSession session, String messageType)
    {
        try
        {
            m_user = this.getUser();
            if ( m_user == null )
            {
                return INVALID_USER;
            }
                int retVal;
            if ((retVal = m_user.checkPassword(session)) != USER_OK)
            {
                return retVal;
            }
            return USER_OK;
        }
        catch (InvalidBizobjException ex)
        {
            ex.printStackTrace();
            return INVALID_USER;
        }
    catch (Exception e)
        {
            e.printStackTrace();
            return INVALID_USER;
        }
    }
}
```

Logout Controllers

This section describes the changes that typically have to be made to logout controllers in a single sign-on enterprise environment. In this example, the aim is to ensure when a user clicks the Logout button on a Sterling Multi-Channel Selling Solution page that they are logged out of both the Sterling Multi-Channel Selling Solution and the Single Sign-on system, and so are logged out effectively all other systems that are being managed by the Single Sign-on system.

```
public class SSLogoutController extends
    com.comergent.dcm.caf.controller.Controller
{
    public void execute() throws ControllerException, ICCException,
        IOException
    {
        session.logout();
        session.setLocale(ComergentI18N.getDefaultLocale());
        String redirect = null;
        if (singleSignonBoolean)
        {
            redirect = Global.getString("SSLogoutURL");
        }
        else
        {
            redirect = request.constructAppURL(
                ComergentAppEnv.getDefaultApp(),
                ComergentAppEnv.getDefaultMessageType());
        }
        response.sendRedirect(response.encodeRedirectURL(redirect));
    }
}
```

This example controller first closes the Sterling Multi-Channel Selling Solution session, and then if the Sterling Multi-Channel Selling Solution is operating in single sign-on environment, it redirects the request to a URL that will close the Single Sign-on system session for this user.

RosettaNet Setup Steps

If you plan to implement support for RosettaNet messaging: either to initiate or respond to RosettaNet messages, then follow these steps:

1. Copy the RosettaNet DTDs from ***debs_home*/Sterling/WEB-INF/rosettanet/** to the working directory of your servlet container. This location varies from one servlet container to another: default locations are listed here:

TABLE 19. **Working Directories**

Servlet Container	Home Location
IPlanet	**/iPlanet/iAS6/**
WebSphere	**/usr/WebSphere/AppServer/bin**
Tomcat	**/usr/local/tomcat/bin**
WebLogic	**/apps/bea61sp2/wlserver6.1/**

2. Identify your enterprise's RosettaNet DUNS (Data Universal Numbering System) identifier: this is used in messages.
3. Log in to the Sterling Multi-Channel Selling Solution as an enterprise administrator and navigate to the Enterprise partner profile.
4. On the detail tab, set the Dun & Bradstreet ID field to your DUNS identifier.
5. Click **Save All**.
6. Click **Logout**.
7. Edit the **RosettaNet.xml** file in ***debs_home*/Sterling/WEB-INF/properties/** to set the value of the GlobalBusinessIdentifier element to your DUNS identifier. This value is used to identify your enterprise when RosettaNet messages are sent to a partner system.
8. Edit the **RosettaNet.xml** file to add the mapping between your partners' DUNS identifiers and the URLs to which RosettaNet messages must be sent to them. This is usually the same as the message URL that you set up for them in the partner profile. Each child element of the returnUrl element looks like this:

```
<bus0231-a34f ...>
    http://<URL to post RosettaNet messages>
</bus0231-a34f>
```

   The DUNS identifier is used as the name of the element. If your partner is also running the Sterling Multi-Channel Selling Solution, then the URL is likely to have this form: http://<*server:port*>/Sterling/msg/matrix.

9. Let your partners know your DUNS identifier and the URL that they should use when sending RosettaNet messages to you.

   If they are using the Sterling Multi-Channel Selling Solution to send RosettaNet messages to you, then they must add this information to the **RosettaNet.xml** on their system: from their point of view, you are one of their partners and so they must add your DUNS identifier and URL to their returnUrl element.

10. If you are using a load balancer to support a clustered implementation of the Sterling Multi-Channel Selling Solution, then set the ServerLoadBalanceUrl element in **Comergent.xml** to point to the URL used to access the load balancer.

Sending XML Messages to an External System

In this release, XML messages can be posted from the Sterling Multi-Channel Selling Solution to an external system using the platform Messaging Service. Earlier releases used a different mechanism as described in "Legacy Mechanism" on page 332.

In general, you define the message as a data bean: any standard data object bean can be used. This request bean is serialized into an XML document, and the messaging service layer is invoked to HTTP post the message to any external URL. The response from the external system is processed by the messaging layer and returned to the application as a data bean: referred to as the response bean. Note that the request bean and response bean do not to have to be of the same type.

You must make sure that there are message converters defined: one to transform a native Sterling representation of the request bean into the outbound message and one to convert the inbound response message into the native Sterling representation of the response bean.

These are the steps to follow:

1. In your application code, assemble the data that you want to post to the external system into a data bean. In the example below, we will refer to this object as requestBean. For example:

   ```
   IOrderFactory fac = OrdersAPI.getFactory();
   IOrder io = fac.createNewOrderObjFromOIL(currentKey);
   OrderBean requestBean = io.getDataBean();
   ```

2. Assemble a com.comergent.api.msgService.MsgContext object to be used to send the message. The object has methods to set the message category and

type, set the external message URL, set authentication information (if required), and set the content type. You can use the Partner Manager API to create a MsgContext object that sets all these properties as they are specified in the partner profile. For example:

```
MsgContext context = null;
IPartnerMgrAPI partnerMgrAPI =
    (IPartnerMgrAPI) OMWrapper.getObject(IPartnerMgrAPI.class);
context = partnerMgrAPI.getPartnerHelper(msgPartnerKey,
    false).getMsgContext();
String msgType = "OrderToERPRequest";
context.setMessageType(msgType);
```

3. Invoke the Messaging Service to post the message:

```
try
{
    MsgService msgService =
        MsgServiceFactory.getMsgService(msgContext);
    IData replyBean = (IData) msgService.service(requestBean,
        msgContext);
}
catch (MsgServiceException mse)
{
    // handle the error condition
}
```

In this example, you must have defined a converter from an Order native Sterling message into an OrderToERPRequest message type, and a converter to turn the response message into the native Sterling representation of a data bean.

Legacy Mechanism

The following procedure describes the steps involved in sending out XML messages from earlier (pre-Release 7.0.1) releases of the Sterling Multi-Channel Selling Solution. This example has been implemented for Order Management (for example, ERPOrderCreate message to send to an ERP system) and this will be used to illustrate the steps. See "Description of HTTP Process" on page 336 for details on what happens at the HTTP level.

An example where you may need this ability is when you create new users in the Sterling Multi-Channel Selling Solution. These may need to be synchronized with an external system, and so you may need to send out an XML message to an external system.

Assume that there is a simple data object that can hold the information to be sent out as an XML message. In this example, this is the Order data object defined in **Order.xml**. In the standard implementation of the Sterling Multi-Channel Selling

Solution, this data object can be created, populated with data, and then persisted to the Knowledgebase by invoking a *persist()* call. The steps below describe how to send out an XML message with the same content.

Step 1: Define a Remote Data Object

The first step is to define a data object to be used for the remote operation. In this example, we use RemoteOrder.

```
<?xml version="1.0"?>
<DataObject Name="RemoteOrder" Extends="Order"
    MsgType="ERPOrderCreateRequest" ObjectType="MSG" Version="6.0">
    <DataFieldList>
    <DataField Mandatory="n" Name="ERPSpecificCreateField"
        Writable="y"/>
    </DataFieldList>
</DataObject>
```

Note that the RemoteOrder data object extends the Order data object. Also note that the ObjectType attribute value is MSG. The MsgType attribute should to be set to whatever message type the outgoing XML message should have.

Add any additional data fields needed (over and above whatever is in the base data object: in this case Order).

Create the required entries in **DsBusinessObjects.xml**, **DsRecipes.xml**, and **DsDataElements.xml** files. The **DsRecipes.xml** entry should look something like this:

```
<Recipe BusinessObject="RemoteOrder"
    Description="RemoteOrder Create Recipe" Name="RemoteOrder"
    Version="4.0">
    <DataObjectList>
        <DataObject Access="RWID" DataSourceName="MESSAGE"
            Name="RemoteOrder" Ordinality="1" Version="6.0"/>
    </DataObjectList>
</Recipe>
```

Note that the DataSourceName attribute is set to "MESSAGE". This value is used by the corresponding DataSource element in the ***DB*DataSources.xml** file to determine which data service class to use. For example:

```
<DataSource Name="MESSAGE" Type="DEFAULT" Version="2.0">
    <Primary DataService="MsgService"/>
</DataSource>
```

Step 2: Define a DTD

Define a DTD for the outgoing message. This must have the same name as the MsgType attribute defined in "Step 1: Define a Remote Data Object" on page 333.

For example, for RemoteOrder, the DTD will be **ERPOrderCreateRequest.dtd**. This file should be saved in the ***debs_home*****/Sterling/WEB-INF/messages/** directory. For example, the **ERPOrderCreateRequest.dtd** is defined as follows:

```
<?xml encoding="UTF-8"?>
<!-- ERPOrderCreateRequest
    Document Type Declaration (DTD)
    Version 1.0
    25-Jul-00
    Authors:
        Comergent
        Contact: (650) 610-6800
        support@comergent.com
-->
<!ENTITY % MessageHeader SYSTEM "MessageHeader.dtd">
%MessageHeader;
<!ENTITY % RemoteUser SYSTEM "RemoteUser.dtd">
%RemoteUser;
<!ENTITY % RemoteOrder SYSTEM "../bizobjs/RemoteOrder.dtd">
%RemoteOrder;
<!ELEMENT Comergent (
    MessageHeader,
    RemoteUser,
    RemoteOrder)
>
```

Note that the DTD includes the generated DTD **RemoteOrder.dtd**. This DTD is generated automatically into the **WEB-INF/bizobjs/** directory when the generateDTD SDK target is run. Consequently, the DTD will remain consistent with the RemoteOrder data object even if changes are made to the underlying Order data object.

Step 3: XML Transmission Properties

The XML messages must be sent to a remote system. The Sterling Multi-Channel Selling Solution needs several parameters to format and send this XML message. These include:

- username
- password
- URL
- message type
- message version
- security parameters for the remote system

The Sterling Multi-Channel Selling Solution uses a Partner data object to hold all this information. For example, for sending Orders to ERP system, there is a ERPPartner object defined with the required information.

This partner key is set using a property defined in the **Comergent.xml** file from where the application can retrieve it. For Order Management applications, this property is ERPIntegrationURL.

Step 4: Code Changes

These steps are implemented in the Java source code of the application:

1. Instantiate a RemoteOrderBean object:

   ```
   RemoteOrderBean remoteOrderBean = (RemoteOrderBean)
   OMWrapper.getObject("com.comergent.bean.simple.RemoteOrderBean");
   ```

2. Call the data services method *copyBean()* to copy the content to this bean. For example:

   ```
   orderBean.copyBean(remoteOrderBean);
   ```

3. Set values for any ERP specific fields in the RemoteOrderBean. For example, the code extract here sets the value for ERPSpecificCreateField we defined in "Step 1: Define a Remote Data Object" on page 333.

   ```
   remoteOrderBean.setERPSpecificCreateField("Order Create Request
   from DEBS to ERP");
   ```

4. Set the partner key (the one defined in "Step 3: XML Transmission Properties" on page 334) on the RemoteOrderBean.

   ```
   String ERPPartnerID =
       Global.getString("C3_Integrator.ERPIntegrationUrl");
   remoteOrderBean.setPartnerId(ERPPartnerId);
   ```

   The *setPartnerId()* method is inherited from the base DataBean class.

5. Set the values for the remote session.

   ```
   RemoteSession remoteSession = new RemoteSession();
   remoteSession.setDistributorKey(new Long(ERPPartnerId));
   remoteSession.setId(null);
   remoteSession.setUser(username);
   remoteSession.setAuthenticator(password);
   ComergentAppEnv.addRemoteSession(remoteSession);
   ```

   The username and password are those needed by the receiving system to authenticate the message.

6. Call *persist()* on the RemoteOrder data object.

