

Sterling Multi-Channel Selling Suite®

SDK Guide

Version 3.5

Sterling Commerce
An IBM Company

Sterling Multi-Channel Selling Suite SDK Guide
Version 3.5

Edition

Copyright © 1998-2007.
Sterling Commerce, Inc.
ALL RIGHTS RESERVED

STERLING COMMERCE SOFTWARE

*****TRADE SECRET NOTICE*****

THE STERLING COMMERCE SOFTWARE DESCRIBED BY THIS DOCUMENTATION ("STERLING COMMERCE SOFTWARE") IS THE CONFIDENTIAL AND TRADE SECRET PROPERTY OF STERLING COMMERCE, INC., ITS AFFILIATED COMPANIES OR ITS OR THEIR LICENSORS, AND IS PROVIDED UNDER THE TERMS OF A LICENSE AGREEMENT. NO DUPLICATION OR DISCLOSURE WITHOUT PRIOR WRITTEN PERMISSION. RESTRICTED RIGHTS.

This documentation, the Sterling Commerce Software it describes, and the information and know-how they contain constitute the proprietary, confidential and valuable trade secret information of Sterling Commerce, Inc., its affiliated companies or its or their licensors, and may not be used for any unauthorized purpose, or disclosed to others without the prior written permission of the applicable Sterling Commerce entity. This documentation and the Sterling Commerce Software that it describes have been provided pursuant to a license agreement that contains prohibitions against and/or restrictions on their copying, modification and use. Duplication, in whole or in part, if and when permitted, shall bear this notice and the Sterling Commerce, Inc. copyright notice.

U.S. GOVERNMENT RESTRICTED RIGHTS. This documentation and the Sterling Commerce Software it describes are "commercial items" as defined in 48 C.F.R. 2.101. As and when provided to any agency or instrumentality of the U.S. Government or to a U.S. Government prime contractor or a subcontractor at any tier ("Government Licensee"), the terms and conditions of the customary Sterling Commerce commercial license agreement are imposed on Government Licensees per 48 C.F.R. 12.212 or § 227.7202 through § 227.7202-4, as applicable, or through 48 C.F.R. § 52.244-6.

These terms of use shall be governed by the laws of the State of Ohio, USA, without regard to its conflict of laws provisions. If you are accessing the Sterling Commerce Software under an executed agreement, then nothing in these terms and conditions supersedes or modifies the executed agreement.

Third Party Software and other Material

Portions of the Sterling Commerce Software may include or be distributed with or on the same storage media as products ("Third Party Software") offered by third parties ("Third Party Licensors"). Sterling Commerce Software may be distributed with or on the same storage media as Third Party Software covered by the following copyrights: Copyright (c) 1999-2005 The Apache Software Foundation. Copyright 2003-2007 CyberSource Corporation. Copyright (C) 2004-2006 Distributed Computing Laboratory, Emory University. Copyright (c) 1987-1997 Free Software Foundation, Inc., Java Port Copyright (c) 1998 by Aaron M. Renn. Copyright (C) 2000-2004 Jason Hunter & Brett McLaughlin. Copyright 1997-2004 JUnit.org. Copyright 2003-2007 Luck Consulting Pty Ltd. Copyright (c) 2005-2006 Mark James <http://www.famfamfam.com/lab/icons/silk/>. Copyright (c) 2002 Pat Niemeyer. Copyright (c) 1994-2006 Sun Microsystems, Inc. Copyright (c) 1996-2001 Ronald Tschalär. Copyright (c) Mark Wutka. All rights reserved by all listed parties.

Third Party Software which is distributed with or on the same storage media as the Sterling Commerce Software where use, duplication, or disclosure by the United States government or a government contractor or subcontractor, is provided with RESTRICTED RIGHTS under Title 48 CFR 2.101, 12.212, 52.227-19, 227.7201 through 227.7202-4, as applicable.

Additional information regarding certain Third Party Software is located at <installdir>\thirdpartylicenses

This product includes software developed by the Apache Software Foundation (<http://www.apache.org>). This product includes software developed by the JDOM Project (<http://www.jdom.org/>). This product includes software developed by Mark Wutka (<http://www.wutka.com/>). SUN, SOLARIS, JAVA, JINI, FORTE, and iPLANET trademarks and all SUN, SOLARIS, JAVA, JINI, FORTE, and iPLANET related trademarks, service marks, logos

and other brand designations are trademarks or registered trademarks of Sun Microsystems, Inc. All trademarks and logos are trademarks of their respective owners.

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, 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 v2.0, 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_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_2.0_license_OrderSelling.doc applies to the Apache 2.0 Software located at <installdir>\WEB-INF\lib\lucene-core-2.0.0.jar, <installdir>\WEB-INF\lib\lucene-demos-2.0.0.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

The Sterling Commerce Software is distributed with or on the same storage media as the BeanShell v1.2b7 (bsh-1.2b7.jar) software (Copyright (c) 2002 Pat Niemeyer) ("BeanShell Software"). The BeanShell Software is independent from and not linked or compiled with the Sterling Commerce Software. Sterling Commerce has not

made any modifications to the BeanShell Software. The BeanShell Software is free software which can be distributed and/or modified under the terms of the Sun Public License Version 1.0 as published by Sun Microsystems, Inc.

A copy of the Sun Public License is provided at <installdir>\thirdpartylicenses\beanshell_license_OrderSelling.doc. This license only applies to the BeanShell Software located at <installdir>\WEB-INF\lib\bsh-1.2b7.jar and does not apply to the Sterling Commerce Software, or any other Third Party Software.

The BeanShell Software is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the license for the specific language governing rights and limitations under the license. The Original Code is BeanShell. The Initial Developer of the Original Code is Pat Niemeyer. Portions created by Pat Niemeyer are Copyright (C) 2002. All Rights Reserved. Contributor(s): None Known.

Sterling Commerce has not made any modifications to the BeanShell Software. Source code for the BeanShell Software is located at <http://www.beanshell.org>

THE BEANSHELL SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, WARRANTIES THAT THE BEANSHELL SOFTWARE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT.

CYBERSOURCE SOFTWARE

The Sterling Commerce Software is distributed with or on the same storage media as the CyberSource Simple Order API v5.0.2 software (or components thereof) (Copyright 2003-2007 CyberSource Corporation) ("Cybersource Software"). Cybersource Software is free software which is distributed under the terms of the Apache License Version 2.0. A copy of the License Version 2.0 is found at <installdir>\thirdpartylicenses\Cybersource_v5.02_license_OrderSelling.doc and only applies to the Cybersource Software found at <installdir>\WEB-INF\lib\cybsclients-5.0.2.jar, <installdir>\WEB-INF\lib\cybssecurity-5.0.2.jar

Unless otherwise stated in a specific directory, the Cybersource Software was not modified. Neither the Sterling Commerce Software, modifications, if any, to the Cybersource 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 Cybersource Software in the specified directory file(s) and does not apply to the Sterling Commerce Software or to any other Third Party Software. License Version 2.0 includes the following provision:

"Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License."

EHCCACHE SOFTWARE AND JINI SOFTWARE

The Sterling Commerce Software is distributed with or on the same storage media as the ehcache software (or components thereof) (Copyright 2003-2007 Luck Consulting Pty Ltd) (the "Ehcache Software") and Jini Technology Starter Kit v2.1 software (or components thereof, including including jini-core.jar and jini-ext.jar) (Copyright 2005, Sun Microsystems, Inc.) ("Jini Software"). The Ehcache Software and Jini Software are 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 Ehcache Software and Jini Software, respectively, found in the specified directory files:

Ehcache Software - <installdir>\thirdpartylicenses\ehcache_1.2.4_license_OrderSelling.doc applies to the Ehcache Software located <installdir>\WEB-INF\lib\ehcache-1.2.4.jar.

Jini Software - <installdir>\thirdpartylicenses\Jini_2.1_license_OrderSelling.doc applies to the Jini Software located at <installdir>\WEB-INF\lib\jini-core-2.1.jar, <installdir>\WEB-INF\lib\jini-ext-2.1.jar.

Unless otherwise stated in the specific directory, the Ehcache Software and Jini Software were not modified. Neither the Sterling Commerce Software, modifications, if any, to Ehcache Software or the Jini 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 Ehcache Software and Jini Software which is the subject of the specific directory file and does not apply

to the Sterling Commerce Software or to any other Third Party Software. License Version 2.0 includes the following provision:

"Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License."

GETOPT SOFTWARE AND HTTPCLIENT SOFTWARE

The Sterling Commerce Software is distributed with or on the same storage media as the Getopt v1.0.12 software (or components thereof) (Copyright (c) 1987-1997 Free Software Foundation, Inc., Java Port Copyright (c) 1998 by Aaron M. Renn (arenn@urbanophile.com)) ("Getopt Software") and the HttpClient version 0.3.2 software (or components thereof) (Copyright (c) 1996-2001 Ronald Tschalär) ("HttpClient Software"). The Getopt Software and HttpClient Software are independent from and not linked or compiled with the Sterling Commerce Software. The Getopt Software and HttpClient Software are free software products which can be distributed and/or modified under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either, with respect to the Getopt Software, version 2 of the License or any later version, or, with respect to the HttpClient Software, version 2 of the License or any later version.

A copy of the GNU Lesser General Public License is provided at
<installdir>\thirdpartylicenses\Getopt_1.0.12_license_OrderSelling.doc,
<installdir>\thirdpartylicenses\HttpClient_0.3.2_license_OrderSelling.doc

This license only applies to the Getopt Software located at <installdir>\WEB-INF\lib\getopt-1.0.12.jar and HttpClient Software located at <installdir>\WEB-INF\lib\HTTPClient-0.3.2.jar, and does not apply to the Sterling Commerce Software, or any other Third Party Software.

Source code for the Getopt Software is located at <http://www.urbanophile.com>

Source code the HttpClient Software is located at <http://www.innovation.ch>

The Getopt Software and HttpClient Software are distributed WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

JUNIT SOFTWARE

The Sterling Commerce Software is distributed on the same storage media as the JUnit Software (or components thereof) (Copyright 1997-2004 JUnit.org.) ("JUnit Software"). Sterling Commerce has not made any additions or changes to the JUnit Software. The Sterling Commerce Software is not a derivative work of the JUnit Software. The Sterling Commerce Software is not a Contribution as defined in the Common Public License - v 1.0.

The source code for the JUnit Software is available at
http://sourceforge.net/project/downloading.php?groupname=junit&filename=junit3.8.1.zip&use_mirror=superb-east

The source code is available from Sterling Commerce under the Common Public License - v 1.0. Contact Sterling Commerce Customer Support in the event that the source code for the JUnit Software is no longer available at the respective, above-listed sites. A copy of the Common Public License - v 1.0 is provided at
<installdir>\thirdpartylicenses\JUnit_3.8.1_license_OrderSelling.doc. This license applies only to the JUnit Software located at <installdir>\WEB-INF\lib\junit-3.8.1.jar and does not apply to the Sterling Commerce Software or any other Third Party Licensor Software.