```
remoteOrderBean.persist();
```

Note that this *persist()* call (which sends out the XML message) may fail or throw an Exception. The application must be aware of this eventuality and surround the *persist()* call in an appropriate throw-catch block. For example, in Order Management, when an Exception is thrown, the Order is marked specially and the message is retried later in a cron job.

Step 5: Using Converters

The *persist()* call in Step 6 of "Step 4: Code Changes" on page 335 will send out a Comergent XML message by default. To send out a message of a different type (say dXML or Rosettanet), you need to install appropriate converters. See CHAPTER 18, "Message Conversion" for more information.

Description of HTTP Process

All the Sterling Multi-Channel Selling Solution messages are dispatched synchronously. The application thread waits for the message to be sent out on a new HTTP connection and for the response to come back. If possible, the HTTP response is used to retrieve any required information and to populate any data beans. If some failure condition arises, then an appropriate exception is thrown and handled accordingly.

Invoking *persist()* (as in Step 6 of our example in "Step 4: Code Changes" on page 335) on a data bean causes the Sterling Multi-Channel Selling Solution to open a new HTTPConnection and to perform a post with the content formatted as an XML string.

The resulting response from this connection is converted to XML and then to a data bean if possible. In our example, this response message is ERPOrderCreateReply. In our reference implementation, this message just contains the same information as the request (which is the RemoteOrder bean). See **WEB-INF/messages/ ERPOrderCreateReply.dtd** for more information.

This data object is copied into the bean (`remoteOrderBean` in the example). Thus the application at the end of the *persist()* method has the handle to the bean with the response content in it. The application may use this response content in its logic.

In the reference implementation, because the response is exactly the same content as in the request, we do not use it. In this specific example of sending an ERPOrderCreateRequest message, if the post is successful, then we move the Order to the Order_Submitted state. Otherwise the integration status is to "N". All orders with integration status set to "N" are handled by the OrderERP cron job. (See the *Sterling Multi-Channel Selling Solution Developer Guide* for the state transitions supported by the Order data object).

A customer implementation may change the response content and use it in its logic (possibly to move the Order state to a different state).

Asynchronous Messages in the OrderMgmt Application

The OrderMgmt application can receive messages from an ERP system asynchronously. In the reference implementation, one such message is the OrderStatusUpdateRequest from the ERP system. The ERP system sends this message after it has received a ERPOrderCreateRequest from the Sterling Multi-Channel Selling Solution and wants to indicate whether it accepts or rejects this ERPOrderCreateRequest message.

The OrderStatusUpdateRequest DTD has a field "ERPResponse" where the ERP system can set "Accepted" or "Rejected" to indicate whether the ERP system will continue processing this ERPOrderCreateRequest message or not. If the Sterling Multi-Channel Selling Solution receives the "Accepted" status, then it moves the order from the Order_Submitted to InProcess status. Otherwise, it moves it to the Rejected status. The Order State Machine described in the *Sterling Multi-Channel Selling Solution Developer Guide* details these transitions in more detail.

Note that the reference implementation of OrderMgmt follows a strict State Machine and thus it accepts and processes incoming messages only in certain valid states. For example, it does not accept ERPOrderCreateReply from the ERP system unless it is a synchronous response to the ERPOrderCreateRequest that the Sterling Multi-Channel Selling Solution has sent to the ERP system.

Trading Partner Enrollment Form

One of the challenges of managing integration with external partners is to effectively coordinate your deployments. To this end, a good practice is to distribute a trading partner enrollment form to your partners so that they can commit to a particular form of integration and schedule. A sample trading partner enrollment form is given here:

TRADING PARTNER ENROLLMENT FORM			
PARTNER NAME			
EDI TECHNICAL CONTACT:			
TELEPHONE:		FAX:	
EMAIL ADDRESS:			
REQUESTED TRANSACTIONS:			

ANSI X12 VERSION	
PLEASE IDENTIFY WHEN YOUR SYSTEMS WILL BE PREPARED TO SEND AND RECEIVE TEST DATA:	MONTH DAY YEAR
PLEASE ADVISE WHEN YOUR TECHNICAL AND BUSINESS TEAM WILL BE AVAILABLE TO DISCUSS IMPLEMENTATION REQUIREMENTS:	MONTH DAY YEAR
ADDITIONAL FUNCTIONALITY? PLEASE DESCRIBE:	

Name	Test Environment	Production Environment
ISA Qualifier		
ISA ID		
GS ID		
Element Terminator		
Sub-Element Terminator		
Segment Terminator		

CHAPTER 22

Emulating a Browser Interface to an External System

This chapter describes how you can create an interface for the Sterling Multi-Channel Selling Solution to retrieve information from a browser-based e-commerce system. We provide a service class, called HTMLService to support this interface.

Overview

In an installation of the Sterling Multi-Channel Selling Solution, you can use a number of different types of data sources. Earlier chapters have described the JDBCService and MsgService classes that enable the data integration layer to retrieve data from and save data to database servers and external systems respectively.

In some circumstances, the system might be required to retrieve data from and save data to a system whose primary interface is designed to be browser-based. For example, a partner server might be required to retrieve price and availability data from an e-commerce system that is designed to provide the same information to customers through a browser.

In these settings, the Sterling Multi-Channel Selling Solution emulates a browser by posting a request to the e-commerce Web server and then the Sterling Multi-Channel Selling Solution parses the response to recover the data provided by the e-commerce system. The response is in the form of an HTML message that provides name-value pairs.

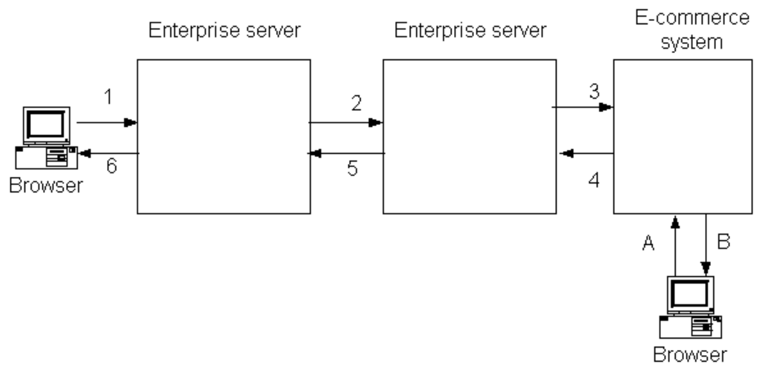


FIGURE 19. **Sterling Multi-Channel Selling Solution Emulation of Browser Interface**

For example, consider Figure 19 on page 340. Suppose that a distributor has an e-commerce system that provides a browser-based interface to customers. A customer points their browser to the e-commerce system and posts their request to the system (A). The e-commerce system responds by returning a Web page to the customer's browser (B).

Now suppose that this distributor is part of a Sterling Multi-Channel Selling Solution network. When a customer initiates a price and availability request from their browser (1), the enterprise server dispatches a price and availability request to the enterprise server at the distributor's site (2). Using the HTMLService class, the data layer of the partner's enterprise server posts a request to the e-commerce system (3). The e-commerce system returns a response (4) providing the data requested. In turn, the partner's enterprise server returns the data to the enterprise server (5), and finally the enterprise server returns a new Web page to the customer (6).

HTMLService Class

Like the MsgService and JDBCService classes, the HTMLService class is designed to provide data layer integration with external data sources. If the DataSource associated to a business object specifies the HTMLService class in its DataService attribute, then the HTMLService class is invoked whenever the *restore()* or *persist()* methods are called on the business object.

The HTMLService class extends the abstract DataService class. It posts a request to the e-commerce system, and uses a series of parsing methods to recover the name-value pairs provided in the HTML response.

Formats of Request and Response

This section describes the structure of the requests and responses used by the HTMLService class.

Post Request

The typical form of the post is:

```
http://<server>:<port>/<location>?param1=value1&param2=value2...
```

Here, server, port, and location are determined at implementation. The location specifies the entry point to the e-commerce system. For example, an active server page-based system might provide a URL of the form:

```
http://<server>:<port>/login.asp?param1=value1&param2=value2...
```

whereas a servlet-based e-commerce system might provide a URL of the form:

```
http://<server>:<port>/LoginServlet?param1=value1&param2=value2...
```

The parameters that are passed to the e-commerce system must provide sufficient information for the request to be processed correctly. These are defined in the specification of the interface as described below (see "Interface Specification" on page 342).

HTML Response

For the HTMLService class to retrieve data from the HTML response, the returned page must conform to the following format:

```
<HTML>
    <BODY>
        Name1=Value1;
        Name2=Value2;
        Name3=Value3;
        Name4=Value4;
    </BODY>
</HTML>
```

Here, the name-value pairs are listed on a separate line terminated by a semi-colon (;). Both Name and Value are strings that must be separated by an equal sign (=).

The returned name-value pairs must be defined in the interface specification and must be sufficient to determine a valid response (see "Interface Specification" on page 342).

Interface Specification

To ensure that the HTMLService class performs correctly, you must specify the location and parameters used in each post and the name-value pairs that must be returned in response to each request.

The specification is provided by a properties file and is read at the time the Sterling Multi-Channel Selling Solution is started. Each time the HTMLService class is invoked the appropriate parameters must be defined by the business object.

A typical table to specify the interface might look something like Table 20, "Sample HTMLService Interface Specification", on page 342:

TABLE 20. Sample HTMLService Interface Specification

Function	Request	Response

Regular Login	USER_ID=userid
	PASSWORD=password
		STATUS_CODE=statusCode
		STATUS_MESSAGE=status-Message
		SESSION_ID=sessionId

Referral Login	USER_ID=userid
	DISTI_SESSION_ID=distiSession-Id
		STATUS_CODE=statusCode
		STATUS_MESSAGE=status-Message
		SESSION_ID=sessionId

Price and Availability	SESSION_ID=sessionId
	MARKET_TYPE=marketType
	CURRENCY_CODE=requested-CurrencyCode
	SKU_AUTHORITY= manufacturerId
	SKU=manufacturerSKU
	QUANTITY=quantity
		STATUS_CODE=statusCode
		SELLER_SKU=distributorSKU
		AVAILABLE_QUANTITY= quantity
		LIST_PRICE=listPrice
		PRICE=discountedPrice
		CURRENCY_CODE=reply-CurrencyCode

TABLE 20. **Sample HTMLService Interface Specification (Continued)**

Function	Request	Response

Shopping Cart Reference	SESSION_ID=sessionId
	BUYER_SHOPPING_CART_ID= manufacturerShoppingCartId
		STATUS_CODE=statusCode
		STATUS_MESSAGE=status-Message
		SELLER_SHOPPING_CART_ID =distributorShoppingCartId

Add Shopping Cart Item	SESSION_ID=sessionId
	SELLER_SHOPPING_CART_ID =distributorShoppingCartId
	MARKET_TYPE=marketType
	CURRENCY_CODE=requested-CurrencyCode
	SKU_AUTHORITY= manufacturerId
	SKU=manufacturerSKU
	QUANTITY=quantity
		STATUS_CODE=statusCode
		SELLER_SKU=distributorSKU
		WAREHOUSE_LOCATION= location
		LIST_PRICE=listPrice
		PRICE=discountedPrice
		CURRENCY_CODE=reply-CurrencyCode

For example, to add a shopping cart item using this interface successfully, the HTMLService class might post a URL that looked like this:

```
http://<server>:<port>/<location>?SESSION_ID=df7686089&
    SELLER_SHOPPING_CART_ID=35&MARKET_TYPE=Education&
    CURRENCY_CODE=USD&SKU_AUTHORITY=Matrix&SKU=MXWS-7600&QUANTITY=5
```

In response, the server might return a page similar to this one:

```
<HTML>
    <BODY>
        STATUS_CODE=0000;
        SELLER_SKU=DXWS-7600;
        SKU AVAILABLE_QUANTITY=67;
        LIST_PRICE=1999.95;
        PRICE=1675.00;
        CURRENCY_CODE=USD;
    </BODY>
</HTML>
```

CHAPTER 23

Customizing Sterling Pricing

The Sterling Pricing application manages both user access to products and the assignment of prices to products. It uses price lists to set prices for products and then by associating price lists with partners determines the products that may be seen by partner users. See the *Sterling Multi-Channel Selling Solution Administration Guide* for a detailed conceptual introduction to price lists.

This chapter covers the possible changes you may make to Sterling Pricing:

- Changing the list of currencies or customer types (formerly known as vertical markets)
- Implementing a Pricing Engine
- Using Sterling Pricing for Entitlement Only
- Implementing New Pricing Rule Variables
- Defining Auxiliary Price Types

Sterling Advisor Pricing

Note that pricing information displayed to users as they browse the product catalog can either be displayed using the pricing engine or by directly querying the Knowledgebase tables. The Sterling Advisor business rule called **Apply Dynamic Pricing to Product List** controls this behavior:

- on: the pricing engine is used to retrieve prices.

- off: pricing data is retrieved from the Knowledgebase tables directly.

Price Lists

Each price list comprises:

- Header information: the name of the price list, its description, the customer type and currency for the price list, status, and effectivity dates for the price list.
- Products: a list of products on the price list.
- Price types: the auxiliary prices that can be associated with products assigned to this price list.
- Rules: rules are associated with product categories and products to modify specific prices based on runtime attributes of the partner user or the quantity of a product.

Header Information

When a price list is created, the pricing administrator specifies the name of the price list and sets effectivity dates for the price list. Each price list name must be unique.

In addition, the administrator specifies the customer type (vertical market) and currency for the price list by selecting values from drop-down lists. The available values for the customer types drop-down list are populated from the CMGT_VERTICAL_MARKETS table. The available currencies for the currencies drop-down list are populated from the CMGT_CURRENCIES table.

In both cases, to manage the available options, make sure that when you first populate these tables, you create all the values you want your implementation to support. See CHAPTER 6, "Creating and Populating the Knowledgebase" for more information about populating the tables. See the *Sterling Multi-Channel Selling Solution Reference Guide* for more information about the Knowledgebase tables.

Rules

Each pricing rule comprises two parts:

- rule variables: these are the variables that determine whether or not a particular rule should be applied.

- adjustment factors: these are the discounts or surcharges that are to be applied to the price if the rule is applicable. These are expressed either as absolute amounts or as percentages of the price.

Rule Variables

A rule variable is any field that can be evaluated at runtime and whose value can be compared to a pre-defined set of values: we refer to these as the rule variable options. Each pricing rule variable that is defined as part of your implementation appears as a selectable item in the First Option and Second Option drop-down lists on the Pricing Rule Administration pages. See the *Sterling Multi-Channel Selling Solution Administration Guide* for details of the Pricing Rule Administration pages.

As part of your implementation of the Sterling Multi-Channel Selling Solution, you must decide what pricing rule variables to create and what options each rule variable can offer. See "Implementing New Pricing Rule Variables" on page 350 for more information.

In general, you should limit your choice of pricing rule variables to fields that will be used to adjust prices displayed to customers. Each pricing rule variable is defined in the PricingRuleVariableDefinition Java class. You should avoid having to modify this frequently because new variables will only be used when the servlet container is restarted.

Definition of a Pricing Rule Variable

Each pricing rule variable has the following member variables that must be properly defined to assure proper usage of the pricing rules. The first five entries in the following table comprise the header for the pricing rule variable.

TABLE 21. Defining Elements of a Pricing Rule Variable

Defining Element	**Description**
ID	ID of the pricing rule variable.
Definition	Type of data source. See "Sources of Pricing Rule Variables" on page 351.
Description	Description of the rule variable (option visible in the drop-down list).
Type	Input type (possible choices) for a variable, including string values, numbers, or number ranges.

TABLE 21. **Defining Elements of a Pricing Rule Variable (Continued)**

Defining Element	Description
OptionNo	Total number of possible choices for each variable.
Option	Choices for each variable. Comprises a definition including an ID starting from "1", a name that must match the entry of a data source, and a description that is displayed on the administration pages.

About the Sterling Pricing Service

The Sterling Customer Order Management (COM) system and Sterling Distributed Order Management (DOM) system applications use the Sterling Pricing Service to send pricing requests to the Sterling Multi-Channel Selling Solution. For information about the pricing service, see the Pricing Service chapter of the *Sterling Multi-Channel Selling Solution Reference Guide*. For information about the Pricing Service web services, see the Web Services chapter of the *Sterling Multi-Channel Selling Solution Developer Guide*.

Implementing a Pricing Engine

The Sterling Multi-Channel Selling Solution provides a pricing engine to retrieve pricing and entitlement data from the Knowledgebase. It is called from the main pricing class PriceCheckAPI and instantiated using the PricingSourceFactory class. You can implement your own pricing engine class or customize only the methods used to compare prices.

To implement your own pricing engine class and use it for either or both of pricing and entitlement checking, follow these steps:

1. Write the Java class for the pricing engine. Any pricing engine class must implement the com.comergent.api.apps.pricingMgr.PricingSource interface. This interface provides a number of similar methods used to check prices. Note that the class should assume that the PriceCheckAPI class has retrieved from the user's session whatever information is required (such as partner key, customer type, and currency) prior to invoking a *checkPrice()* method.
2. Modify the **ObjectMap.xml** file by changing the element whose ID attribute declares the pricing engine:

```
<Object ID="com.comergent.apps.pricingMgr.pmComergent.core.-
    PricingEngine">
    <ClassName>com.comergent.apps.pricingMgr.pmComergent.core.-
    NewPricingEngine</ClassName>
</Object>
```

3. If you plan to use the pricing engine only for entitlement checking, then see "Using Sterling Pricing for Entitlement Only" on page 349. If the pricing engine is to be used for entitlement checking, then it must implement the com.comergent.api.apps.pricingMgr.EntitlementSource interface and its *checkEntitlement()* method.

Note that you can use a combination of pricing engines, one for pricing and one for entitlements, or you can use the same pricing engine for both.

There are two separate interfaces for customizing the price comparison methods:

- The com.comergent.api.apps.pricingMgr.IPriceCompareOneTimePrice interface contains a method, *comparePrice()*, for comparing one-time prices.
- The com.comergent.api.apps.pricingMgr.IPriceCompareAuxPrices interface contains a method, *compareAuxPrices()*, for comparing a set of auxiliary prices.

You can change the methods used by creating new implementations of these interfaces, and then mapping the new classes to the existing interfaces in the **ObjectMap.xml** file as described above.

Using Sterling Pricing for Entitlement Only

It is possible to implement Sterling Multi-Channel Selling Solution so that the price lists are used to manage the access users have to products (known as entitlement checking), but not to serve pricing information. If you do this, then you must create an external pricing engine that does provide pricing information. See "Implementing a Pricing Engine" on page 348. This section covers the steps you perform to implement this feature.

1. Using the Business Rule Administration page, set the Pricing Engine Type to “Entitlement Only”. This has the effect of setting the value of the pricingEngineMode element in the **StorefrontBusinessRule.xml** file to “2”.
2. For the external pricing engine, create a Java class that implements the PricingSource interface.
3. Modify the **ObjectMap.xml** file by changing the elements whose ID attribute declares the pricing engine to be used for entitlement and the external pricing engine to be used for pricing:

```
<Object ID="com.comergent.apps.pricingMgr.pmComergent.core.-
    PricingEngine">
    <ClassName>com.comergent.apps.pricingMgr.pmComergent.core.-
    NewPricingEngine</ClassName>
</Object>
<Object ID="com.comergent.apps.pricingMgr.pmComergent.core.-
    ExternalPricingEngine">
    <ClassName>com.comergent.apps.pricingMgr.pmComergent.core.-
    NewPricingEngine</ClassName>
</Object>
```

By default, you can leave the internal pricing engine to point to the standard PricingEngine class: its *checkEntitlement()* method can still be called. However, if you choose to point to a different pricing engine, then make sure that it implements the *checkEntitlement()* method.

Note that these steps will only provide the functionality that you need if your external pricing source uses the same (or less) information to determine pricing for products. If your external pricing source uses different or more complex information in determining prices, then you will have to extend the PricingSource interface to support methods that enable this information to be used to retrieve prices. You will also have to modify the PricingSourceFactory class to return the new interface and the application classes that access pricing to invoke these new methods.

Implementing New Pricing Rule Variables

This section describes how to go about creating a pricing rule variable. You must have access to the Sterling Multi-Channel Selling Solution source tree to do this.

Modifying PricingRuleVariableDefinition

Using your preferred text editor or Java development application, open the Java source file **PricingRuleVariableDefinition.java** to be found in the com.comergent.apps.pricingMgr.pmComergent.core package.

Modify the *getRules()* method to define the set of rule variables that you want to use. Each rule variable is created with the same basic code:

```
// Definition of Rule Variable
rule = new PricingRule();
rule.setDateFormat(format);
rule.setID(counter++);
// ID is the key stored in CMGT_PRICELIST_LINES-RULE_VARIABLE
rule.setDefinition("BusinessObject.Field"); // Source location
rule.setDescription(rb.getString("Description")); // Description
rule.setType(PricingEngineConstants.Type);
/**
    Code to set rule options
*/
rules.add(rule);
}
```

In general, you can leave the date format unchanged. The ID of the rule, set through the *setID()* method, simply assigns a number to the rule. Rule ID numbers must be unique and consecutive.

The *setDefinition()* method is used to specify what attribute of the business objects should be used to compare values against the value set in the rule. For example, if you set the definition to be “Partner.PartnerType”, then the value of the user’s partner type is compared to the value set in the rule.

setDescription() is used to determine what is displayed to administrators in the price list rule editor page. The *setType()* method simply sets the type of the data to be compared.

The code that adds the possible values of the rule options will depend on the source of the pricing rule variable. In general, the values will usually come from one of these sources:

- Values retrieved from CMGT_LOOKUPS
- Values retrieved from another Knowledgebase table such as CMGT_TERRITORIES
- Values set in the code that creates the rule variable

Add each option to the rule with the method *setOption()* which takes three arguments:

- its index as an int
- its name as a String
- a description as a String

Sources of Pricing Rule Variables

The following types of data sources are supported as rule variables:

- Fields in a partner profile (for example, partner level, partner territory, and so on). The data object, **Partner.xml**, contains the variable definitions.

- Quantity of a line item assigned to a product inquiry list business object, a pricing and availability business object, a price check business object, or an internal pricing application program interface (API).

Defining Auxiliary Price Types

This section describes how to define auxiliary price types and load them into your Sterling Multi-Channel Selling Solution knowledgebase. You must have access to the Sterling Multi-Channel Selling Solution database and to the SDK to define and then load auxiliary price types into the knowledgebase. You can define auxiliary price types either before you create the knowledgebase (preferred) or you can add price types to an existing knowledgebase. If you add price types after implementing your Sterling Multi-Channel Selling Solution, then restart the servlet container to make the price types available.

Price Type Objects

Sterling Multi-Channel Selling Solution objects are defined as XML elements. These elements are contained in a set of **.lst** files (list files) that are used by the SDK's XMLLoader data loading script to populate the Sterling Multi-Channel Selling Solution database. The list files reside in the directory ***SDK_home*\workspaces*debs_home*\WEB-INF\xmldata**. As part of defining auxiliary price types, you create a file, **PriceTypesList** (no file extension) to contain the auxiliary price types' XML elements, and add it to the list of files contained in the **WEB-INF\xmldata\LightWeightLookupList.lst** file. The XMLLoader script will load your auxiliary price types as part of loading the rest of the knowledgebase data.

Price Type Group Codes

Each price type is associated with a single price group code. Price group codes are useful for determining the types of prices to display for a given product and for ease of maintenance. For example, suppose that your Sterling Multi-Channel Selling Solution implementation has a Potential Costs price group code for costs such as Overage and Cancellation fees. The Configurator can display all auxiliary prices that belong to the Potential Costs group for a particular product, then exclude those prices from display in the submitted order. You could add a new price type to the Potential Costs group called lateFee without changing any display code.

The price group codes are defined in a constants file. Since price group codes are not displayed to end-users, there is no associated lookup for these codes.

When determining the price groups for your implementation, consider categories that have meaning in the real world. For example, you could define a price group, Recurring Costs, for costs that must be paid on a regular basis such as monthly fees, and Potential Costs for costs that are incurred depending on circumstances or triggers, such as cancellation fees.

Miscellaneous is not a good name for a price group. If the price types that would go into a Miscellaneous group have nothing in common, then they should be in separate groups even if each group contains only one price type.

Do not create a separate price type group for one-time prices: all required costs that are paid once only for a service contractable product are stored as product SKU's one-time price and are always displayed when a price is displayed.

Required Information for Price Type Definition

Before defining price types, determine the price types and related information required for your Sterling Multi-Channel Selling Solution implementation. Related information includes:

- Price type code: a unique numeric value that maps to a lookup of type PriceType for use in the localized labeling of prices at display time. Out of the box price type codes are 0 (One Time Price), 1000 (Monthly), 2000 (Cancellation), and 3000 (Overage). To check the values in use in your implementation, select the values in the CMGT_PRICE_TYPE table.
- Description: a string describing the price type, such as "One Time Price" or "Overage".
- Price group code: a category used to group price types to allow the display of similar price types to the end user, or to exclude certain price types from display.
- Property name: this name conforms to the property naming conventions used in the Configurator and Visual Modeler for use in mapping the price type to the component of a model. Price type property names begin with "PRICE:"

Adding Price Types

The following procedures cover the en_US locale. To create lookups for other locales, follow the process you normally use for other localizations.

To Define Price Types Before Creating the Knowledgebase

To define price types before creating the knowledgebase using the SDK:

1. Add the new price types to the lookup list. There must be one lookup for each of the price types you want to add.

   a. In the SDK, retrieve the **LightWeightLookupList** file for customizing:

   ```
   sdk customize WEB-INF/xmldata/Minimal/LightWeightLookupList
   ```

   b. Open the LightWeightLookupList file with a text editor and add your new price types to the file. For example, the following code shows the out-of-the-box lookup entries for Monthly, Cancellation, and Overage price types, as well as an entry for a new price type, Activation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<LightWeightLookupListData>
    ....
    ....
    <LightWeightLookupList state="INSERTED" type="BusinessObject">
        <LightWeightLookup state="INSERTED">
            <LookupType state="INSERTED">PriceType</LookupType>
            <LookupCode state="INSERTED">1000</LookupCode>
            <Locale state="INSERTED">en_US</Locale>
            <Description state="INSERTED">Monthly</Description>
        </LightWeightLookup>
        <LightWeightLookup state="INSERTED">
            <LookupType state="INSERTED">PriceType</LookupType>
            <LookupCode state="INSERTED">2000</LookupCode>
            <Locale state="INSERTED">en_US</Locale>
          <Description state="INSERTED">Cancellation</Description>
        </LightWeightLookup>
        <LightWeightLookup state="INSERTED">
            <LookupType state="INSERTED">PriceType</LookupType>
            <LookupCode state="INSERTED">3000</LookupCode>
            <Locale state="INSERTED">en_US</Locale>
            <Description state="INSERTED">Overage</Description>
        </LightWeightLookup>
       <LightWeightLookup state="INSERTED">
            <LookupType state="INSERTED">PriceType</LookupType>
            <LookupCode state="INSERTED">4000</LookupCode>
            <Locale state="INSERTED">en_US</Locale>
            <Description state="INSERTED">Activation</Description>
        </LightWeightLookup>
    </LightWeightLookupList>
</LightWeightLookupListData>
```

2. Create a new file in the directory **WEB-INF/xmldata** called **PriceTypeList**, to contain your price type definitions. The contents must be plain text: do not use special characters or characters such as smart quotes.