SUN MICROSYSTEMS

The Sterling Commerce Software is distributed with or on the same storage media as certain redistributable portions of the following software products: Sun JavaBeans™ Activation Framework ("JAF") (activation.jar) version 1.1, Sun JavaHelp version 2.0 ("JavaHelp"), and Sun JavaMail version 1.4 (mail.jar) (collectively, "Sun Software"). Sun Software is free software which is distributed under the terms of the specific Sun Microsystems, Inc. license agreement for each individual Sun products. A copy of the specific Sun Microsystems, Inc. license agreement relating to the Sun Software are found in the following locations and apply only to the individual pieces of the Sun Software located in the specified directory file(s):

SUN JAF - The specific Sun Microsystems, Inc. license agreement located at <installdir>\thirdpartylicenses\Sun_activation_jar_JAF_1.1_license_OrderSelling.doc applies to the Sun Software located at <installdir>\WEB-INF\lib\activation-1.1.jar.

SUN JavaHelp - The specific Sun Microsystems, Inc. license agreement located at <installdir>\thirdpartylicenses\JavaHelp_2.0_license_OrderSelling.doc applies to the Sun Software located at <installdir>\WEB-INF\lib\javahelp-2_0_02.jar

SUN JavaMail - The specific Sun Microsystems, Inc. license agreement located at <installdir>\thirdpartylicenses\Sun_JavaMail_1.4_license_OrderSelling.doc applies to the Sun Software located at <installdir>\WEB-INF\lib\mail-1.4.jar

Such licenses only apply to the Sun Software located in the specified the specified directory file(s) and does not apply to the Sterling Commerce Software or to any other Third Party Software.

WARRANTY DISCLAIMER

This documentation and the Sterling Commerce Software which it describes are licensed either "AS IS" or with a limited warranty, as set forth in the Sterling Commerce license agreement. Other than any limited warranties provided, NO OTHER WARRANTY IS EXPRESSED AND NONE SHALL BE IMPLIED, INCLUDING THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR USE OR FOR A PARTICULAR PURPOSE. The applicable Sterling Commerce entity reserves the right to revise this publication from time to time and to make changes in the content hereof without the obligation to notify any person or entity of such revisions or changes.

The Third Party Software is provided "AS IS" WITHOUT ANY WARRANTY AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. FURTHER, IF YOU ARE LOCATED OR ACCESSING THIS SOFTWARE IN THE UNITED STATES, ANY EXPRESS OR IMPLIED WARRANTY REGARDING TITLE OR NON-INFRINGEMENT ARE DISCLAIMED.

Without limiting the foregoing, the BeanShell Software, GetOpt Software, HttpClient Software, and JUnit Software, are all distributed WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Sterling Commerce, Inc.
4600 Lakehurst Court Dublin, OH 43016-2000 *
614/793-7000

Preface

Welcome to the Sterling Multi-Channel Selling Suite. This *SDK Guide* and the associated documentation provides all the information you need to implement the Sterling Multi-Channel Selling Suite at your enterprise.

Purpose

This guide provides an overview to extending current Sterling Commerce applications and developing new applications for the Sterling Multi-Channel Selling Suite. It presents a description of the system architecture, the main Java classes, and a description of the Sterling Multi-Channel Selling Suite Software Development Kit (SDK).

Audience

You should have an advanced level of information systems knowledge, familiarity with basic network and database concepts, Java (including the J2EE specification) and XML. Readers must have a firm understanding of developing Web applications in Java.

Conventions

Throughout this guide, we will use the conventions shown in the following table:

TABLE 1. Conventions

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

Contents

<i>STERLING COMMERCE SOFTWARE</i>	<i>ii</i>
<i>Third Party Software and other Material</i>	<i>ii</i>
<i>THE APACHE SOFTWARE FOUNDATION SOFTWARE</i>	<i>iii</i>
<i>BEANSHELL SOFTWARE</i>	<i>iii</i>
<i>CYBERSOURCE SOFTWARE</i>	<i>iv</i>
<i>EHCACHE SOFTWARE AND JINI SOFTWARE</i>	<i>iv</i>
<i>GETOPT SOFTWARE AND HTTPCLIENT SOFTWARE</i>	<i>v</i>
<i>JUNIT SOFTWARE</i>	<i>v</i>
<i>SUN MICROSYSTEMS</i>	<i>v</i>
<i>WARRANTY DISCLAIMER</i>	<i>vi</i>

CHAPTER 1 Introduction 1

Overview	1
<i>Releases, Projects, and Builds</i>	3
<i>Building and Merging</i>	4
<i>Building and Deploying</i>	6
<i>Upgrading Releases and Projects</i>	7
<i>SDK Properties</i>	8
Installing the SDK	9
<i>UNIX Steps</i>	10
<i>Test Installation</i>	10

Managing Your Project	11
<i>Setting up the SDK</i>	11
<i>Creating Your Customizations</i>	15
<i>Customizing Configuration Files</i>	16
<i>Building the Customized Web Application</i>	19
<i>Deploying the Web Application</i>	19
<i>Managing Different Releases</i>	19
<i>Upgrading Projects</i>	19
Building and Installing Patches.....	20
Merge Tools	21
<i>diff3</i>	21
<i>sibermerge</i>	21
Directory Organization.....	22
Release Documentation.....	24
 CHAPTER 2 Target Reference	25
List of Targets	25
<i>Project Management Targets</i>	26
<i>Project Migration Targets</i>	39
<i>Servlet Specification Targets</i>	40
 CHAPTER 3 Eclipse Plugin	43
Overview	43
Installation.....	44
Using the Plugin.....	45
<i>Building Projects Using the Plugin</i>	46
 Index	47

You can use the Software Development Kit (SDK) to install and customize your implementation of the Sterling Multi-Channel Selling Suite. This guide covers:

- “Overview” on page 1
- “Installing the SDK” on page 9
- “Managing Your Project” on page 11
- “Merge Tools” on page 21
- “Directory Organization” on page 22
- “List of Targets” on page 25

This guide covers Version 3.5 of the SDK. SDK 3.5 supports the following releases:

- Release 6.x with JDK 1.4
- Release 7.x with JDK 1.4
- Release 8.x with JDK 1.5

Overview

The SDK provides:

- A framework of Ant targets that enable you to install the Sterling Multi-Channel Selling Suite and manage your customizations to the system. Customizations can be as simple as setting the connection information for the Knowledgebase database server or as complex as changing the look-and-feel of the end-user pages or adding new functionality to the Sterling Multi-Channel Selling Suite.
- A comprehensive suite of documentation that includes an HTML version of this guide, a complete set of the product documentation, a tutorial designed to walk you through using the SDK, Javadoc for the public APIs supported, and an integrated SDK index to the complete set of message types, JSP pages, and data objects used by the Sterling Multi-Channel Selling Suite.

Use the SDK to install and customize your implementation of the Sterling Multi-Channel Selling Suite. The following diagram summarizes the process of creating a customized implementation of the Sterling Multi-Channel Selling Suite:

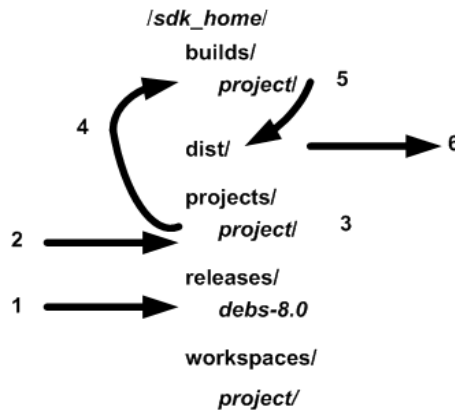


FIGURE 1. SDK Process to Create Customized Implementations

The steps are as follows:

1. Install a release of the Sterling Multi-Channel Selling Suite into the *sdk_home/releases/* directory using the **install** target.
2. Create a project for your implementation using the **newproject** target. A project directory is created: *sdk_home/projects/project/* and a copy of the release is copied to the *sdk_home/builds/project/* directory.

3. Set configuration properties and create customizations in the ***sdk_home/projects/project/*** directory.
4. Merge your customizations into your project using the **merge** target. The **merge** target builds your customizations over to the ***sdk_home/builds/project/*** directory. At the same time, the data bean classes are generated and the configuration properties merged in to the configuration files.
5. Create a new version of the **Sterling.war** Web application file using the **dist** target. The **Sterling.war** file is created in the ***sdk_home/dist/*** directory.
6. Deploy the new **Sterling.war** file to your servlet container.

Releases, Projects, and Builds

Releases

Each release of the Sterling Multi-Channel Selling Suite can be installed into the SDK under the ***sdk_home/releases/*** directory. For example, Release 8.0 is installed into ***sdk_home/releases/debs-8.0/***. You install a release by running the **sdk install** target.

- For Release 6.3.1 and earlier, the application and product documentation are installed together using one execution of the **install** target.
- For Release 6.4.1 and later, the application and product documentation are installed by executing the **install** target twice: once to install the application and once to install the documentation.

Projects

When you work on an implementation of the Sterling Multi-Channel Selling Suite, you are essentially working on a customization of a release of the Sterling Multi-Channel Selling Suite. Your customization might be as simple as setting the database connection information specific to your implementation, or it might be as complex as adding a new functional component to the system. In either case, the changes you want to make are maintained in a *project*: the set of files and directories you maintain to capture all the changes you want to make to the release of the Sterling Multi-Channel Selling Suite.

Each project consists of a set of directories and files in a project directory in ***sdk_home/projects/***. All the changes that you need to make to the Sterling Multi-Channel Selling Suite must be made here: do not make changes in the ***sdk_home/releases/*** directory. You can make the following types of customizations:

- configuration parameters: using the properties files, set the relevant values for properties such as database type and connection information, logging levels, and so on.
- changes to existing files: make changes to JSP pages and controllers by copying the file from the release to your project and changing the file in the project location. Use the **customize** target to copy the files from their location in the release to the corresponding location in the project directory.
- new files: you can create new data objects, new Java classes by creating the corresponding files in the appropriate locations under the project directory.

Builds

After you finish making your customizations in the project directory, build your customized release of the Sterling Multi-Channel Selling Suite by merging the customizations with the release on which the customizations are made. When you build your implementation, it is created under the *sdk_home/builds/* directory. See “Building and Merging” on page 4 for what happens during the build process.

You can build your project as many times as you like, but each build overwrites the previous one. Typically, as you build your project, you can watch to see that the data objects are generated and compiled successfully and that other Java classes that you have created compile successfully. The build terminates if a compilation error occurs.

Building and Merging

The SDK provides a framework within which you can maintain different releases of the Sterling Multi-Channel Selling Suite and different customization projects. At any one time, the SDK has one active project and environment, and the *debs.version* property in the corresponding *project_env.properties* file determines the active release for the project.

Environments

When you work on a project, you may need to maintain different versions of the configuration parameters so that you can deploy the customized implementation into different locations. For example, you may need to work with a local development installation first, then test the customizations in a QA environment, before moving the customized implementation into the production environment. You may want to connect to different databases for these deployments or set different logging levels.

To help this process, we provide the ability to work with different environments in the same project: dev, local, qa, and prod. At any one time, the `deploy.environment` property of the **sdk-settings.properties** file indicates which one is active. When you run the merge target, the properties from the corresponding **project_env.properties** file are merged with the configuration files, and these are the ones set in the configuration files you find in the *sdk_home/builds/project/* directories.

Building Process

When you build your implementation, the files from the active project are merged with the active release as follows:

1. When you first create the project using the **newproject** target, a new directory is created in the *sdk_home/builds/* directory: this is essentially a copy of the active release. A project directory for the new project is created under the *sdk_home/projects/* directory.
2. To start customizing your project, install the appropriate database connection files: use the **installDB2**, **installODBC** (Release 6.x-7.0), **installMSSQLJDBC** (Release 7.1.1 and later), or **installOracle** target to do this.
3. You must specify the database type using the **env.setDBType** target.
4. You create customizations in the project directory, *sdk_home/projects/project/*. The `customize` target helps you to move files from the release to the project directory.
5. After you have created your customizations in your project directory, you run the merge target. This merges the changes from your project directory by copying the new files from the project directory into the build directory. In the process, newer versions of files from the *sdk_home/projects/project/* directory overwrite existing files that were copied from the *sdk_home/releases/* directory in Step 1, so that changes made in your project overwrite the original release file in the build directory, not in the releases directory.
 - a. In Release 6.0 through Release 6.7, properties defined in the **project_env.properties** files are used to substitute placeholder tokens in configuration files. For each property in a configuration file, if a token value is found, such as `@SMTP_SENDER@`, then if a corresponding value is specified in the properties file, it is replaced with the specified value:

```
SMTP_SENDER=sales@matrix.com
```

- b. In Release 7.0 and later releases, properties defined in the ***project_env.properties*** files are used to populate the ***prefs.xml*** configuration files.
6. Once you run the **merge** target, you can run the schema creation and data loading targets: **createDB** and **loadDB** respectively. These targets use the database connection information that was copied from the project properties file to the script and schema files in the ***sdk_home/workspaces/project/*** directories.
7. Run the **dist** target. This creates a new Web application file that is ready to deploy to the servlet container.
8. By running the **switchenv** target, you can change the active environment used to build the project: this enables you to set different values of the configuration parameters; for example, to point to a different database.

Building and Deploying

Keep in mind that when you are working in the SDK, you are building your customized implementation of the Sterling Multi-Channel Selling Suite. Once you have completed the customization process, you deploy the resulting **Sterling.war** Web application file to the servlet container. In general, the machine(s) on which you run the SDK can be completely separate from the machine(s) on which you run the Sterling Multi-Channel Selling Suite.

TABLE 2. SDK and Deployment

	SDK	Servlet Container
Machine	Usually your desktop machine	Usually a server-strength machine
Software requirements	You must have a JDK installed	You must have a servlet container installed
Database connections	Must be able to connect to the Knowledgebase to create schema and load data	Must be able to connect to the Knowledgebase to run the Sterling Multi-Channel Selling Suite
Purpose	To build your implementation of the Sterling Multi-Channel Selling Suite	To run the Sterling Multi-Channel Selling Suite Web application

In general, each servlet container provides a specific deployment mechanism and we suggest that you use this to deploy your customized Web application from the

SDK to the servlet container. If you use the servlet container deployment tool, then this looks after any configuration or registration steps that the servlet container must perform to use the Web application. You should also consult the *Sterling Multi-Channel Selling Suite Implementation Guide* for any specific steps required to deploy the Sterling Multi-Channel Selling Suite Web application.

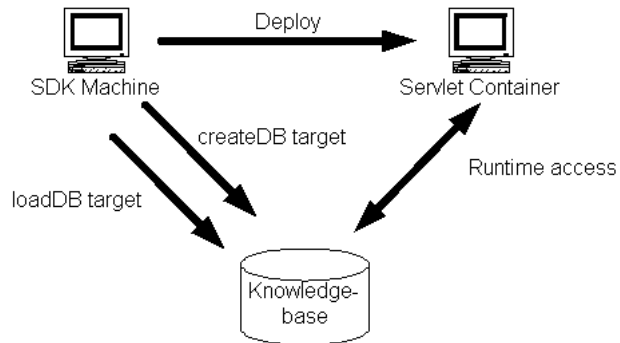


FIGURE 2. Logical Separation of Machines

Usually, the database server that holds the Knowledgebase is a separate machine from both the machine on which you run the SDK and the machine running the servlet container. By running the **createDB** and the **loadDB** (or **loadMatrixDB**) targets, you can create and populate the Knowledgebase schema from the SDK machine. When you run the **merge** target, the database connection information is set in the data source configuration files under the *sdk_home/builds/project/* directory. When you deploy the Web application to the servlet container, then the database connection information is used to provide runtime access between the Sterling Multi-Channel Selling Suite and the database server.

Deploy Targets

If you run the SDK and servlet container on the same machine, then it is possible to deploy the customized Web application using SDK targets. However, these targets are shortcuts useful for early testing: they should probably not be used to perform the final deployment of the implementation to your QA and production servlet containers. See “List of Targets” on page 25 for more information about the deploy targets.

Upgrading Releases and Projects

The SDK is designed to support you through the process of implementing the Sterling Multi-Channel Selling Suite at your site and to provide a means to

maintain your customizations as you update the implementation over time. If a new release of the Sterling Multi-Channel Selling Suite is produced, then you may need or want to upgrade to the new release either to add additional functionality or to enhance existing functionality.

The SDK supports this process by enabling you to use the SDK to upgrade from one release (the source) to another (the target) and then to upgrade your project from the source release to the target release. In this process, the SDK helps you to identify what changes have been made in the new release and how they intersect with customizations that you have made.

The following files are created as part of the upgrade process:

- **conflictingfiles.log**: this lists all files that exist in the target release and the project, but not in the source release. It also lists Java classes whose source file exists in the project, and whose class file is in the target release.
- **filestobedeleted.log**: this lists all files that exist in the project and the source release, but which have been deleted in the target release.
- **filestobeupgraded.log**: this lists all files that exist in both the source and target releases and in the project.

SDK Properties

When you run SDK targets, the targets use properties that you have specified in the properties files. Bear in mind that properties can be specified in one or more files and so you must understand the order in which the properties files are read.

When you run an Ant target, the property files are read by the target in the following order. Once a property is read in from one property file, it is ignored in any files that are read after that. Consequently, the file in which you define a property is important: if the same property is defined in a file that is read before the intended file, then your intended value will not be used.

1. **sdk-settings.properties**: the SDK uses this file to track what environment, project, and release are currently in use. You should not manually edit this file: use the corresponding targets: **switchenv**, **switchproject**, and **switchdebs** respectively.
2. **project_env.properties**: use this file to set properties specific to the project and the stage of the project
3. **project.properties**: use this file for properties that are common to all stages of your project.

4. **my_sdk.properties**: use this file for properties that you use in all your SDK projects. This file is not overwritten when you perform an upgrade to your SDK and so you can use it to preserve properties that would otherwise be lost if you put them in **sdk-settings.properties** or **default_sdk.properties**. This file is created when you run the **setup** target.
5. **default_sdk.properties**: sets default values for anything not set in higher precedent files.

Note that leaving an entry blank is interpreted by Ant as an empty string, and not that the property is not defined. Thus the following will cause the `container.name` property to be set to "" in any SDK target that is run:

```
container.name=
```

Installing the SDK

To install the SDK, identify a directory on your machine to use as the development directory: we refer to this as *sdk_home*.

Attention: If you have an installation of an earlier version of the SDK on your machine, and want to install the current version, you must install the current version into a <i>different</i> directory. .
--

The SDK is delivered as a JAR file, **sdk-framework.jar**. Simply unjar the JAR file in your *sdk_home* directory.

Attention: You must use the SDK with JDK 1.4 (Releases 6.x and 7.x) or with JDK 1.5 (Releases 8.x and later). Ensure that your <code>JAVA_HOME</code> variables is set to point to the appropriate JDK home on your machine.

The SDK framework is updated regularly. Contact your Sterling Commerce representative to obtain updates. Releases of the SDK are managed on a separate timetable from releases of the Sterling Multi-Channel Selling Suite. Check whether your version of the SDK supports the Sterling Multi-Channel Selling Suite release that you are working with.

The Software Development Kit is designed to run a collection of Ant targets. Before you run these targets, make sure that you set the `PATH` and `JAVA_HOME` environment variables to point to the location of your installation of the Java Development Kit. For example:

```
set JAVA_HOME=C:/jdk-1.5
set PATH=%PATH%;%JAVA_HOME%\bin
```

(on a Windows platform)

```
setenv JAVA_HOME /usr/java1.5
setenv PATH=$PATH:$JAVA_HOME/bin
```

(on a UNIX platform)

UNIX Steps

You may need to change the permissions set on the SDK files as follows:

1. At the command line navigate to the *sdk_home* directory.
2. Enter:

```
chmod +x *sh
```

This makes all script files in the *sdk_home* directory executable by all users.

3. Note that the UNIX batch file is provided as **sdk.sh**. You can rename it to **sdk** (or create an alias called `sdk: alias sdk='./sdk.sh'`) or run the targets below by substituting “`sdk.sh`” for “`sdk`”.

Test Installation

You can test that you have installed the SDK correctly as follows:

1. At the command line, navigate to *sdk_home*.
2. Enter:

```
sdk version
```

You should see output similar to the following:

```
Buildfile: C:\sdk_home\sdk.xml
displaySDKInfo:
=====Configuration Info=====
SDK Environment:
  Project Name: ..... matrix80
  Default Comergent Version: ..... debs-8.0
  Environment: ..... dev
Directory Information:
  Comergent SDK Home..... C:\sdk_home
  Default language..... en
  Default Country..... US
displayProjectInfo:
===== Project Info=====
Project Environment:
  Application Name ..... Comergent
```

```
Comergent Version: ..... debs-8.0
Java Version: ..... 1.5.0_08
Directory Information:
Project ..... C:\sdk_home/
projects/matrix80
Build ..... C:\sdk_home/builds/
matrix80
Dist ..... C:\sdk_home/dist
container-chooser:
displayContainerInfo:
default-displayContainerInfo:
===== Container Info=====
Container Name ..... tomcat
Container Version: ..... 5.5
prepare:
init:
version:
Comergent Technologies Inc. Software Development Kit 8
SDK Framework version : 3.5(RTQA0a)-build624
BUILD SUCCESSFUL
Total time: 4 seconds
```

3. Enter:

```
sdk setup
```

This creates the **my_sdk.properties** file. Use this file to manage SDK properties that are specific to your installation of the SDK.