   The format of the PriceTypeList file is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<PriceTypeListData>
<PriceTypeList state="INSERTED" type="BusinessObject">
  <PriceType state="INSERTED">
    <PriceTypeCode state="INSERTED">1000</PriceTypeCode>
    <Locale state="INSERTED">en_US</Locale>
    <PriceTypeGroupCode state="INSERTED">10</PriceTypeGroupCode>
    <PriceTypePropertyName state="INSERTED">PRICE: MONTHLY</Price-
TypePropertyName>
    <UpdateDate state="INSERTED">2006-09-28 00:00:00.0</UpdateDate>
    <UpdatedBy state="INSERTED">1</UpdatedBy>
    <CreateDate state="INSERTED">2002-05-28 00:00:00.0</CreateDate>
    <CreatedBy state="INSERTED">1</CreatedBy>
    <ActiveFlag state="INSERTED">Y</ActiveFlag>
  </PriceType>
  <PriceType state="INSERTED">
    <PriceTypeCode state="INSERTED">2000</PriceTypeCode>
    <Locale state="INSERTED">en_US</Locale>
    <PriceTypeGroupCode state="INSERTED">20</PriceTypeGroupCode>
    <PriceTypePropertyName state="INSERTED">PRICE: CANCELLATION</
PriceTypePropertyName>
<UpdateDate state="INSERTED">2006-09-28 00:00:00.0</UpdateDate>
    <UpdatedBy state="INSERTED">1</UpdatedBy>
    <CreateDate state="INSERTED">2002-05-28 00:00:00.0</CreateDate>
    <CreatedBy state="INSERTED">1</CreatedBy>
    <ActiveFlag state="INSERTED">Y</ActiveFlag>
  </PriceType>
  <PriceType state="INSERTED">
    <PriceTypeCode state="INSERTED">3000</PriceTypeCode>
    <Locale state="INSERTED">en_US</Locale>
    <PriceTypeGroupCode state="INSERTED">20</PriceTypeGroupCode>
    <PriceTypePropertyName state="INSERTED">PRICE: OVERAGE</Price-
TypePropertyName>
    <UpdateDate state="INSERTED">2006-09-28 00:00:00.0</UpdateDate>
    <UpdatedBy state="INSERTED">1</UpdatedBy>
    <CreateDate state="INSERTED">2002-05-28 00:00:00.0</CreateDate>
    <CreatedBy state="INSERTED">1</CreatedBy>
    <ActiveFlag state="INSERTED">Y</ActiveFlag>
  </PriceType>
    <PriceType state="INSERTED">
    <PriceTypeCode state="INSERTED">4000</PriceTypeCode>
    <Locale state="INSERTED">en_US</Locale>
    <PriceTypeGroupCode state="INSERTED">20</PriceTypeGroupCode>
    <PriceTypePropertyName state="INSERTED">PRICE: ACTIVATION</Price-
TypePropertyName>
    <UpdateDate state="INSERTED">2007-03-28 00:00:00.0</UpdateDate>
    <UpdatedBy state="INSERTED">1</UpdatedBy>
    <CreateDate state="INSERTED">2007-03-28 00:00:00.0</CreateDate>
    <CreatedBy state="INSERTED">1</CreatedBy>
    <ActiveFlag state="INSERTED">Y</ActiveFlag>
  </PriceType>
</PriceTypeList>
</PriceTypeListData>
```

3. Add your customized PriceTypeList to the list of minimal data files to be loaded into the knowledgebase:

   a. Retrieve the **MinimalData.lst** file for customization:

   ```
   sdk customize WEB-INF/scripts/MinimalData.list
   ```

   b. Open the **MinimalData.lst** file with a text editor and add the following line:

   ```
   WEB-INF/xmldata/PriceTypeList
   ```

4. Merge the customizations into the build:

   ```
   sdk merge
   ```

5. Create the knowledgbase:

   ```
   sdk createDB
   ```

6. Load the data:

   ```
   sdk loadDB
   ```

To Add Price Types To an Existing Knowledgebase

To add price types to an existing knowledgebase:

1. Insert an entry in the CMGT_LOOKUPS table. The entry maps a localized description of the auxiliary price type to its numeric code. For example, to insert the lookup entry for a new price type called Activation that has a lookup code of 6000 in a SQL Server database:

   ```
   insert into CMGT_LOOKUPS (LOOKUP_TYPE,LOOKUP_CODE,
   DESCRIPTION,LOCALE,ACTIVE_FLAG)
   values ('PriceType',6000,'Activation','en_US','Y')
   ```

2. Insert an entry in the CMGT_PRICE_TYPE table to map the price type codes to the property name and to price type group codes. For example, to insert the price type code entry for the Activation price type with the property name "PRICE: ACTIVATION" in a SQL Server database:

   ```
   insert into CMGT_PRICE_TYPE (PRICE_TYPE_CODE,GROUP_CODE,
   PROPERTY_NAME,UPDATED_BY,CREATED_BY,ACTIVE_FLAG)
   values(6000,20,'PRICE:ACTIVATION',1,1,'Y')
   ```

Examples

In this section we provide some examples of how to modify the PricingRuleVariableDefinition class to add new pricing rule variables.

Data Source from Partner Business Object and a String Input Type

The following code fragment demonstrates how to define a pricing rule variable that contains a partner business object data source and a string input type:

```
// Definition of Rule Variable
rule = new PricingRule();
rule.setDateFormat(format);
rule.setID(counter++);
rule.setDefinition ("Partner.MemberLevel");
rule.setDescription(rb.getString("Partner Membership Level"));
rule.setType(PricingEngineConstants.VARIABLE_TYPE_DISCRETE_STRING);
lookupType = I18NOkayConstants.STRING_PARTNER_LEVEL;
v = getLookup(lookupType);
if (v.size()>0)
{
    String[] s = getLookupDescription(v);
    rule.setOptionNo(v.size()); // total number of possible options
    for (int i=0; i<v.size(); i++)
    {
        rule.setOption(i+1, s[i], s[i]);
    }
    rules.add(rule);
}
```

It does the following:

1. Declares a new pricing rule variable.
2. Sets date format to MM/dd/yyyy:HH:mm:ss.
3. Sets ID to an integer number.
4. Sets definition as “Partner.MemberLevel”. This is the field that is compared to the values generated below.
5. Sets description as “Partner Membership Level”. This is the text that is visible to a price list administrator.
6. Sets the type as “PricingEngineConstants.VARIABLE_TYPE_DISCRETE_STRING”, a pre-defined string constant.

7. Loop through the possible lookup tables for all options. The *getLookup()* method retrieves a vector of values from the CMGT_LOOKUPS table whose LOOKUP_TYPE is “member_level”. For each option, add the option to the rule.
8. Add the rule variable to the vector of rules.

Data Source from Business Objects and a Number Range Input Type

The following code fragment demonstrates how to define a pricing rule variable that contains a business object data source and a number range input type:

```
// Definition of Rule Variable
rule = new PricingRule();
rule.setDateFormat(format);
rule.setID(counter++);
rule.setDefinition("LineItem.Quantity");
rule.setDescription(rb.getString("Ranged Quantity"));
rule.setType(PricingEngineConstants.VARIABLE_TYPE_NUMBER_RANGE);
rule.setOptionNo(4);
rule.setOption(1, "1-5", rb.getString("1-5 items"));
rule.setOption(2, "6-9", rb.getString("6-9 items"));
rule.setOption(3, "10-29", rb.getString("10-29 items"));
rule.setOption(4, "30->", rb.getString("more than 30 items"));
rules.add(rule);
```

1. Declares a new pricing rule variable.
2. Sets date format to MM/dd/yyyy:HH:mm:ss.
3. Sets ID to an integer number.
4. Sets definition as “LineItem.Quantity”.
5. Sets description as “Ranged Quantity”.
6. Sets the Type as “PricingEngineConstants.VARIABLE_TYPE_NUMBER_RANGE”, a predefined string constant.
7. Sets the variable option number to four.
8. Sets each option by giving the index, range, and a description of the range.

Customized Data Source and a Number Range Input Type

If you want to create a pricing rule using a customized data source, then see your Sterling representative or Sterling-authorized representative.

Customizing Shipping

Shipping rules associate shipping charge discounts with cart order total tiers and associate shipping surcharges with specific line items within a cart. The Sterling Multi-Channel Selling Solution leverages the implementation of pricing rules for the shipping rules. Shipping rules can be customized in the same way as pricing rules. See "Rules" on page 346 for further information.

The following procedures cover the en_US locale. To create lookups for other locales, follow the process you normally use for other localizations.

To Define Shipping Methods Before Creating the Knowledgebase

To define price types before creating the knowledgebase using the SDK:

1. Add the new price types to the lookup list. There must be one lookup for each of the price types you want to add.

   a. In the SDK, retrieve the **LightWeightLookupList** file for customizing:

   ```
   sdk customize WEB-INF/xmldata/Minimal/LightWeightLookupList
   ```

   b. Open the LightWeightLookupList file with a text editor and add your new price types to the file. For example, the following code shows the out-of-the-box lookup entries for Standard Shipping, Premium 2-Day, and Express Overnight price types, as well as an entry for a new price type, International Air:

```
<?xml version="1.0" encoding="UTF-8" ?>
<LightWeightLookupListData>
    ....
    ....
    <LightWeightLookupList state="INSERTED" type="BusinessObject">
         <LightWeightLookup state="INSERTED">
            <LookupType state="INSERTED">ShippingMethod</LookupType>
            <LookupCode state="INSERTED">10</LookupCode>
            <Locale state="INSERTED">en_US</Locale>
            <Description state="INSERTED">Standard Shipping</Descrip-
tion>
        <ExternalLookup state="INSERTED">Standard Shipping</External-
Lookup>
        </LightWeightLookup>
        <LightWeightLookup state="INSERTED">
            <LookupType state="INSERTED">ShippingMethod</LookupType>
            <LookupCode state="INSERTED">20</LookupCode>
            <Locale state="INSERTED">en_US</Locale>
            <Description state="INSERTED">Premium 2-Day</Description>
```

```
        <ExternalLookup state="INSERTED">Premium 2-Day</External-
Lookup>
        </LightWeightLookup>
        <LightWeightLookup state="INSERTED">
            <LookupType state="INSERTED">ShippingMethod</LookupType>
            <LookupCode state="INSERTED">30</LookupCode>
            <Locale state="INSERTED">en_US</Locale>
            <Description state="INSERTED">Express Overnight</Descrip-
tion>
        <ExternalLookup state="INSERTED">Express Overnight</External-
Lookup>
        </LightWeightLookup>
        <LightWeightLookup state="INSERTED">
            <LookupType state="INSERTED">ShippingMethod</LookupType>
            <LookupCode state="INSERTED">40</LookupCode>
            <Locale state="INSERTED">en_US</Locale>
            <Description state="INSERTED">International</Description>
        <ExternalLookup state="INSERTED">International</External-
Lookup>
        </LightWeightLookup>