Managing Your Project

In general, your development of a deployment project follows these phases:

- “Setting up the SDK” on page 11
- “Creating Your Customizations” on page 15
- “Building the Customized Web Application” on page 19
- “Deploying the Web Application” on page 19

Setting up the SDK

To set up the SDK for your project, you must perform the following steps:

1. Decide on a project label for this project: typically this is a short, unique identifier for the deployment. For example, if you are modifying the Matrix reference application, you might choose “matrix80” as the project label. We refer to this project label as *project*.

2. Check whether the release on which you want to build your project is installed already in the SDK.

- a. If the release is installed, run the **switchdebs** target to set the current default release to the desired release. For example:

```
sdk switchdebs debs-8.0
```

Skip to Step 4.

- b. If the release is not installed, copy the **Sterling.war** Web application file and content JAR file for the release to a temporary location on your machine.
3. Run the **install** target:

- a. For releases later than Release 6.3, using SDK Archive files:

```
sdk install <Location of Comergent.war file>  
sdk install <Location of documentation JAR file>
```

This creates a sub-directory for the release under **sdk_home/releases/** and extracts all of the Web application files from the specified **Comergent.jar** file. For example:

```
skd install C:\temp\Comergent-7.0.1-def-7_0_1-RC2.jar  
sdk install C:\temp\doc-7.0.1-def-7_0_1-RC2.jar
```

The second execution of the install target installs the product documentation (SDK index docs, Javadocs, ERD diagrams, and PDF Guides).

- b. For releases prior to Release 6.3.1, to populate the release directory with the files for this release of the Sterling Multi-Channel Selling Suite:

```
sdk install <release> <Location of Comergent.war file>  
<Location of Content.jar file>
```

Specify exactly the correct form of the *release* string. This creates a sub-directory for the release under **sdk_home/releases/** and extracts all of the Web application files from the specified **Comergent.war** file. For example:

```
sdk install debs-6.3.1 C:\temp\Comergent.war  
c:\temp\sdk-content-debs-6.3.1.jar
```

4. Run the **newproject** target to create directories for this project. Enter:

```
sdk newproject <project>
```

This creates a project directory called **sdk_home/projects/project/**. It also creates a copy of the release in **sdk_home/builds/project/**.

5. Install the database files:

- a. If you are running against a SQL Server database server, run the **installMSSQLJDBC** target:

```
sdk installMSSQLJDBC <full path to JAR file>
```

You must specify the location of the JAR or ZIP file that provides the JDBC drivers for your SQL Server database server. For example:

```
sdk installMSSQLJDBC C:\MSSQLJDBC\sqljdbc_1.1\enu\sqljdbc.jar
```

- b. If you are running against an Oracle database server, run the **installOracle** target:

```
sdk installOracle <full path to JAR file>
```

You must specify the location of the JAR or ZIP file that provides the JDBC drivers for your Oracle database server. For example:

```
sdk installOracle C:\oracle\ora90\jdbc\lib\classes12.zip
```

- c. If you are running against a DB2 database server, run the **installDB2** target:

```
sdk installDB2 <full path to JAR file>
```

You must specify the location of the JAR or ZIP file that provides the JDBC drivers for your Oracle database server. For example:

```
sdk installDB2 C:\IBM\SQLLIB\java\db2jcc.jar
```

- d. (Release 6.4.1 to Release 7.1) If you are running against an SQL Server database server, then run the **installODBC** target:

```
sdk installODBC
```

- e. (Release 6.0 to Release 6.3.1) If you are running against an SQL Server database server, then run the **installMsSql** target:

```
sdk installMsSql
```

6. Run the **env.setDBType** target to specify the database type for this project with one of the following:

```
sdk env.setDBType Oracle
```

```
sdk env.setDBType MSSQLJDBC
```

```
sdk env.setDBType DB2
```

```
(Releases earlier than 7.1) sdk env.setDBType ODBC
```

7. In the project directory, modify the **project_env.properties** files to ensure that the correct values are set for the project properties, particularly those of the database connections. In general, you must set values for the machine name of the database server and the username and password you want to use to create

the database schema in the database server. For Oracle, you must also provide the TNS alias used to connect to the database from the SDK machine. This is the ORACLE_DATABASE property in the properties file.

Note that you will probably use different database connections for your development (dev), QA (qa), and production (prod) environments. (See “Environments” on page 4 for information about managing project environments.) The deploy.environment property of the *sdk_home/sdk-settings.properties* file determines which environment is active at any moment. For example, if the *sdk_home/sdk-settings.properties* file reads:

```
project.name=matrix
deploy.environment=dev
```

then the *sdk_home/projects/matrix/matrix_dev.properties* file determines the database connection information.

8. If you are installing your Web application into a Servlet Specification 2.2-compliant servlet container, then run the setservletspec2.2 target:

```
sdk setservletspec2.2
```

Note: The **setservletspec2** target works acts on the *current* project. If you start a new project, you must run this target again to deploy the new project into a Servlet Specification 2.2-compliant servlet container.

The *Sterling Multi-Channel Selling Suite Implementation Guide* provides a table of Servlet Specifications are supported for each servlet container.

9. Run the **merge** target: this copies the release to the *sdk_home/builds/project/* directory. It merges the project properties that you set in Step 7 into the build.
10. Run the **createDB** target. This creates the database schema (tables, views, procedures, and so on) in the database specified by the connection properties set in Step 7. Log files are created as the schema creation scripts run: you should check these to ensure that no errors have occurred. Typically, errors can occur if you do not run merge before running this target or if you have made an error in entering the properties. Note that if you are running against a DB2 server, then you must open up a DB2 command window first using the db2cmd command, and then run the createDB target in this window.
11. Run the **loadDB** target or the **loadMatrixDB** target: the former populates the Knowledgebase with the minimal data for the Sterling Multi-Channel Selling Suite; the latter populates the Knowledgebase with the full Matrix reference data.
12. Run the sdk **dist** target. This creates a new **Sterling.war** file that you can now deploy to your servlet container. When you deploy it, the Web application is

already configured to connect to the right database and the Knowledgebase is already populated with either the minimal or reference data, so that you can log in to the Sterling Multi-Channel Selling Suite and start testing it.

Creating Your Customizations

Typically, when you customize the Sterling Multi-Channel Selling Suite, you make modifications in these main areas:

- the Java classes that comprise the business logic of the system
- the JSP pages used to present content to users
- configuration files
- the XML schema definitions of the set of data objects
- the database schema to reflect the changes made in the XML schema

You can use the `sdk customize` target to select files from the *sdk_home/releases/release/* source directories for customization. Enter:

```
sdk customize <filename>
```

The file is copied from the release directory identified by the `debs.version` property to the corresponding *sdk_home/projects/project/* directory. For example:

```
sdk customize AdvisorMessageTypes.xml
```

copies the file *sdk_home/releases/release/WEB-INF/properties/AdvisorMessageTypes.xml* to *sdk_home/projects/project/WEB-INF/properties/AdvisorMessageTypes.xml*.

The structure of the project sub-directories is modeled on the structure of the Web application which is based on the J2EE specification as described in the *Sterling Multi-Channel Selling Suite Developer Guide*. Additional project directories are provided for the source files (**src/**) and the configuration file templates (**templates/**). Note that you do not have to include “src”, “web”, “WEB-INF” in the file names; these are included automatically.

- Java classes: in general, you should not need to modify any of the Java classes that comprise the Sterling Multi-Channel Selling Suite application delivered in the **Sterling.war** file. Your goal should be to customize the business logic of the system by extending classes. The sub-classes should

follow a consistent naming convention that reflects the parent's class package and name.

For example, to modify the way that the `com.comergent.apps.catalog.advisor.AdvisorProductDetailController` class works, create a new class in the same package structure called `MatrixAdvisorProductDetailController`. This class should extend the `AdvisorProductDetailController` class and you can overwrite any methods described in the SDK Javadoc documentation.

Maintain these class files in ***sdk_home/projects/project/src/com/*** so the changes can be merged with the standard Sterling Multi-Channel Selling Suite class hierarchy.

- JSP pages: You can customize existing JSP pages or create new ones. Simply follow the guidelines described in the *Sterling Multi-Channel Selling Suite Developer Guide*.
- Configuration files: Typically, you will customize configuration files either to modify the logical flow of the Sterling Multi-Channel Selling Suite using the **MessageTypes.xml** files, or to add parameters required by the customized business logic. These files should be maintained in ***sdk_home/projects/project/WEB-INF/properties/*** so the changes can be merged with the standard Sterling Multi-Channel Selling Suite configuration files.
- Data objects: New data objects created by your implementation or changes to existing data objects must be maintained in your ***sdk_home/projects/project/WEB-INF/schema/*** directory so that they can be deployed along with the standard Sterling Multi-Channel Selling Suite data objects.

Customizing Configuration Files

Typically, you will make two sorts of changes to configuration files during the customization of your project:

- Tokenizing Properties
- Adding Properties

Tokenizing Properties

If you want to change a property so that it is tokenized, then you must customize the property file in which it is specified.

Using your text editor, edit the file so that the property is defined as a token. For example, this text declares the `listPriceDecimalPrecision` element as a tokenized property in the **Comergent.xml** file:

```
<listPriceDecimalPrecision controlType="text"
runtimeDisplayed="true" ChangeOnlyAtBootTime="false" visible="true"
boxsize="45" displayQuestion="Allowed Decimal Places for displaying
list prices" defaultChoice="2" help="Enter the number of decimal
places displayed for list prices.">@listPriceDecimalPrecision@</list-
PriceDecimalPrecision>
```

Now add a property to your **project_env.properties** file:

```
listPriceDecimalPrecision=2
```

When the project is built, the resulting **Comergent.xml** will include:

```
<listPriceDecimalPrecision controlType="text"
runtimeDisplayed="true" ChangeOnlyAtBootTime="false" visible="true"
boxsize="45" displayQuestion="Allowed Decimal Places for displaying
list prices" defaultChoice="2" help="Enter the number of decimal
places displayed for list prices.">2</listPriceDecimalPrecision>
```

Only files on the list of template files will be processed to have tokens replaced. If you want to add a token to a file that is not on the list, then you must add it:

1. Open **sdk_home/etc/templates/8.x/template_files.list** and add the file name to the list of files.
2. Save the file.
3. Enter:

```
sdk customize <File to be tokenized>
```

4. In the appropriate sub-directory of **sdk_home/projects/project/** locate this file, and insert the token using the standard syntax: **@TOKEN_NAME@**.
5. In the **project_env.properties**, add a line for this new token and its intended value.

```
TOKEN_NAME=Token value
```

6. Run:

```
sdk merge
```

7. The file in the **sdk_home/builds/project/** directory now has the token value in place of the token.

Adding Properties

If you want to add a property to a configuration file, then you must customize the configuration file in which it is specified. Enter:

```
sdk customize <Configuration file>
```

Using your text editor, edit the file so that the property is declared. For example, this text declares the `viewAllComments` element as a property in the **Comergent.xml** file:

```
<viewAllComments ChangeOnlyAtBootTime="false"
controlType="select" button="radio"
multipleChoice="false"
runtimeDisplayed="true"
visible="true" boxsize="45"
displayQuestion="Enable users to see all comments"
displayOptions="true,true,false,false"
defaultChoice="true" help="Allow users to see all comments made
about a product">false</viewAllComments>
```

When you next build your project, this property is included in the **Comergent.xml**.

If you want the property to be customizable as a token, then follow the instructions in “Tokenizing Properties” on page 16. For example:

```
<viewAllComments ChangeOnlyAtBootTime="false"
controlType="select" button="radio"
multipleChoice="false"
runtimeDisplayed="true"
visible="true" boxsize="45"
displayQuestion="Enable users to see all comments"
displayOptions="true,true,false,false"
defaultChoice="true" help="Allow users to see all comments made
about a product">@ViewAllComments@</viewAllComments>
```

Now add a property to your *project_env.properties* file, for example:

```
ViewAllComments=true
```

When the project is built, the resulting **Comergent.xml** will include:

```
<viewAllComments ChangeOnlyAtBootTime="false"
controlType="select" button="radio"
multipleChoice="false"
runtimeDisplayed="true"
visible="true" boxsize="45"
displayQuestion="Enable users to see all comments"
displayOptions="true,true,false,false"
defaultChoice="true" help="Allow users to see all comments made
about a product">true</viewAllComments>
```

Building the Customized Web Application

Use the **merge** target to compile the classes and copy over and merge the customizations that you have created in the *sdk_home/projects/project/* directory. Check that the deployment source includes all of the files and modifications to files that you expect.

```
sdk merge
```

Deploying the Web Application

Use the **distWar** target to generate the Web application. The Web application is copied and compiled into the *sdk_home/dist/* directory, and from there can be deployed to your servlet container.

```
sdk distWar
```

Note that the automatically-generated name of the WAR file includes a date stamp and the value of the `deploy.environment` property defined in the **sdk-settns.properties** file. The first part of the file name is controlled by the `app.name` property defined in the **default-sdk.properties** file. For example, **Sterling_20071211_dev.war**.

Alternatively, you can use the **dist** target: this creates a full install JAR file. Use this when you need to pass the SQL database schema creation scripts to a database administrator.

You can also use the **deploy** target to deploy the Web application to the servlet container environment.

```
sdk deploy
```

Note however, that you must take care to modify the SDK properties so that the application is installed into the correct location. Use the `container.home` and `apps.dir` properties defined in the **my-sdk.properties** file to specify the deployment location.

Managing Different Releases

You may find that you are working on different projects at the same time. If the different projects use different versions of the Sterling Multi-Channel Selling Suite, then use the **install** target to install the new release and the **switchdebs** target to switch from one installed release to another.

Upgrading Projects

If you build a project against one release of the Sterling Multi-Channel Selling Suite and then you install a new release, then you can upgrade the project to the

new release using the `project.premigrate`, `project.migratefiles`, and `project.postmigratecleanup` targets.

You should always run `sdk merge -clean` after upgrading a project to a new release.

Building and Installing Patches

The SDK provides a means to build patches to releases. Other users can use the SDK to patch their copies of the release using the patch.

To Create a Patch

1. Determine that you need to create a patch to a release, for example, Release 8.0.
2. If you have not already installed this release into your SDK, then do so using the **install** target.
3. Switch to this release using the **switchdebs** target.
4. Create a new project using the **newproject** target.
5. Working in this project, modify existing files using the `customize` target to copy them over to the project, and create new files as appropriate for the patch.
6. Use the **merge**, **dist**, and **deploy** targets to test that the patch works.
7. Create the patch JAR file by running the **project.createHotFix** target. A JAR file is created that contains the files required by the patch.

This patch file is now ready for distribution.

To Install a Patch

1. Determine that you need to apply a patch to a release, and obtain the patch as a JAR file from Sterling Commerce.
2. If you have not already installed this release into your SDK, then do so using the `install` target.
3. Copy the patch JAR file to a convenient temporary location on your SDK machine.
4. By running the `install` target, install the patch:

```
sdk install <Location of patch JAR file>
```

The patch is automatically unpacked into an **overlay/** directory under the release directory, for example: `sdk_home/releases/debs-8.0/overlay/`.

5. When you build a project using this release, then files from the **overlay/** directory overwrite files from the release WAR file as they are copied over to the *sdk_home/builds/project/* directory.

If you run the examine or customize targets for a file, then if the file exists in the patch, then that is the copy that is used, and not the original file in the release WAR file.

You can apply one or more patches to a release. However, if the same file is present in more than one patch, then the version of the file under the **overlay/** directory is the copy from the *most recently-applied* patch.

Merge Tools

When you run the **project.migratefiles** target, the SDK attempts to merge files using a merge tool. The merge tool that is used is configurable: its use is specified using the following properties in the **my_sdk.properties** file:

- `automerge.nonconflicting.files=true`: selects or deselects the auto-merge of non-conflicting changes
- `mergetool.name=diff3`: sets the merge tool to be used in the merge process

The merge tools are installed in *sdk_home/sdktools/mergetools/*. Each merge tool is installed in its own directory, and a wrapper is provided to call the merge tool. The file **handler.xml** is the wrapper: two wrappers are provided, one for sibermerge and the other for diff3 (which is freely distributed with the SDK). The directory name where the merge tool is installed must be the one used for the `mergetool.name` property value.

diff3

Make sure that the location of the diff3 executable is specified using the `diff3.executable` property.

sibermerge

Make sure that the JAR file containing the Java form of the sibermerge application is copied to the specified directory: *sdk_home/plugins/mergetools/sibermerge/* and make sure that the license key file is in the same directory.

Directory Organization

While you work with the SDK, you see under the *sdk_home* directory the following directories and files. Note that not all of these are created as part of the initial installation of the framework:

- **builds/**: contains the deployment source for the customized Web application. This directory is created the first time the *newproject* target is run. The content of each project is moved here prior to running the build scripts. It can contain one or more target projects, for example: **builds/matrix80/** and **builds/anderel/**.
- **dist/**: contains the built image of the Web application used to deploy to the servlet container.
- **docs/**: contains the product documentation and generated Javadoc for the Sterling Multi-Channel Selling Suite.
- **etc/**: contains the project templates and development tools used by the SDK. This includes a sub-directory **etc/lib/** that holds the various JAR files used by the tools.
- **projects/**: used to manage the source for SDK projects. This directory is created the first time the *newproject* target is run. Each project directory must have a unique name (such as *matrix2*). Typically, project directories each contain source directories, customized JSP and HTML pages for each project, and the project's WEB-INF directory and its configuration files. For example, to manage two projects, *matrix80* and *anderel*, you will use two sub-directories: **projects/matrix80/** and **projects/anderel/** to manage the customized content for each.

Each project directory contains the following properties files:

project.properties, ***project_local.properties***, and ***project_env.properties***.

These files specify the values of properties used by the SDK targets. The values of *env* are: *dev*, *prod*, and *qa*. When you run an SDK target, the *deploy.environment* property of the ***sdk_home/sdk-settings.properties*** file determines which properties file is used to set the SDK properties. See “Environments” on page 4 for more information.

- **releases/**: contains the Java classes, Java libraries, JSP pages, and configuration files for the Sterling Multi-Channel Selling Suite. This directory is created the first time the **install** target is run. When you work with a particular release of the Sterling Multi-Channel Selling Suite, you run the **install** target in order to put the Sterling Multi-Channel Selling

Suite release in a directory under the **releases/** directory. For example, if you install Release 8.0 into the SDK, a directory is created called **sdk_home/releases/debs-8.0/**. The following sub-directories are created in each release directory:

- **examine/**: when you run the examine target, files are copied under this directory from the release WAR file or from the **overlay/** directory if they are present there.
- **image/**: this directory holds the WAR for the release.
- **overlay/**: this directory holds files installed through patches. When you run the **customize** or **examine** targets, files under this directory take precedence over corresponding files in the release WAR file and when the **merge** target is run to build projects.
- **tmp/**: a temporary workspace used by SDK targets. This directory is created the first time the **merge** target is run.
- **workspaces/**: contains the SQL scripts and XML data files that will be used to generate the database schema and load data into the database for each project.
- **ComergentCopyrightNotice.txt**
- **my_sdk.properties**: use this file to set any SDK properties specific to your environment. Properties set in this file take precedence over properties set in **sdk-settings.properties**. The **sdk-settings.properties** file can be overwritten during an upgrade to the SDK framework, whereas properties set in **my_sdk.properties** will be preserved.
- **sdk-settings.properties**: the main file used to configure the SDK when targets are run. It determines which project is current, which environment is current, and which release is to be used when a new project is created.
- **sdk.bat**: the main script file for Windows.
- **sdk.sh**: the main script file for UNIX.
- **sdk.xml**: the SDK configuration file with target definitions.
- **sdk-6.x.xml**: the SDK configuration file with target definitions for Release 6.x releases.
- **sdk-7.x.xml**: the SDK configuration file with target definitions for Release 7.x releases.

- **sdk-8.x.xml**: the SDK configuration file with target definitions for Release 8.x releases.

Release Documentation

A JAR file containing product documentation ships with each release of the Sterling Multi-Channel Selling Suite. The documentation comprises:

- Product documentation guides in the form of PDF files
- Index docs: a complete guide to the message types, controllers, JSP pages, and data objects, and database schema for the release
- Javadocs: a comprehensive set of Javadoc to the supported APIs for the release
- ERD diagrams: a graphical representation of the underlying database table schema for the release

This JAR file is installed into the SDK using the **install** target.

This chapter describes the supported targets provided by the SDK. Each target is an Ant target and uses the Ant syntax. See the Ant documentation available at <http://ant.apache.org> for further details.