</LightWeightLookupListData>
```

2. Merge the customizations into the build:

   ```
   sdk merge
   ```

3. Create the knowledgbase:

   ```
   sdk createDB
   ```

4. Load the data:

   ```
   sdk loadDB
   ```

To Add Shipping Methods To an Existing Knowledgebase

To add shipping methods to an existing knowledgebase:

1. Insert an entry in the CMGT_LOOKUPS table. The entry maps a localized description of the shipping method to its numeric code. For example, to insert the lookup entry for a new price type called International Air that has a lookup code of 40 in a SQL Server database:

   ```
   insert into CMGT_LOOKUPS (LOOKUP_TYPE,LOOKUP_CODE,
   DESCRIPTION,LOCALE,ACTIVE_FLAG)
   values ('ShippingMethod',40,'International Air','en_US','Y')
   ```

CHAPTER 24

Testing the Sterling Multi-Channel Selling Solution

This chapter provides a description of the tests that you can perform once implementation is complete.

Starting the Sterling Multi-Channel Selling Solution Server

In general, you can start the Sterling Multi-Channel Selling Solution by starting the servlet container in which the Sterling Multi-Channel Selling Solution is installed. The order in which the servlets load is specified in the Sterling Multi-Channel Selling Solution Web application **web.xml** file and you can read this file in any text editor.

As the Sterling Multi-Channel Selling Solution starts, the servlet console window displays preliminary logging information. Once the Sterling Multi-Channel Selling Solution has initialized its logging environment, then it uses the logging methods to record events. See CHAPTER 10, "Managing Sterling Multi-Channel Selling Solution Logging" and the *Sterling Multi-Channel Selling Solution Developer Guide* for more information about the logging capabilities of the Sterling Multi-Channel Selling Solution.

Troubleshooting

This section covers some basic steps that you must perform to ensure that the system starts correctly. This list is not comprehensive; rather it covers some check points that are a common source of problems. In general, you should troubleshoot your installation using the SDK to ensure that any modifications you make are contained in your project directory. See the *Sterling Multi-Channel Selling Solution Developer Guide* for more information about the SDK and its targets.

To Perform Pre-startup Checks

1. Review the **prefs.xml** configuration file. Check that it is in the correct location as this is the most frequent cause of problems on startup. Remember:
   a. By default, the location of this file is assumed to be ***user_home*/cmgt/debs/conf/** where ***user_home*** is the home directory for the operating system user running the servlet container.
   b. This location can be overridden by:
      - Either: specifying the location of the file as a system property:

      ```
      -Dcomergent.preferences.store=/path/prefs.xml
      ```

      - Or: specifying its location using the comergent.preference.store parameter in the Sterling Multi-Channel Selling Solution **web.xml** configuration file:

      ```
      <init-param>
          <param-name>comergent.preferences.store</param-name>
          <param-value>/path/prefs.xml</param-value>
      </init-param>
      ```

2. Review the **Comergent.xml** configuration file. Check that:
   - It contains the value of system properties that you expect to see (or that are overridden by the **prefs.xml** configuration file).
3. Using the SDK, run the generateDTD target.
   - If you get a series of lines of the form: "Writing DTD for ACL...done!", then the DTDs have been successfully generated. Look in the ***debs_home*/Sterling/WEB-INF/bizobjs/** directory to verify that a complete set of DTDs are there.
   - If you get an error message, then review the steps outlined above.

   See "Data Objects" on page 151 for more information.

4. Using the SDK, run the generateBean target. This should generate all the beans specified by the data objects. If you see any error messages, then you should fix their cause before proceeding.
5. Using the SDK, run the merge target. If this runs successfully, then run the dist target to generate the Web application WAR file.

Error Messages on Startup

When the Sterling Multi-Channel Selling Solution starts, you can see initialization information in either the console window or the servlet container log file. See CHAPTER 8, "Troubleshooting and Backing Up the Sterling Multi-Channel Selling Solution" for a summary of the most likely error messages, together with their causes and how to resolve them.

To troubleshhoot problems with message types, you can set the messageTypeValidate element in the **Comergent.xml** file to “TRUE”.

Runtime Troubleshooting

This section covers some problems identified during testing.

TABLE 22. **Troubleshooting Problems and Solutions**

Problem	Solution
On Solaris, the servlet container cannot find a certain servlet or URL.	First make sure that you did not make a typo. If you are certain that there was no mistake, then do the following: 1. Run the following command on **web.xml**: java com.comergent.dcm.util.CheckWebXML web.xml > newWeb.xml 2. Edit the file **newWeb.xml**. Look for the following string `<!-- (8192) XXX BOUNDARY BREAK -->` The start of the comment <!-- is the start of a 8192 boundary break. If it falls within a value for an XML node, then that node will get truncated. A work around is to pad the **web.xml** file such that the boundary break will fall inside a comment. For more information, see the comments at the start of file **CheckWebXML.java**.

TABLE 22. **Troubleshooting Problems and Solutions (Continued)**

Problem	Solution
You see parser errors such as: `java.lang.NoSuchMethodError at org.apache.xpath.DOM2Helper.getNamespaceOfNode (DOM2Helper.java:348) at org.apache.xml.utils.TreeWalker.startNode (TreeWalker.java:281) at org.apache.xml.utils.TreeWalker.traverse (TreeWalker.java:119) at org.apache.xalan.transformer.TransformerIdentityImpl.transform (TransformerIdentityImpl.java:320)`	Check that you have followed the instructions to copy the XML parser-related JAR files to the servlet container's **lib/** directory, and that you have removed any default **parser.jar** files.
Running iPlanet, you see the following in your browser: `GX Error (GX2GX) socket result code missing!!!`	There is a mismatch between the **web.xml** and **ias-web.xml** files. All servlets mentioned in **web.xml** must have a corresponding entry in the **ias-web.xml** file. Use the kguidgen utility to generate a GUID for the servlet.

Communication Between Enterprise Servers

In testing whether an enterprise server can send price and availability requests and product inquiry list transfer requests to another enterprise server installed at a partner, check for the following problems:

1. Determine if the Message URL defined in the partner profile is correct.
    - On the enterprise server side, you can view the Message URL in the partner profile detail page: check that both the host name and port of the partner's enterprise server are correct. If it is correct, then check that the NamingManager entries of the **Comergent.xml** file connect to the same database specified in the **DataSources.xml** file.
    - If you see a message in the enterprise server log of the form:

      ```
      XML message does not conform to the PriceAvailability.dtd
      ```

      then check the ***debs_home*/Sterling/WEB-INF/bizobjs/** directory to see that the correct DTDs are present.
    - If you see an error displayed in the enterprise server browser window of the form:

`ProcessingFailure`

then the partner’s enterprise server received the price and availability request, but for some reason failed to process it correctly.

On the partner side, by looking at the log or console window, check that the partner’s enterprise server receives price and availability requests from the enterprise server.

2. If the partner’s enterprise server shows no sign of receiving a price and availability request that you initiate from the enterprise server, then:
   - *Either* the Message URL is incorrect or it is not retrieved correctly through the NamingManager.
   - *Or* a network problem is preventing the enterprise server from connecting to the partner’s enterprise server. From your enterprise server, point a browser to the partner’s Message URL: if you cannot obtain a response from the partner’s enterprise server, then a network problem is preventing the two enterprise servers from communicating.
3. If the partner’s enterprise server log or console window indicates that the price and availability request has been received, but an error is generated in processing the request, then you should check that the partner’s enterprise server has correct DTDs in its ***debs_home*****/Sterling/WEB-INF/bizobjs/** directory.

CHAPTER 25 Installing the Mobile Configurator

You can run the Sterling Configurator in a “mobile” mode. Users who install the Mobile Configurator on their laptop computer can configure models and save the configured models. They can subsequently synchronize their configured models back to the online Sterling Multi-Channel Selling Solution, and then complete placing orders for the configured products.

This chapter covers the steps that must be followed on the online Sterling Multi-Channel Selling Solution to set up support for the Mobile Configurator, and the steps that mobile users perform to install the Mobile Configurator on their laptops:

- "Installing the Online Component" on page 367
- "Installing the Mobile Configurator" on page 369

Installing the Online Component

Follow these steps to set up the Online Component of the Mobile Configurator:

Install the Online Component into the SDK

1. Copy the **ComergentMobile.jar** file to a temporary location (referred to here as ***temp_home***)on your SDK system. The full file name will be something like: **ComergentMobile-def-7_1-RC-x-y.jar**, where x and y will depend on the release.

1. At the command line, navigate to the ***sdk_home*** directory.
2. Enter the command to install the ComergentMobile.jar:

   ```
   sdk install temp_home/ComergentMobile-def-7_1-RC-x-y.jar
   ```

3. Enter the command:

   ```
   sdk switchdebs Mobile-def-7_1-RCx-y
   ```

4. Enter the command to create a new project. You can use any name for the project: we refer to this as *project_name*:

   ```
   sdk newproject project_name
   ```

5. Build the project by entering:

   ```
   sdk merge -clean
   ```

6. Build the online component JAR file by entering:

   ```
   sdk project release
   ```

7. Locate the built JAR file in the ***sdk_home*/dist/** directory: it is called ***project_name*.jar**.

Install the Built JAR file into the Online System

8. Copy this JAR file over to the machine on which the online Sterling Multi-Channel Selling Solution is running.
9. On the online Sterling Multi-Channel Selling Solution machine, log in as the user running the servlet container, and create a directory called: ***debs_home*/Sterling/WEB-INF/data/synchmaster/**.

Note	If you are working in a clustered environment, then you must copy the JAR file to a shared location.

10. Unjar the built JAR file into the ***debs_home*/Sterling/WEB-INF/data/synchmaster/**directory.

Configure the Online System for Mobile Support

11. Log into the Sterling Multi-Channel Selling Solution as an enterprise administrator.
12. Click **Business Rules**.
13. Click **Configurator**.
14. Set the Is Offline Configuration and Quoting Installed? business rule to true.

15. Click **Save All and Return to List**.

Enable Users for Mobile Access

16. Determine which users you want to provide mobile access for. For each such user, you must access their user profile and check the check box for Offline Access.
17. Save your change to their user profile.

Installing the Mobile Configurator

Follow these instructions to install the Mobile Configurator on your machine.

Requirements

The Mobile Configurator can be installed on any personal computer that meets the following requirements:

- Operating system: Windows 2000 or Windows XP
- Memory: 128 MB
- Hard drive space: 10 MB
- Java Development Kit: you must install the Sun Java Development Kit (JDK) Version 1.4.2, 1.5, or subsequent compatible version.

To Install the Mobile Configurator

1. Identify a directory on your laptop machine which you will use for running the Mobile Configurator: we will call this directory ***mobile_home***.
2. Check that you have been enabled for mobile access to the Sterling Multi-Channel Selling Solution.
3. Log in to the Sterling Multi-Channel Selling Solution and scroll down the home page to the instructions in the Install the Offline Configuration and Quoting Tool panel.
4. Right-click the link to **synchro.jar** and download the file to the ***mobile_home*** directory.
5. Run the **Launcher** application:
   - If you have set up JAR files to be associated with the Java application that comes with your JDK, then you can use Windows Explorer to navigate to the ***mobile_home*** directory, and then double-click the **synchro.jar** file.

- Alternatively, at the command line, navigate to the ***mobile_home*** directory and run:

```
java -jar synchro.jar
```

6. The Setup window is displayed.

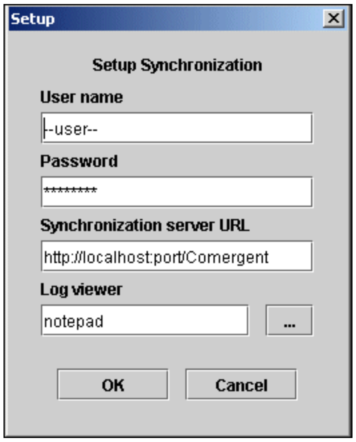


FIGURE 20. **Setup Window**

7. Enter your username and password that you use to log into the online Sterling Multi-Channel Selling Solution.
8. Enter the Synchronization URL as it is displayed to you in the Install the Offline Configuration and Quoting Tool panel.
9. Optionally, specify an application to view the log file for the Mobile Configurator.
10. Click **OK**.
11. The Synchronization Required window is displayed. Click **Yes**.
12. Once synchronization is complete, you are prompted to quit the application and restart it.

13. When it restarts, the Launcher application will again prompt you to synchronize files, so click **Yes**.

Note: If you get an out of memory exception, you must boost the memory allocation for synchro.jar. Open a command window and run the following command from the ***mobile_home*** directory:

```
java -jar -Xmx256M synchro.jar
```

14. The Synchronization - Progress window displays progress as it synchronizes files and data between the online Sterling Multi-Channel Selling Solution and your Mobile Configurator.
15. A dialog box appears when synchronization completes. Click **OK**.
16. The Launcher window is displayed.

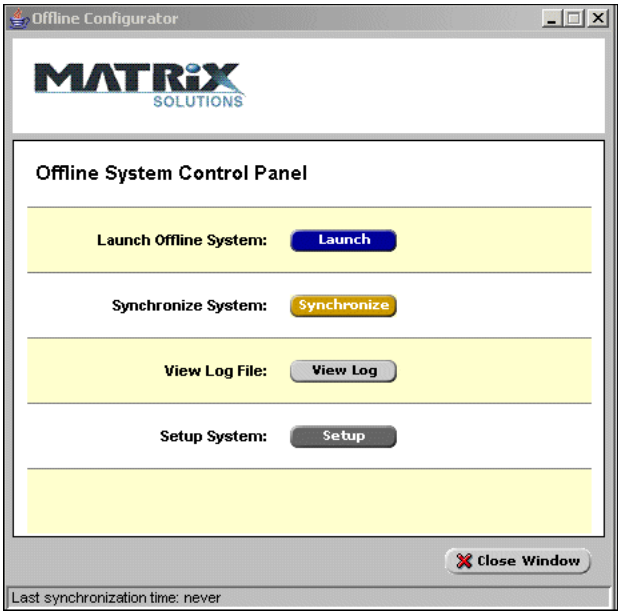


FIGURE 21. **Launcher Window**

17. Click **Launch**. A browser window is opened and the Mobile Configurator home page displays.

If you see the Mobile Configurator home page, then you have successfully installed Mobile Configurator.

Troubleshooting

Problems During Installation

If the installation program fails to find your installation of the JDK, then make sure that you have the following environment variables set to correctly point to the location of the JDK: JAVA_HOME and JDK_HOME. Typically, their values should be set to something like “C:\jdk1.5”.

1. On your Windows desktop, right-click the My Computer icon and select Properties.
2. Click the Advanced tab.
3. Click **Environment Variables...**.
4. In the User variables section, click **New...**.
5. In the Variable Name field, enter “JAVA_HOME”.
6. In the Variable Value field, enter the directory location of the JDK.
7. Click **OK**.
8. Click **OK**.
9. Click **OK**.

Problems After Installation

If the Mobile Configurator fails to start up when you click **Launch** from the Launcher window, then check that you do not have a server process already listening on port 8080. If you do, then you must either stop this server process or modify the port at which the Mobile Configurator listens. You can modify the **server.xml** file in the ***mobile_home*/jakarta-tomcat-5.5.9/conf/** directory by changing the value of the port attribute in this element:

```
<Connector
    className="org.apache.catalina.connector.http.HttpConnector"
    port="8080" minProcessors="5" maxProcessors="75"
    enableLookups="false" redirectPort="8443"
    acceptCount="10" connectionTimeout="60000" debug="0"
    scheme="http" secure="false"/>
```

Loading Models and Prices

In order to configure models using the Mobile Configurator, you must load the XML files for each model and the pricing information to be used with your Mobile Configurator on your machine. Typically, this task is performed as part of a “synchronization” with an existing installation of the Sterling Multi-Channel Selling Solution.

This synchronization process can be initiated by starting the Launcher application, and then click **Synchronize**.

CHAPTER 26

Installing a Clustered Implementation

This chapter describes how to set up the Sterling Multi-Channel Selling Solution in a clustered environment. It covers:

- "General Steps" on page 375
- "Setting Up a WebLogic Cluster" on page 381
- "Setting Up a WebSphere Cluster" on page 388

In addition to following the steps described in one of the servlet container sections, you should also set up the global cache using JavaSpaces. See:

- "Setting up a Database for Caching" on page 394
- "Setting up JavaSpaces for Caching" on page 394

General Steps

Terminology and Overview

A cluster provides an environment that supports higher performance and reliability than a single machine can. Typically, a cluster comprises two or more member machines that from the outside world appear to work as one machine: when users submit a request to the cluster URL, they are not aware of which machine in the cluster processes the request and returns the response.

The cluster URL is usually directed to a Web server that sits “in front” of the cluster: this Web server provides the entry point for users, and it is responsible for distributing the requests to cluster members as requests come in. The Web server acts as a load balancer and distributes requests using an algorithm to determine which cluster member machine should receive each inbound request.

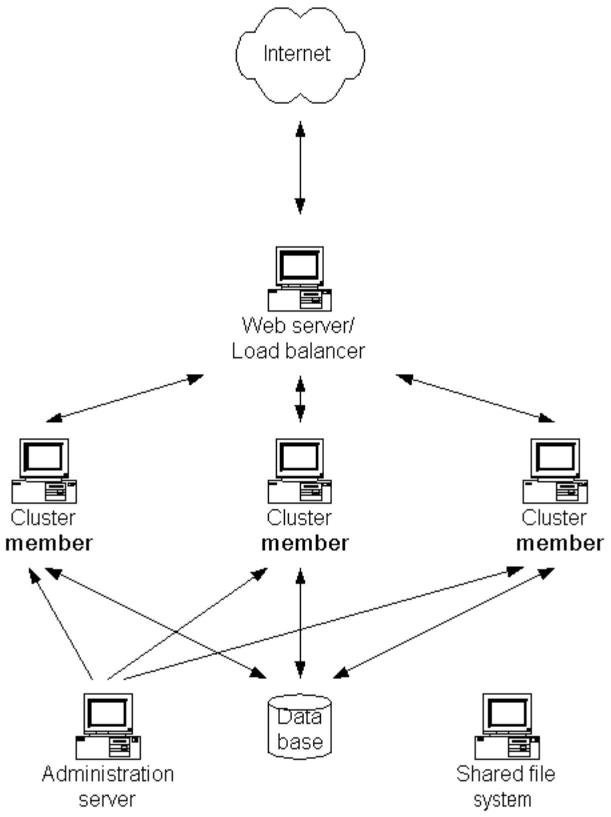


FIGURE 22. General Cluster Configuration

Administration Servers

In some cluster configurations, each cluster member is effectively independent of the others: you install the Sterling Multi-Channel Selling Solution into each cluster member and configure it independently of the other members of the cluster. Other cluster configurations make use of an administration server: this is a machine that manages the cluster. Cluster members are typically registered with the administration server and the administration server maintains a single image of the Sterling Multi-Channel Selling Solution. When a machine joins the cluster, the administration server pushes a copy of the Web application to the new cluster member. In this case, each cluster member has the same configuration information because it has been pushed to them from the administration server.

The Sterling Multi-Channel Selling Solution uses Ehcache to provide the notification mechanism required to synchronize cluster members.

Shared Files

To ensure that cluster members behave consistently with each other, they must access configuration files, templates, and image files that are common to all members of the cluster. You do this by establishing a shared file server and point to a common location on this file server.

- On UNIX systems, use an NFS file system to share common files. For example:

  ```
  <context-param>
      <param-name>WritableDirectory.share.public.loadable
      </param-name>
      <param-value>/usr/Comergent/shared</param-value>
  </context-param>
  ```

- On Windows systems, use one of two methods to set up a shared file server.
  - Using one method, on each cluster member you map the same drive letter to the shared file server, then use the drive letter to provide a common reference to the location of the shared files. For example:

    ```
    <context-param>
        <param-name>WritableDirectory.share.public.loadable
        </param-name>
        <param-value>T:/Comergent/shared</param-value>
    </context-param>
    ```

    Here, the T: drive on each machine has been mapped to the C: drive on the file server machine.
  - Or, using the other method, use the UNC convention to refer to the shared directory location. For example:

    ```
    <context-param>
        <param-name>WritableDirectory.share.public.loadable
        </param-name>
        <param-value>\\fileserver\Comergent\shared</param-value>
    ```

```
        </context-param>
```

Load Balancer

If you run your cluster using a load-balancing solution (either a hardware- or software-based solution), then make sure that the load-balancing is done in a session-sticky fashion. That is, all requests relating to a session should be handled by the same member machine in the cluster.

General Installation Instructions for Clustered Deployment

1. Depending on the cluster architecture, install the Sterling Multi-Channel Selling Solution on each instance or into the Administrator server that deploys the Web application to the managed servers.
2. If you are using SQL Server as the Knowledgebase database server, then make sure that you set the ServerId system property and element of the **DataServices.xml** file to a unique two-digit value on each machine that makes up the cluster. This ensures that generated keys are managed correctly. See "Support for SQL Server" on page 85 for more information.
3. If you are using 2-way encryption anywhere in the implementation, then follow these steps:
    a. Make sure that you start one of the machines before the others.
    b. Perform a persist operation that requires the use of 2-way encryption.
    c. Identify the location of the **dcmsKey.ser** file on this machine and copy this file to the corresponding location on the other machines of the cluster. See "Storing Data in Encrypted Form" on page 200 for more information.
4. Follow the steps described in "Sharing Directories" on page 93.
5. As a site administrator, set the value of the useSessionCaching system property to “true”. This property is in the Profile Manager section of the system properties.
6. Enable your Sterling Multi-Channel Selling Solution implementation as a distributed implementation as follows:
    a. As a site administrator, set the value of the GlobalCache: Implementation Class system property to com.comergent.globalcache.DistributedCache. This property is in the GlobalCache section of the system properties.

       This tells the Sterling Multi-Channel Selling Solution to use the Ehcache configuration file **WEB-INF\properties\DistributedCache-Config.xml**.

b. Enable the DistributedEventService by uncommenting the RefreshServiceHelper listener code in the **WEB-INF/web.xml** configuration file:

```
<!-- Start of Listeners -->
  <listener>
    <listener-class>
     com.comergent.reference.appservices.cache.CacheManagersHelper
    </listener-class>
  </listener>
<!-- comment this out to allow preferences refresh event to propagate
     to other nodes -->
<!--
   <listener>
     <listener-class>
      com.comergent.reference.appservices.cache.RefreshServiceHelper
      </listener-class>
   </listener>
-->
   <listener>
     <listener-class>
       com.comergent.dcm.core.SessionMonitor
     </listener-class>
    </listener>
<!-- End of Listeners -->
```

c. As a site administrator, set the value of the cronRefreshTime property. The cronRefreshTime property specifies the polling interval, in seconds, at which a node should check for modified or added cron jobs. Set the value of this property in the Job Scheduler refresh time in seconds field of the Job Scheduler section of the system properties. The default value, -1, prevents the node from periodically checking for changes to cron jobs.

7. By default, distributed nodes are discovered automatically using the Ehcache configuration for both the GlobalCache and EventService. However, you can also modify the cacheManagerPeerProviderFactory property settings for multicastGroupAddress and multicastGroupPort in the **\WEB-INF\properties\DistributedGlobalCache-Config.xml** and **WEB-INF\classes\DistributedEventService-config.xml** files to specify the unique IP addresses and ports for a cluster to adjust the scoping of the discovery mechanism.

```
<cacheManagerPeerProviderFactory
      class="net.sf.ehcache.distribution.RMICacheManagerPeerPro-
viderFactory"
      properties="peerDiscovery=automatic,
                  multicastGroupAddress=230.0.0.1,
                multicastGroupPort=4567, timeToLive=1" />
```

You can also modify the timeToLive property setting to restrict how far packets should go. The setting values are:

- 0 - the same host
- 1 - the same subnet
- 32 - the same site
- 64 - the same region
- 128 - the same continent
- 255 - unrestricted

The default timeToLive value is 1, the same subnet.

The GlobalCache and EventService configuration must be the same on each cluster node, and must be unique for each cluster. For example, if you have two separate clusters, each cluster's configuration must be consistent across that cluster's nodes. The clusters themselves must each have unique configurations so that they do not conflict.

8. Copy the **prefs.xml** configuration file to a shared location which is visible to all member machines of the cluster. The location of the file must be specified in the startup script for each cluster member as follows:

   ```
   -Dcomergent.preferences.store=<Path to prefs.xml>
   ```

9. Configure the cluster to check for new and updated files as soon as possible. This ensures that all servers are in sync and will serve the same information to customers accessing your site. This is especially important in ensuring that the latest generated product index file is available at all times.

   Place your configuration property XML files in a shared location accessible by all member machines of the cluster. Then, activate the AutoReload element of the **SearchConfigurationProperties.xml** configuration file as follows:

   ```
   <AutoReload activated="true" reloadFilePeriod="30"/>
   ```

   This activates the AutoReload function and instructs the cluster to check for updates every 30 seconds.

10. Follow any remaining steps required by your servlet container or load balancer to implement their specific solution. See:

    a. "Setting Up a WebLogic Cluster" on page 381

b. "Setting Up a WebSphere Cluster" on page 388

Contact your Sterling Commerce representative for information about setting up other clustering architectures.

Setting Up a WebLogic Cluster

You can use the clustering capabilities to set up a cluster of WebLogic Release 9.2 servlet containers to run your implementation of the Sterling Multi-Channel Selling Solution. In general, you should follow the instructions provided by BEA Systems to set up the cluster. This section provides some additional information used to install the Sterling Multi-Channel Selling Solution in the cluster.

Web Server

We suggest that you set up the cluster by placing a Web server or separate WebLogic Server as a front-end to the servlet container cluster. You should choose one of these options:

1. Set up a Web server with the appropriate WebLogic Web server plug-in. Supported Web servers include Apache and Microsoft Internet Information Server.

Note:	If you use Apache, ensure that your Apache release matches the mod_wl_20.so version. At this time of writing (October 2003), Apache 2.0.42 works with the current mod_wl_20.so provided by WebLogic.

2. Set up a WebLogic Server with the HttpClusterServlet Web application. The HttpClusterServlet maintains the list of all servers in the cluster, as well as the load balancing logic to use when accessing the cluster.

When the user’s browser makes a request, the Web server or HttpClusterServlet proxies the request to the WebLogic Server cluster. See the WebLogic documentation for further details.

Administration and Managed Servers

Typically, a WebLogic cluster comprises an Administration Server and one or more Managed Servers. The Web applications are deployed into the Administration Server and then as Managed Servers start or join the cluster, the Administration Server deploys the Web applications to each Managed Server. Consequently, you must deploy the Sterling Multi-Channel Selling Solution Web application **Sterling.war** file into the Administration Server first.

Note that when a Managed Server restarts, the Administration Server redeploys the Web applications to the Managed Server: this can take a considerable time, and so you should restart servers at times that ensure that they can be offline for the time they need to restart.

Preparation to Deploy the Sterling Multi-Channel Selling Solution Web Application

Because the same WAR file is used to deploy to all cluster members, you must make sure that this WAR file is correctly configured before you deploy the WAR file to the Administration Server. In particular:

1. Make sure that you have used the SDK to build the deployment WAR file.
2. While using the SDK, make sure that the following configuration properties are correctly set:
   a. **web.xml**: make sure that the WritableDirectory parameters are correctly set to point to the shared directory location. See "Common Directories" on page 384 for more information. Make sure that you have declared the SharedPublicServlet class as described in "SharedPublicServlet Class" on page 386.
   b. **weblogic.xml**: make sure that you have added a **weblogic.xml** file to the ***sdk_home*/projects/*project*/WEB-INF/** directory. See the example file in "WebLogic Releases" on page 65. To support session-sharing across the cluster members, consider adding the element described in "Session Sharing" on page 386.
   c. Make sure that you have correctly specified the database connection information in the appropriate properties file so that they are correctly set in the **prefs.xml** configuration file.
3. Build the **Sterling.war** file using the SDK distWar target.
4. Copy the **prefs.xml** configuration file to a shared location which is visible to all member machines of the cluster. The location of the file must be specified in the startup script for each cluster member as follows:

```
-Dcomergent.preferences.store=<Path to prefs.xml>
```

Deploying the Sterling Multi-Channel Selling Solution Web Application

Follow these steps to deploy the Sterling Multi-Channel Selling Solution Web application into the cluster. These instructions assume that you have set up the cluster using the WebLogic administration console on the Administration Server: we refer to the name of the cluster as *cluster_name*. We also assume that the managed servers are up and running. Make sure that you have used the SDK to create the **Sterling.war** file and that you have moved a copy of the file to a location on the Administration Server.

1. Log into the administration console of the WebLogic Administration Server.
2. Click **Servers** and verify that the managed servers are listed.
3. Click **Clusters** and verify that the name of the target cluster is *cluster_name*.
4. Click **Lock & Edit**.
5. Click **Deployments**.
6. Click **Install**.
7. In the next window, navigate to the location of the **Sterling.war** file and select the radio button next to the **Sterling.war** file name.
8. Click **Next**.
9. Select the **Install this deployment as an application** radio button.
10. Click **Next**.
11. Check the check box next to the cluster named *cluster_name*. By default, the **All servers in the cluster** radio button is selected. You should usually leave this setting unchanged.
12. Click **Next**.
13. In the **Name** field of the General panel on the Optional Settings page, enter the name of your Sterling deployment, for example, Sterling. Accept the defaults for the other values on the Optional Settings page.
14. Click **Next** to review your choices, then click **Finish** to complete the deployment.
15. Click **Activate Changes** to activate the deployment.

    Deployment can take ten to twenty minutes. At the end of the deployment process, a page displays a Success message.

SQL Server

Because more than one deployment of the Sterling Multi-Channel Selling Solution is accessing the same Knowledgebase on SQL Server, you must set a two-digit server ID for each deployment. You must modify the servlet container command or script that starts the servlet container on each machine so that a Java system property is set: Comergent.DataServices.General.ServerId. This should be set on each machine so that each has a unique value: 01, 02, and so on.

For example, in a Tomcat installation, you can modify the starting batch file to include:

```
set JAVA_OPTS=-DComergent.DataServices.General.ServerId=12
```

Cron Jobs

The Sterling Multi-Channel Selling Solution distinguishes between system cron jobs and application cron jobs. Typically, system cron jobs are run without an associated user and run on every system in a clustered environment whereas application cron jobs must be run associated to a user and usually should be run only by one machine in a cluster.

To set this up, you must do the following:

1. Make sure that in the deployment WAR file, the value of the cronApps system configuration property is set to “system”.
2. For the one application server that should run application cron jobs, make sure that a system property is set as follows:

   ```
   -DComergent.Cron.cronApps=both
   ```

   For example, in a Tomcat installation, you can modify the starting batch file to include:

   ```
   set JAVA_OPTS=-DComergent.Cron.cronApps=both
   ```

   Note that how you do this will vary from one servlet container to another: see for example, "Command Line Settings" on page 392 for WebSphere instructions. Note that the valid values for this property are: “application”, “both”, “none”, and “system”.
3. Set the value of the Cron Job URL system property to the value of the URL used to access the cluster: for example:

   ```
   http://loadbalancer/Sterling/msg/matrix
   ```

Common Directories

All the Managed Servers in the cluster must be able to access the same directory locations in the file system: this is where configuration files, shared data files, and other related files such as pagination data is stored for the cluster. You must ensure that all members of the cluster access this location using the same directory paths.

The location of the shared directories is specified in the Sterling Multi-Channel Selling Solution **web.xml** file using context parameter elements of this form:

```
<context-param>
    <param-name>WritableDirectory.share.public.loadable</param-name>
    <param-value>/tmp</param-value>
</context-param>
<context-param>
    <param-name>WritableDirectory.share.public.noloadable
    </param-name>
    <param-value>/tmp</param-value>
</context-param>
<context-param>
    <param-name>WritableDirectory.share.private.loadable</param-name>
    <param-value>/tmp</param-value>
</context-param>
<context-param>
    <param-name>WritableDirectory.share.private.noloadable
    </param-name>
    <param-value>/tmp</param-value>
</context-param>
```

See "Shared Files" on page 377 for the form that the values of these parameters can take. Note that by default, these elements are commented out: in this case, each instance of the Sterling Multi-Channel Selling Solution Web application acts independently of the other instances in the cluster. All file accesses are performed locally on the machine running the Web application.

The following table summarizes which files should go where:

TABLE 23. Shared File Locations

Location	Purpose
share.public.loadable	Do not use.
share.public.noloadable	Image files and other files that should be accessible to Web servers to serve up static content. Examples include GIF files associated with promotions and storefront partners.
share.private.loadable	Class files to be shared across the cluster: this directory is used primarily for ***Sterling*** Configurator and Visual Modeler.
share.private.noloadable	Configuration files, pagination files, and other files that must be shared across the cluster, but which should not be accessible from users' browsers.

SharedPublicServlet Class

You must uncomment in the element that declares the SharedPublicServlet class: this class is used to serve up static content such as partner logos and promotion images that are uploaded to the Sterling Multi-Channel Selling Solution.

```
<servlet>
    <servlet-name>SharedPublicServlet</servlet-name>
    <servlet-class>
        com.comergent.dcm.core.SharedPublicServlet
    </servlet-class>
</servlet>
```

You must also uncomment in the following elements that map URLs to the SharedPublicServlet:

```
<servlet-mapping>
    <servlet-name>SharedPublicServlet</servlet-name>
    <url-pattern>/htdocs/partnerlogos/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
    <servlet-name>SharedPublicServlet</servlet-name>
    <url-pattern>/htdocs/promotions/images/*</url-pattern>
</servlet-mapping>
```

For each supported locale, uncomment in the corresponding element:

```
<servlet-mapping>
    <servlet-name>SharedPublicServlet</servlet-name>
    <url-pattern>/la/CO/htdocs/*</url-pattern>
</servlet-mapping>
```

For example, uncomment in the following element for the en_US locale:

```
<servlet-mapping>
    <servlet-name>SharedPublicServlet</servlet-name>
    <url-pattern>/en/US/htdocs/*</url-pattern>
</servlet-mapping>
```

Session Sharing

You must also provide information about how sessions are to be shared across the cluster using the **weblogic.xml** deployment file. You may have already created this file to pass in information about the WebLogic environment or you may have to create it only for this purpose. It should be located in your **Sterling.war** Web application file at the same level as the **web.xml** file.

You must add the following fragment to the **weblogic.xml** file:

```
<session-descriptor>
```

```
    <session-param>
        <param-name>PersistentStoreType</param-name>
        <param-value>file</param-value>
    </session-param>
    <session-param>
        <param-name>PersistentStoreDir</param-name>
        <param-value>
            <Directory location common to all members of cluster>
        </param-value>
    </session-param>
</session-descriptor>
```

Note that a more common setting is:

```
<session-param>
    <param-name>PersistentStoreType</param-name>
    <param-value>memory</param-value>
</session-param>
```

This setting does not support session-failover.

Reloading Files

If shared configuration files can be updated, then each managed server may need to reload the shared copy to pick up changes made by other servers in the cluster. For example, the **SearchConfigurationProperties.xml** file has a setting:

```
<SearchSystemConfigurations>
    <AutoReload activated="true" reloadFilePeriod="30"/>
</SearchSystemConfigurations>
```

Set the activated attribute to "true" and set the reloadFilePeriod attribute to an interval (in seconds) to specify that if an interval of more than 30 seconds elapses between accesses, then the file should be reloaded.

Running a Clustered WebLogic Installation

In a clustered deployment of WebLogic, you must also perform these steps to ensure that the DTDs used by the Sterling Multi-Channel Selling Solution are correctly located. On each machine in the cluster:

1. Create or identify a designated directory that may be used to store the DTDs. For example, you can create a sub-directory called ***container_home*/local/working/** in each WebLogic installation.
2. Unjar the Sterling Multi-Channel Selling Solution WAR file, and copy the DTD files from their locations under **WEB-INF/** to the designated directory.
3. Modify **startManagedWebLogic.cmd** or **startManagedWebLogic.sh** to set a new runtime flag: `-DComergent.workingDir`. You can use the

$WL_HOME variable if the designated directory lies under the ***container_home/*** location. For example:

```
-DComergent.workingDir=$WL_HOME/local/working
```

Setting Up a WebSphere Cluster

This section describes how to set up a WebSphere Release 5.1 cluster and to deploy the Sterling Multi-Channel Selling Solution to the cluster.

General

Releases of the Sterling Multi-Channel Selling Solution before Release 6.4.1 have known issues with WebSphere and clustering: you should make sure that you have applied hot fixes to your release that ensure that the SDK builds the WAR file for WebSphere correctly, and so that configuration properties can be passed to the Sterling Multi-Channel Selling Solution by setting command line parameters. Contact your Sterling Commerce representative for further details.

Setting up a WebSphere Cluster

A typical WebSphere cluster comprises:

- A deployment manager server
- two or more applications servers running on the same or different machines: these are the cluster member servers
- a load balancing server: this is the Web server to which users point their browsers and which distributes requests to the cluster member servers.

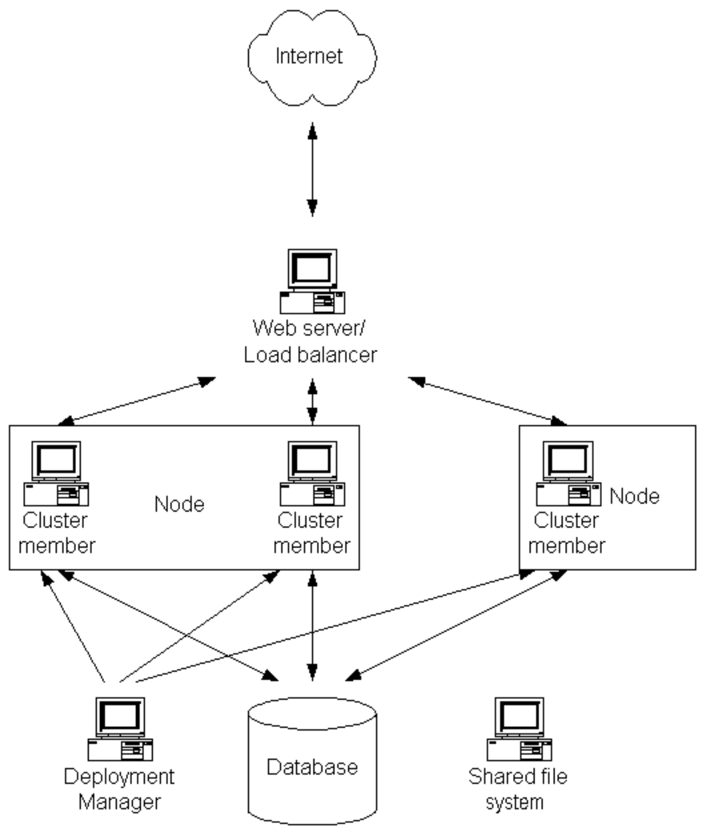


FIGURE 23. **WebSphere Cluster Configuration**

1. Set up the deployment manager on a machine in your network. It need not run on the same machine on any of the application servers, but the deployment manager and application servers should all be running on the same segment of a local area network. Start the deployment manager (either through the UI or at the command line in ***websphere_home*/AppServer/bin/**, enter: startServer dmgr).

   You use the deployment manager to manage the WebSphere cell: this is where the cluster is run.

2. Install your application servers on the cluster member machines. Each cluster member machine is referred to as a node. You can create one or more application servers on each node.
3. Start the node agent on each node: typically you do this at the command line, by navigating to ***websphere_home*/AppServer/bin/** and entering: startNode.

   The node agent is what the deployment manager uses to communicate with each node in its cell.
4. Log in to the Deployment Manager administration UI.
5. Add nodes to the cell by clicking **System Administration -> Nodes**, and clicking **Add Node** in the Nodes panel.
6. For each node, add the application servers running on the node to the cell by clicking **Servers -> Application Servers**, and clicking **New** in the Application Servers panel. You must select each node in turn and specify a unique name for each server on the node.

When you have completed creating all the application servers, then you must create the cluster.

7. In the Deployment Manager UI, click **Servers -> Clusters**, and then click **New** in the Server Cluster panel.
8. Enter a name for the cluster and then without add servers to the cluster complete the creation of the cluster.
9. Once the cluster is created, select the cluster form the list of clusters. On the Server Cluster page, click Cluster members and use the Cluster members page to add application servers to the cluster.

When you have added all the application servers to the cluster, check that you can start and stop the cluster by clicking **Servers -> Clusters**, check the check box for the new cluster, and click **Start**. Verify that the cluster starts without any errors, and then click **Stop**.

Building the Sterling Multi-Channel Selling Solution Deployment WAR File

You must build the deployment WAR file using the SDK, and must make sure that the cluster settings have been made before you build the WAR file for deployment. Typically, this means that you must customize the **web.xml** and **Comergent.xml** configuration files to set the cluster settings and the location of the shared directories.

Attention:	You must do this in the SDK because once the WAR file is deployed, WebSphere processes the configuration parameters by assigning them IDs. If the cluster settings are commented out, then they will be missed.

Releases prior to Release 6.4.1 should be built using the patched version of the SDK 2.0.4, and you must make sure that you applied any patches relating to your release: notably to support WebSphere and to allow for configuration properties to be set at the command line.

Deploying the Sterling Multi-Channel Selling Solution Web Application

Now that you have created the cluster and verified that the cluster can be stopped and started successfully, you are ready to deploy the Sterling Multi-Channel Selling Solution Web application to the cluster. Make sure that you have built the deployment WAR file as described in "Building the Sterling Multi-Channel Selling Solution Deployment WAR File" on page 390.

1. Copy the deployment WAR file to the deployment manager machine.
2. Log in to the Deployment Manager UI.
3. Click **Applications -> Install New Application**.
4. On the Preparing for the application installation page, select the **Server path** radio button, and click **Browse...** to where you copied the deployment WAR file. Select the radio button next to the WAR file. Click **OK**.
5. Specify a context root, typically of the form: /Sterling.
6. Click **Next**.
7. On the next page, click **Next**.
8. On the next page, click **Continue**.
9. On the Install New Application page, you can change the name of the application, or just click **Next**.
10. On the next page, select a virtual host for the Sterling Product Suite Web application and click **Next**.

11. On the next page, you must specify that the application is to be deployed to the cluster. You do this by selecting the cluster in the list box of Clusters and Servers, checking the Sterling Product Suite check box, and then click **Apply**.
12. Click **Next**.
13. On the next page, check the deployment details for the new application, and then click **Finish**.
14. After the Web application finished deploying, then click **Save** to save the details of the new deployment to the master configuration.

When the WAR file is deployed to the cluster members, it is expanded into a directory ***websphere_home*/AppServer/installedApps/** sub-directory.

Configuration

You must do the following configuration steps before you can start the cluster and access the new deployment.

Updating the Web Server Plugin

The load-balancing Web server makes use of a configuration XML file to know which requests it receives should be forwarded to the cluster and which machines are in the cluster. This file is called **plugin-cfg.xml** and you must regenerate it after deploying the Web application to the cluster.

1. Log in to the Deployment Manager UI.
2. Click **Environment -> Update Web Server Plugin**.
3. Click **OK**.

The new version of the file is created on the Deployment Manager machine as ***websphere_home*/DeploymentManager/config/cells/plugin-cfg.xml**.

4. Copy this file to the appropriate location on the load-balancer Web server machine. The precise destination location depends on your Web server solution. For example, if you use the IBM HTTPServer that is part of the IBM WebSphere suite, then you must copy it to the ***websphere_home*/AppServer/config/cells/plugin-cfg.xml** location on the Web server machine.
5. Restart the Web server to pick up the changed configuration file.

Command Line Settings

All of the application servers in the cluster will deploy exactly the same form of the Sterling Multi-Channel Selling Solution Web application in them. Typically, there are some configuration settings that must be set at the application server level: for example, the for SQL Server ServerId setting described in "General Installation Instructions for Clustered Deployment" on page 378 and cron job settings that determine whether application cron jobs run on the cluster member. To set these:

1. Log in to the Deployment Manager UI.
2. Click **Application Servers**.
3. For each application server in the cluster, repeat these steps:
    a. On the Application Servers page, click the link to the application server.
    b. Under Additional Properties, click **Process Definition**.
    c. Under Additional Properties, click **Java Virtual Machine**.
    d. In the Generic JVM arguments text field, enter the application server properties as appropriate. Note that you only have to do this for properties whose value must be different from that set in the deployed Web application.

       For example, to specify that application and system cron jobs should be run on this application server, enter:
       ```
       -DComergent.Cron.cronApps=both
       ```
       To specify a ServerId value, enter:
       ```
       -DDataServices.General.ServerId=12
       ```
    e. Click **OK**.
4. Once you have completed this step for all members of the cluster, then Save these changes to the master configuration.
5. Stop and then restart the cluster.

Testing the WebSphere Cluster

You can test that the cluster is working correctly simply by pointing your browser to the load-balancing Web server, and verify that you can access the Sterling Multi-Channel Selling Solution Web application using the context name that you specified when you deployed the Web application as described in "Deploying the Sterling Multi-Channel Selling Solution Web Application" on page 391. For example:

http://loadbalancer/Sterling/en/US/enterpriseMgr/matrix

By logging in simultaneously from different browsers, and by examining the application server logs on each application server, you should be able to verify that the browser requests are being distributed among the application servers by the load-balancer. You should also be able to verify that if an application server goes down, that the load-balancer stops routing requests to that application server.

Setting up a Database for Caching

Introduction

This implementation of the distributed Global Cache uses the Knowledgebase database server to store session information. Note that in Release 7.1, only implementations that use the Oracle database server are supported.

1. Log in to the Sterling Multi-Channel Selling Solution system administration site as a site administrator.

   Your system administration site URL is similar to:
   `http://`*`server`*`:`*`port`*`/Sterling/en/US/enterpriseMgr/admin`
2. Click **System Services**.
3. Click **Commerce Manager**.
4. In the GlobalCache: Class Name property field, enter:

   com.comergent.dcm.cache.impl.db.DBCache
5. Click **Save All and return to List**.

Setting up JavaSpaces for Caching

Introduction

This implementation of the distributed Global Cache uses the JavaSpaces technology from Sun Microsystems. This requires a dedicated machine to run the Jini Lookup server, the Javaspaces server, and optionally the transaction server. The steps needed to install and run the JavaSpaces server are described below.

Install the Required Servers

All the Jini servers have an implementation that can be activated using the **rmid** (RMI activation daemon). This means that the servers need to be registered with the rmid once, after that they will be automatically restarted if they crash or the machine has been rebooted. The system administrator needs to make sure that the **rmid** daemon is running at all times. In the following section we describe the steps needed to install the Jini Lookup server and JavaSpaces server, and to register the servers with the **rmid** daemon for the first time.

To Implement JavaSpaces

1. Download the Jini Starter Kit v1.2.1_001 from http://wwws.sun.com/software/communitysource/jini/download.html
2. Install the Jini Starter Kit by unzipping the file. This creates a directory called **jini1_2_1_001/** (this is version-dependent, so you may see slightly different numbers). In the following instructions we refer to this directory as ***jini_home***.
3. Create a logs directory where you want the servers to store their logs: we will refer to it ***logs_home***. This directory can be anywhere on the machine.
4. Start the **rmid** with the following arguments:

   ```
   rmid -d log jini_home/logs -J-Djava.security.policy=none
   ```

5. You need to make the jar files with '–dl' postfix accessible via a Web server, you can do this by one of the following steps:
   a. Either, run the supplied Web server using the following command (all on one line):

      ```
      java -jar jini_home/lib/tools.jar -port <port>
      -dir jini_home/jini1_2_1/lib
      ```

      You should select a value for <*port*> that is private to your network.
   b. Or, copy all the ***jini_home*/lib/*-dl.jar** files to one of your current Web servers.
6. At the command line, navigate to the ***jini_home*** directory.
7. Run the Jini Lookup server by entering:

   ```
   java -jar ./lib/reggie.jar http://<host>:<port>/reggie-dl.jar ./
   policy/policy.reggie logs_home/reggie_log public
   ```

   Replace <*host*> by the name of the machine and <*port*> by the value specified in Step 5a or the port at which the Web server you selected in Step 5b is listening.
8. (Optional) Run the Transaction server by entering:

   ```
   java -jar
   -Djava.security.policy=./policy/policy.all
   -Dcom.sun.jini.mahalo.managerName=TransactionManager ./lib/mah-
   alo.jar http://%HOST%:%PORT%/mahalo-dl.jar ./policy/policy.all
   logs_home/txn_log public
   ```

9. Run the JavaSpaces server by entering:

   ```
   java -jar -Djava.security.policy=./policy/policy.all
   -Dcom.sun.jini.outrigger.spaceName=JavaSpace ./lib/outrigger.jar
   http://<host>:<port>/outrigger-dl.jar ./policy/policy.all
   logs_home/frontendspace_log public
   ```

If you delete the ***logs_home***/ directory, or if the directory is not available, then the **rmid** will not be able the restart the servers.

To Set Up the Sterling Multi-Channel Selling Solution

1. Change the globalCacheImplClass property in **Comergent.xml** from:

   ```
   com.comergent.dcm.cache.impl.AppContextCache
   ```

   to:

   ```
   com.comergent.dcm.cache.impl.space.SpacesCache
   ```

2. Add the globalCacheParameters property to **Comergent.xml** in the General group:

   ```
   <globalCacheParameters controlType="text"
   runtimeDisplayed="true" ChangeOnlyAtBootTime="true"
   visible="true" boxsize="60" displayQuestion="GlobalCache: Parame-
   ters to be passed to the Global Cache implementation" default-
   Choice="" help="Enter a comma separated key value pairs to be
   passed to the global cache implementation">
   javaspacesname=JavaSpace,transactionservername=TransactionManager
   </globalCacheParameters>
   ```

   The javaspacesname and transactionservername are the properties used by the Sterling Multi-Channel Selling Solution to find both the JavaSpaces server and the Transaction server respectively. Those are the same names used when the servers are started above. If the transactionservername is set to an empty String, or is not defined, then the Sterling Multi-Channel Selling Solution will not use the Transaction server when accessing the JavaSpaces server.

3. In **Comergent.xml**, set the cronRefreshTime property to the interval, in seconds, at which a node will poll the Knowledgebase to check for modified or added cron jobs. Set cronRefreshTime to a negative number to prevent the node from periodically polling the Knowledgebase. The default value is -1.

4. Modify the security policy settings for the servlet container as follows:

   a. Copy the **policy.all** file to the directory in which the servlet container binaries are stored.

   b. Modify the command that starts the servlet container JVM by adding:

```
-Djava.security.policy=./policy.all
```

c. For WebLogic servers, add the following XML fragment to the **weblogic.xml** file:

```
<security-permission-spec>
    grant { permission java.security.AllPermission "", "";};
</security-permission-spec>
```

5. Optional: to test that the JavaSpaces server connection is working properly you can set the log level to VERBOSE, and the log flag to GlobalCache in the **Comergent.xml** properties file to get more information.

While the JavaSpaces services can be run on an application server with the Sterling Multi-Channel Selling Solution, it is better to run it on an independent, but highly available system. This prevents a single point of failure issue if the joint Sterling Multi-Channel Selling Solution/JavaSpaces server fails.

Index