Some target names are designed to reflect the scope of the target. For example `env.setDBType` sets the database type in the current environment, and `project.premigrate` performs an analysis of files in the current project.

You can create new or customized targets by modifying the **sdk-x.y[custom-example].xml** files in *sdk_home/etc/custom/*. These targets can be added to your SDK by as follows:

1. Edit the *sdk_home/etc/custom/custom-components-list.txt* file to add the string `custom-example` to the list of custom files.
2. At the command line, navigate to *sdk_home/etc/build/* directory.
3. Run the command:

```
fmwkbuilder customizeSDK
```

The new targets can now be used along with the existing targets.

List of Targets

This list divides the SDK targets into the following categories:

- “Project Management Targets” on page 26
- “Project Migration Targets” on page 39
- “Servlet Specification Targets” on page 40

Project Management Targets

The project management targets provided with the SDK include:

addMigrateData

Use this target to upgrade a project data from one release (the source) to another (the target). Check the *Sterling Multi-Channel Selling Suite Implementation Guide* to see if an upgrade is supported for your release. To run this target, enter:

```
sdk addMigrateData DEBS_m_to_DEBS_n
```

Where *m* is the release number of the source release and *n* is the target release. For example:

```
sdk addMigrateData DEBS_7_0_2_to_DEBS_7_1
```

addMigrateSegData

Use this target to upgrade segmentation project data from one release (the source) to another (the target). Check the *Sterling Multi-Channel Selling Suite Implementation Guide* to see if an upgrade is supported for your release. To run this target, enter:

```
sdk addMigrateSegData DEBS_m_to_DEBS_n
```

Where *m* is the release number of the source release and *n* is the target release. For example:

```
sdk addMigrateSegData DEBS_7_0_2_to_DEBS_7_1
```

cleandeploy

Use this target to build the current project and copy it over to the deployment directory of your servlet container. This target deletes the contents of the build directory and any pre-existing deployment of the project in the servlet container. To run this target, enter:

```
sdk cleandeploy
```

copyproject

Use this target to make a copy of the current project. To run this target, enter:

```
sdk copyproject <Name of new project>
```

After running this target to create a new project, then the first time you run the merge target, make sure to run it with the `-clean` flag: that is:

```
sdk merge -clean
```

createCustomDB

Runs the custom SQL scripts that you have created as part of your project. To run this target, enter:

```
sdk createCustomDB
```

createDB

Creates the transactional database schema in the target database. The database type used is determined by the value of the `DB_TYPE` property specified in the ***project_env.properties*** file. To run this target, enter:

```
sdk createDB
```

Make sure that you double-check the entries for the database properties in the ***project_env.properties*** before running this target: any existing data will be erased as the database schema is created.

To run this target, make sure that you have the appropriate database client applications installed. See the *Sterling Multi-Channel Selling Suite Implementation Guide* for further information.

createSegDB

Creates the segmentation database schema in the target database. The database type used is determined by the value of the `DB_TYPE` property specified in the ***project_env.properties*** file. To run this target, enter:

```
sdk createSegDB
```

This target applies to Release 8.x and later.

customize

Copies files from the appropriate ***sdk_home/releases/*** directory to the corresponding directory under the project label. To run this target, enter:

```
sdk customize <name of file>
```

This target copies files from the release WAR file or from the release overlay directory: if the file exists under the release ***overlay/*** directory, then this copy takes precedence.

customizeFile

Copies files from the appropriate *sdk_home/releases/* directory to the corresponding directory under the project label. Specify the full pathname of the file. To run this target, enter:

```
sdk customize <pathname>\<name of file>
```

This target copies files from the release WAR file or from the release overlay directory: if the file exists under the release **overlay/** directory, then this copy takes precedence.

default-dist

Called by the distWar command.

deleteproject

Removes the project and build directory for the named project. To run this target, enter:

```
sdk deleteproject <name of project>
```

deploy

This target should only be used when you run Apache Tomcat on the same machine as the SDK. It copies the files from the *sdk_home/builds/project/* directory over to the servlet container directory. To run this target, enter:

```
sdk deploy
```

To use this target you must specify the container.home and container.version properties in the *sdk_home/local-sdk.properties* file. For example:

```
container.version=4.1  
container.home=C:/Program Files/Apache Tomcat 4.0
```

The container version should be in the form *m.n*. You must use forward slashes on both UNIX and Windows systems.

dist

Creates a WAR file from the merged project and puts it in the *sdk_home/dist/* directory. The WAR file is named with a timestamp. To run this target, enter:

```
sdk dist
```

An optional -release flag will set the name of the generated WAR file to **Comergent.war** (releases earlier than 8.x) or **Sterling.war** (releases 8.x and later). If the container.name property is set to “weblogic” and the container.version

property is set to “7.0.2” or “7.1”, then the generated WAR file is created with the JSP pages under **web/** rather than **WEB-INF/web/**.

distData

Use this target to create a JAR file to use with the data loading tool. To run this target, enter:

```
sdk distData
```

distSQL

Use this target to create a JAR file to use with the schema creation. To run this target, enter:

```
sdk distSQL
```

distWar

Use this target to create a WAR file for your project. To run this target, enter:

```
sdk distWar
```

eclipseSetup

Use this target to enable a project to work with the Eclipse plugin. See CHAPTER 3, “Eclipse Plugin” for more information about the Eclipse plugin. This target creates files within the *sdk_home/projects/project/* directory so that the SDK project can be run as an Eclipse project. To run this target, enter:

```
sdk eclipseSetup
```

The target creates a **.project** configuration file that specifies Eclipse properties for the project, and a **.classpath** configuration file that specifies the classpath that should be used when using Eclipse to build and debug the project.

encryptVal

Use this target to encrypt strings. To run this target, enter:

```
sdk encryptVal <string to be encrypted>
```

The result is an encrypted form of the string. The string is encrypted using a default encryption scheme that is provided with the Software Development Kit.

env.setDBType

Sets the DB_TYPE property in the current *project_env.properties* file. It also modifies the **DataServices.xml** file to comment in or out the JDBC Driver elements. To run this target, enter:

```
sdk env.setDBType <Database type>
```

Valid values for the database type parameter are:

- DB2: for installations running against IBM DB2 Universal Database connecting through a JDBC driver.
- MsSql: for Release 6.3.1 and earlier installations running against SQL Server and connecting through the MSSql DLL.
- MSSQLJDBC: for Release 7.0.1 and later installations running against SQL Server and connecting through a JDBC driver.
- ODBC: for Release 6.4.1 through 7.2 installations running against SQL Server and connecting through the ODBC DLL.
- Oracle: for installations running against Oracle Database Server connecting through a JDBC driver.

The optional `-all` flag sets the `DB_TYPE` property for all environments in the project. For example:

```
sdk env.setDBType Oracle -all
```

examine

Copies files from the release WAR file or from the release overlay directory to a ***sdk_home/releases/release/examine/*** directory. This is a convenience method so you can examine files without copying them over to the project directory. To run this target, enter:

```
sdk examine <name of file>
```

fastcleandeploy

This target is the equivalent of running `merge -clean` followed by `fastdeploy`. To run this target, enter:

```
sdk fastcleandeploy
```

fastdeploy

This target should only be used when you run Apache Tomcat on the same machine as the SDK. Note that it behaves differently depending on your version of Tomcat as follows.

For Version 4.1.x, this target:

- Modifies the Tomcat ***server.xml*** configuration file by specifying that the Comergent context uses the files under the ***sdk_home/builds/project/*** directory as the Web application.

- Adds an XML file to the *container_home/webapps/* directory that declares the Comergent context.

For Version 5.5.x, this target:

- Adds an XML file to the *container_home/conf/Catalina/localhost/* directory that declares the Comergent context.
- Modifies the *sdk_home/builds/project/WEB-INF/web.xml* configuration file to point to a local copy of the **prefs.xml** configuration file.

To use this target you must specify the *container.name*, *container.home*, and *container.version* properties in the *sdk_home/my-sdk.properties* file. For example:

```
container.name=tomcat
container.version=4.1
container.home=C:/tomcat4129
```

or

```
container.name=tomcat
container.version=5.5
container.home=C:/tomcat559
```

Note that this target also changes the Tomcat server configuration using the *container.server* properties. Check these before running *fastdeploy*.

To undo the effect of this target, you must remove the **Comergent.xml** from the appropriate directory and remove the appropriate Context element from the **server.xml** file.

fileConvert

For Release 7.0 and higher, use this target to convert Java and JSP files so that they use the public APIs supported by the current release. This target takes a parameter specifying the conversion path. For example:

```
sdk fileConvert 702to71
```

converts Release 7.0.2 Java and JSP files to use the public APIs supported by Release 7.1.

fileConvert67to70

Use this target to convert Java and JSP files so that they use the public APIs supported by Release 7.0.1. To run this target, enter:

```
sdk fileConvert67to70
```

When this target encounters a file that needs to be upgraded it saves a copy of the original file with the suffix “.orig”.

generateBean

Generates beans from the *sdk_home/builds/project/WEB-INF/schema/* files. Before generating the beans, any customized business objects in *sdk_home/projects/project/WEB-INF/schema/* and *sdk_home/projects/project/WEB-INF/schema/custom/* are merged with the data objects in *sdk_home/builds/project/WEB-INF/schema/*. To run this target, enter:

```
sdk generateBean
```

Note that if you add data objects to the schema, then you must add the corresponding data elements and recipes to the **DsDataElements.xml** and **DsRecipes.xml** configuration files in the *sdk_home/projects/project/WEB-INF/schema/custom/* directory. You must run the merge target before running the generateBean target because the schema files are read from the builds directory for the project.

generateDTD

Generates the DTDs from the *sdk_home/builds/project/WEB-INF/schema/* files. Before generating the DTDs, any customized data objects in *sdk_home/projects/project/WEB-INF/schema/* and *sdk_home/projects/project/WEB-INF/schema/custom/* are merged with the data objects in *sdk_home/builds/project/WEB-INF/schema/*. To run this target, enter:

```
sdk generateDTD
```

generateIndexFile

Generates the SDK index page. This is the main home page for the SDK documentation and information about your releases and projects. To run this target, enter:

```
sdk generateIndexFile
```

generateJSPResourceBundles

Use this target to generate JSP resource bundles for internationalizing a project. This target creates .properties files for each .jsp in your project directory. You can then bundle the .properties files for translation. To run this target, enter:

```
sdk generateJSPResourceBundles
```

getVal

Use this target to retrieve the current value of a property in your project. To run this target, enter:

```
sdk getVal <name of property>
```

The name of the property should be specified as its qualified location in the property tree, but without the “Comergent.” root: for example, “C3_Campaigns.CampaignSMTPServer”.

help

Use this target to obtain a list of executable targets. If you provide the name of a target as a parameter, then you can read what the target does and what parameters it takes. To run this target, enter:

```
sdk help
```

or

```
sdk help <name of target>
```

importProperties

Use this target to generate a **prefs.xml** file using your project’s current properties. To run this target, enter:

```
sdk importProperties
```

info

Use this target to query an installation JAR file: the target reports on the vversion information provided by the **version.xml** file contained in the JAR file. To run this target, enter:

```
sdk info <location of JAR file>
```

install

Copies the files from the Sterling Multi-Channel Selling Suite Web application WAR file to the appropriate sub-directory of the **sdk_home/releases/** directory. This target supports two modes:

- A mode to support installations of SDK Archive files built to the Version 2.0 standard. You can install releases, documentation, tools, and hot fixes using this mode. To run this target in this mode, enter:

```
sdk install <location of JAR file>
```
- A backward-compatible mode to support installations of older releases. To run this target in this mode, enter:

```
sdk install debs-6.3.1 <location of Comergerent.war file>
<location of Source.jar file>
```

If you install a hot fix using this target, then when you next build your project, you must run `merge -clean` or `merge -full`.

installDB2

Updates the project files to include the appropriate JDBC driver JAR file. When new projects are created, then JAR file is copied over to the **lib**/directory of the project. To run this target, enter:

```
sdk installDB2 <full path to JAR file>
```

installMsSql

Copies the **MsSqlJNI.dll** file to the **system32**/directory of the machine on which the SDK is running. In Version 2.0.1 and higher, a separate target, `env.setDBType`, is used to change the **DataServices.xml** file. To run this target, enter:

```
sdk installMsSql
```

Note:	This does not copy the MsSqlJNI.dll file to the servlet container machine. Copying this file to the system32 / directory of the servlet container machine must be done manually.
--------------	--

installMSSQLJDBC

Installs the JDBC JAR file provided as the parameter into the **overlay**/ directory of the current release, and **WEB-INF/lib**/ directory of the current project. When new projects are created, then the JAR file is copied over to the **lib**/directory of the project. To run this target, enter:

```
sdk installMSSQLJDBC <full path to JAR file>
```

installODBC

Copies the **ODBC.JNI.dll** library file to the **system32**/ directory of the machine on which the SDK is running. Use the `windows.system.dir` property in **my_sdk.properties** file to specify where this (usually `C:/WINDOWS/system32` or `C:/WINNT/system32`). To run this target, enter:

```
sdk installODBC
```

installOracle

Installs the JAR file provided as the parameter into the **overlay**/ directory of the current release, and **WEB-INF/lib**/ directory of the current project. When new projects are created, then the JAR file is copied over to the **lib**/directory of the project. To run this target, enter:

```
sdk installOracle <full path to JAR file>
```

loadAnderelDB

Used for development and testing purposes. This target loads the Anderel reference data for the specified project. To run this target, enter:

```
sdk loadAnderelDB
```

loadDB

This target will load data for the specified project. It executes the **MinimalData.lst** file in the *sdk_home/builds/project/WEB-INF/scripts/* directory to load a minimal set of data. Developers should extend the **MinimalData.lst** file to include references to any additional XML files they require to be loaded as part of a database initialization. To run this target, enter:

```
sdk loadDB
```

By default, only the minimal data set as defined by the Sterling Multi-Channel Selling Suite is loaded.

loadMatrixDB

Used for development and testing purposes. This target loads the Matrix reference data for the specified project. To run this target, enter:

```
sdk loadMatrixDB
```

loadSegDB

This target loads the segmentation database created with the **createSegDB** target. This target applies to Release 8.x and later.

loadSegMatrixDB

Used for development and testing purposes. This target loads the segmentation database created with the createSegDB target with Matrix reference data. To run this target, enter:

```
sdk loadSegMatrixDB
```

merge

Merges in the customizations from the *sdk_home/projects/project/* directory. It compiles any classes that are either modifications of a Sterling Multi-Channel Selling Suite Java source or new class files written as part of the customization. To run this target, enter:

```
sdk merge
```

If you run this target with the `-clean` flag, then the *sdk_home/builds/project/* directory is deleted, and the project is built against a fresh copy of the current release. If you have installed any hot fixes into the release, then you must run `merge-clean` to ensure that the hot fix is built into the build of the project.

If you run this target with the `-light` flag, then only the project Java code is compiled.

methodology

This target copies the methodology template files.

migrateDB

Use this target to upgrade a project's transactional database schema from one release (the source) to another (the destination). Check the *Sterling Multi-Channel Selling Suite Implementation Guide* to see if an upgrade is supported for your release. To run this target, enter:

```
sdk migrateDB DEBS_m_to_DEBS_n
```

Here *m* is release number of the source release and *n* is the destination release. For example:

```
sdk migrateDB DEBS_7_0_2_to_DEBS_7_1
```

In Release 8.x and later, the `migrateDB` target migrates the transactional database to the new release.

migrateSegDB

Use this target to upgrade a project's segmentation database schema from one release (the source) to another (the destination). Check the *Sterling Multi-Channel Selling Suite Implementation Guide* to see if an upgrade is supported for your release. To run this target, enter:

```
sdk migrateSegDB DEBS_m_to_DEBS_n
```

Here *m* is release number of the source release and *n* is the destination release. For example:

```
sdk migrateSegDB DEBS_7_0_2_to_DEBS_7_1
```

In Release 8.x and later, the `migrateSegDB` target migrates the segmentation database to the new release.

newproject

Creates the new project sub-directories. It takes the following argument:

- the name of the project

This target copies files from the appropriate release to the *sdk_home/builds/* directory for use in this project. The release is set in the *default.debs.version* property of **sdk-settings.properties**. It also generates custom *.sql files that can be used to manage any customizations that you want to make to the database schema. To run this target, enter:

```
sdk newproject <name of new project>
```

project

This target is used to invoke a target that is defined in a **build.xml** file that is present in the project. It provides a way of providing custom targets that are specific to a release or a project. To run this target, enter:

```
sdk project <name of target>
```

The named target must be defined in the *sdk_home/projects/project/build.xml* file (if it exists).

project.analyze

Use this target as part of preparation for upgrading your project. This target examines the current project, identifies files that have been customized, and creates two files in the *sdk_home/reports/project/upgrade* directory:

project_files_list.xml, containing the list of the changed files, and
project_status.html for viewing the list. Use this list to ensure that your customizations are preserved during the upgrade process.

project.createHotFix

Creates a JAR file that can be applied to a release as a patch using the install target. You must provide a designator that identifies this hot fix: this can be a bug number or other unique identifier. To run this target, enter:

```
sdk project.createHotFix <hot fix designator>
```

The resulting JAR file is created in the *sdk_home/dist/* directory. Its name is **hotfix_<Hot fix designator>_<Timestamp>.jar**. An optional flag *-release* removes the timestamp from the name of the hot fix JAR file. For example:

```
sdk project.createHotFix -release 5208
```

project.generateIndexFile

This target generates a new index file for the specified project. To run this target, enter:

```
sdk project.generateIndexFile debs-<release>
```

For example:

```
sdk generate.generateIndexFile debs-8.0
```

project.preCompileAllJSPs

This target precompiles all JSPs in your project. To run this target, enter:

```
sdk project.preCompileAllJSPs
```

setupDB

This target creates the transactional and segmentation databases and loads them with the reference dataset. This target applies to Release 8.x and later.

The setupDB target runs the following targets in this order:

1. createDB
2. loadDB
3. createSegDB
4. loadSegDB

Note that setupDB runs the loadDB target to load the transactional database with the minimal data set. If you plan to use the Matrix reference data, you must run each of the setupDB targets manually, substituting loadMatrixDB for loadDB.

To run this target, enter:

```
sdk setupDB
```

setupMatrixDB

This target creates the transactional and segmentation databases and loads them with the Matrix reference dataset. This target is used for development and testing purposes. This target applies to Release 8.x and later.

The setupMatrixDB target runs the following targets in this order:

1. createDB
2. loadMatrixDB
3. createSegDB
4. loadSegMatrixDB

To run this target, enter:

`sdk setupMatrixDB`

Project Migration Targets

The following project targets are used to upgrade a project from one release of the Sterling Multi-Channel Selling Suite to another:

project.premigrate

Performs an analysis of the project by comparing it to the new release to which you want to upgrade it. It takes one argument: the release to which you want to upgrade the project. The target release must support upgrades from the current release.

`sdk project.premigrate debs-6.7`

project.migratefiles

Migrates the files that require modification in the project.

Attention: Make sure that you have your three-way merge tool set up correctly before running this target. See “Merge Tools” on page 21 for more information.

When this target encounters a file that needs to be merged it saves a copy of the original file with the suffix “.save”.

project.postmigratecleanup

This target tidies up temporary files and directories generated by the migration process.

In general, after you have migrated a project to a new release, make sure that you switch to using the new release (using the `switchdebs` target) and then run `merge -clean` to delete the current project builds directory for the project and re-populate it with the new release. If you are working with the Eclipse plugin, then you should also run the `eclipseSetup` target: this will regenerate the **.classpath** configuration file against the new release.

project.merge

Merges the current project with a second project to create a third. To run this target, enter:

`sdk project.merge <second project> <new project>`

project.mergfiles

Run this target after you run the `project.merge` target.

projectjar

Use this target to JAR up the current project. To run this target, enter:

```
sdk projectjar
```

Servlet Specification Targets

The setservletspec targets ensure that when the Sterling Multi-Channel Selling Suite Web application **Sterling.war** file is created, it conforms to the servlet specification suitable for the intended servlet container. You must run a setservletspec target for each project, so that if you switch projects or create a new project, make sure that you run one of these targets after switching to the new project. The *Sterling Multi-Channel Selling Suite Implementation Guide* provides a summary of which versions of the servlet specification are supported by each servlet container.

setservletspec2.2

Switches to using the J2EE Servlet Specification 2.2. To run this target, enter:

```
sdk setservletspec2.2
```

setservletspec2.3

Switches to using the J2EE Servlet Specification 2.3. To run this target, enter:

```
sdk setservletspec2.3
```

setup

Generates the **my_sdk.properties** file. Run this target immediately after you install the SDK. To run this target, enter:

```
sdk setup
```

To migrate projects from an earlier installation of the SDK, enter:

```
sdk setup <sdk_home of Version 7.2>
```

Projects in the Version 1.1 SDK are copied over to the new SDK. You must run merge -clean on these migrated projects to ensure that the builds directory is created for each project.

setVal

Use this target to set the value of a property used by a Release 7.0 or later installation of the Sterling Multi-Channel Selling Suite. To run this target, enter:

```
sdk setVal <name of property> <value of property>
```

startPreferencesTool

Use this target to run the Preferences tool used to set values in the **prefs.xml** configuration file. To run this target, enter:

```
sdk startPreferencesTool
```

switchdebs

Changes the value of the default.debs.version property in **sdk-settings.properties**. To run this target, enter:

```
sdk switchdebs <new release of DEBS, such as debs-6.3>
```

switchenv

Changes the value of the deploy.environment property in the **sdk-settings.properties** file. This property determines which **project_env.properties** file is used in building projects. To run this target, enter:

```
sdk switchenv <dev|local|prod|qa>
```

switchproject

Changes the value of the project.name property in **sdk-settings.properties** so that you can work on a different project. To run this target, enter:

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

switchproject-helper

Changes the value of the project.name property in **sdk-settings.properties** so that you can work on a different project. To run this target, enter:

```
sdk switchproject-helper <new project name>
```

touchfile

Use this target to update the timestamp of the specified file(s). To run this target, enter:

```
sdk touchfile
```

You can use the “*” wild card character to match the name of more than one file in your project.

touchproject

Updates the timestamp of all the files in the current (as specified by the project.name property) project directory. This ensures that these files are all regarded as being newer than the corresponding files in the build directory, and hence they will be copied across to the build directory. To run this target, enter:

`sdk touchproject`

version

Provides basic information about the SDK. To run this target, enter:

`sdk version`

Use the optional `-check` flag to test whether the SDK has been modified.

This chapter describes the Eclipse plugin that may be used to integrate the SDK with an installation of Eclipse Release 3.1. It covers:

- “Overview” on page 43
- “Installation” on page 44
- “Using the Plugin” on page 45

Overview

The Eclipse application has become a widely used integrated development environment (IDE) for Java developers. Its architecture supports the ability to add features to the core Eclipse application through *plugins*: these are separately deployable components that add functionality to the core application. Eclipse comes with a set of standard plugins, and you can add plugins to your installation of Eclipse at any time.

The functionality provided by a plugin can vary a great deal. For example, plugins are available to support debugging Web applications running in Tomcat, developing Web services, working with a source control system such as CVS, and for developing new plugins.

Typically, a plugin is delivered as a ZIP file which contains the files that make up the plugin. To install the plugin, you simply unzip the ZIP file into the

eclipse_home/plugins/ directory. When Eclipse is next started up, the new plugin's functionality is available.

Sterling Commerce provides a plugin for Eclipse that integrates the SDK with Eclipse. This plugin can be used to run all the SDK targets, and to build and debug SDK projects using Eclipse.

Installation

To Install the SDK Plugin

1. In Eclipse, click **Help -> Software Updates -> Find and Install...**
2. Select the **Search for new features to install** radio button and click **Next>**.
3. Click **New Remote Site...**
4. In the New Update Site dialog box, enter the URL to the server hosting the plugin and click **OK**.
5. Click **Finish**.
6. In the Search Results window, select the Comergent SDK check box and click **Next>**.
7. In the Feature License window, review and accept the License. Click **Next>**.
8. Click **Finish**.

The plugin will be downloaded and installed. You will be prompted to restart Eclipse.

9. Restart Eclipse.
10. In the Eclipse window, click **Window -> Preferences...**
11. Click the Comergent SDK node in the navigation tree.
12. In the Comergent SDK panel, click the ... button and browse to the location of the **sdk.bat** file in your *sdk_home* directory. Select **sdk.bat** and click **Open**.
13. Click **OK**.
14. In the **Preferences** window, select Java -> Build Path -> Classpath Variables.
15. Click **New...**
16. Create the following new variables with their values:
 - **SDK_HOME**: set to the *sdk_home* directory.

17. In the **Preferences** window, select Java -> Compiler -> Building.
18. Uncheck the **Scrub out folders when cleaning projects** check box.
19. Click **OK**.
20. In the **Preferences** window, select General -> Workspace -> Linked Resources.
21. Click **New...**
22. Create the following new variables with their values:
 - SDK_HOME: set to the *sdk_home* directory.
23. In the **Linked Resources** window, click **OK**.
24. Click **OK**.

Using the Plugin

You can use the plugin as a front-end to all your SDK tasks. In particular, you can run SDK targets as follows:

To Run an SDK Target

1. If it is not already open, then open the Comergent Perspective, by clicking **Window -> Open Perspective -> Other** and select the Comergent SDK perspective from the list of available perspectives.
2. Click **Open**.
3. In the Comergent Perspective, navigate through the Tasks panel to locate the target you want to run.
4. Double-click the target. If the target requires parameters, then a dialog box prompts you for the parameter values.

The target will run as if you had entered it at the SDK command line.

To Import an SDK Project into Eclipse

If you have created a project using the SDK, then you can import it into Eclipse to make an Eclipse project.

1. Click **File -> Import...**
2. Select **Existing Projects into Workspace** and click **Next>**.

3. Select a root directory for the Eclipse project by clicking **Browse...** next to the Select root directory text field and browse to the *sdk_home/projects/project/* directory.
4. Click **Finish**.

The SDK project is imported as an Eclipse project with the same name.

Building Projects Using the Plugin

You can use the native capabilities of Eclipse to build and debug your project. However, before doing so, check that the following line in the project **.classpath** configuration file uses the correct location of the source ZIP file:

```
<classpathentry
    sourcepath="C:\sd31_home/releases/debs-6.7/reference/src.zip"
    kind="lib" path="projectbuilds" />
```

Index

Symbols

.classpath configuration file 29
.project configuration file 29

A

addMigrateData target 26
addMigrateSegData target 26
app.name property 19
apps.dir property 19

B

builds directory 22

C

clean flag
 use in merge target 36
cleandeploy target 26
configuration files
 customizing 16
conflictingfiles.log upgrade file 8
container.home property 19, 28, 31
container.name property 28, 31
container.server properties 31
container.version property 28, 31
copyproject target 26
createCustomDB target 27
createDB target 6, 7, 14, 27

creating patches 20
customize target 4, 5, 15, 27
 handling patch files 21
 use of overlay files 23
customized targets 25
customizeFile target 28
customizing configuration files 16

D

data elements
 customizing in SDK 32
data objects
 adding to schema 32
DB_TYPE property 27, 29
debs.version property 4
default.debs.version property 37, 41
default_sdk.properties configuration file 9
default-dist 28
deleteproject target 28
deploy target 19, 28
deploy.environment property 5, 19, 22, 41
diff3.executable property 21
dist directory 22
dist target 6, 14, 19, 28
distData target 29
distSQL target 29
distWar target 19, 29

docs directory 22
documentation 3, 12
DsDataElements.xml configuration file 32
DsRecipes.xml configuration file 32

E

Eclipse 43
eclipseSetup target 29
encryptVal target 29
env.setDBType target 5, 13, 29, 34
environment variables
 JAVA_HOME 9
environments
 SDK 14
ERD diagrams 12, 24
etc directory 22
examine directory 23
examine target 23, 30
 handling patch files 21
 use of overlay files 23

F

fastcleandeploy target 30
fastdeploy target 30
fileConvert target 31
fileConvert67to70 target 31
filestobedeleted.log upgrade file 8
filestobeupgraded.log upgrade file 8

G

generateBean target 32
generateDTD target 32
generateIndexFile target 32
generateJSPResourceBundles target 32
getVal target 33

H

help target 33
hot fix
 creating 37
hot fixes
 installing 34
 running merge -clean 36

I

image directory 23
importProperties target 33
Index docs 24
info target 33

install target 3, 12, 20, 22, 24, 33
 installing a hot fix 37
 installing new releases 19
installDB2 target 5, 13, 34
installing the product documentation 3
installMsSql target 13, 34
installMSSQLJDBC target 5
installODBC target 5, 13, 34
installOracle target 5, 13, 34

J

J2EE Servlet Specification 2.2
 switching using the SDK 40
J2EE Servlet Specification 2.3
 switching using the SDK 40
Javadocs 12, 24
JSP pages
 customizing 16

L

loadAnderelDB 35
loadDB target 6, 7, 14, 35
loadMatrixDB target 7, 14, 35
loadSegMatrixDB 35

M

merge 35
merge target 5, 7, 19, 32, 35
 -clean option 20
 use of overlay files 23
 using -clean when migrating
 projects 39
merge tools 21
mergetool.name property 21
methodology target 36
migrateDB target 36
migrateSegDB target 36
my_sdk.properties configuration file 9, 21,
 23, 40

N

newproject target 5, 12, 36

O

ODBCJNI.dll library file 34
orig suffix
 created by the fileConvert target 32
overlay directory 23
 use in patches 21

P

- patches
 - applying multiple patches 21
 - creating 20
- plugins 43
- prefs.xml configuration file 31, 41
- product documentation 3, 12
- project target 37
- project.analyze target 37
- project.createHotFix target 37
 - using to create patch 20
- project.generateIndexFile 37
- project.merge target 39
- project.mergfiles 39
- project.migratefiles target 20, 21, 39
- project.name property 41
- project.postmigratecleanup target 20, 39
- project.preCompileAllJSPs 38
- project.premigrate target 20, 39
- project.properties configuration file 8
- projectjar target 40
- projects directory 22
- properties
 - setting in SDK 8

R

- recipes
 - customizing in SDK 32
- releases directory 22

S

- save suffix
 - created by the project.migratefiles target 39
- scope
 - of a target 25
- SDK 1
 - directories 22
 - merge tools 21
- SDK 3.5 supported releases 1
- SDK index docs 12
- SDK properties 8
- sdk-settings.properties configuration file 8, 23
- setservletspec2.2 target 40
- setservletspec2.3 target 14, 40
- setup target 9, 11, 40
- setupMatrixDB 38
- setVal target 40

- Software Development Kit 1
- startPreferencesTool target 41
- substituting token values during merge 5
- substituting token values in projects 16
- switchdebs target 8, 19, 20, 41
 - use in migrating projects 39
- switchenv target 6, 8, 41
- switchproject target 8, 41
- switchproject-helper target 41

T

- targets
 - createDB 27
 - customize 15, 27
 - customizeFile 28
 - deploy 19, 28
 - dist 28
 - distWar 19
 - generateBean 32
 - generateDTD 32
 - install 12, 33
 - installOracle 34
 - loadDB 35
 - loadMatrixDB 35
 - merge 19, 35
 - metholdology 36
 - newproject 12, 36
 - project.analyze 37
 - SDK 9
 - setservletspec2.2 40
 - setservletspec2.3 40
 - switchproject 41
 - switchproject-helper 41
 - swtichdebs 41
- tmp directory 23
- tokenized property values 16
- tokens
 - substituting during merge 5
- touchfile target 41
- touchproject target 41

U

- upgrade files 8

V

- version target 42

W

- web.xml configuration file

modified by fastdeploy target 31
windows.system.dir property 34
workspaces directory 23