

Visual Modeler

Application Guide

Release 9.0

Last updated in HF1

Copyright © 1998-2010.
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, MERCHANTABILITY, 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."

EHCACHE 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

Contents

CHAPTER 1 Checklist.....	31
Tasks Checklist.....	31
Part 1: Installation	35
CHAPTER 2 Architecture and Configuration Overview ...	37
Architecture	37
CHAPTER 3 Implementation Overview	41
Implementation Tasks.....	41
Implementation Methodology	41
Steps to Implementation.....	42
Implementing the Visual Modeler Integration	44
High Availability and Load Balancing	45
Integration Security Issues	45

CHAPTER 4 Installation Worksheet.....	47
CHAPTER 5 Installation Requirements	49
Hardware Requirements.....	49
<i>Windows 2000</i>	49
<i>UNIX</i>	50
Software Requirements	50
<i>Operating Systems</i>	50
Hewlett-Packard UNIX	50
Linux	50
Microsoft Windows	50
Sun SPARC Solaris	50
<i>Java Development Kit</i>	51
SDK	51
<i>Servlet Containers</i>	51
Servlet Container Clustering	52
Network Requirements	52
Browser Requirements	52
<i>Security Settings</i>	52
Firefox	52
Internet Explorer	53
Mozilla	53
<i>Character Sets</i>	53
Database Server Requirements	54
<i>Database Owner Requiements</i>	54
<i>Microsoft SQL Server Requirements</i>	55
General Requirements	55
SQL Server 2005 Communication Requirements	56
<i>Oracle Requirements</i>	56
UNIX	56
Oracle Communication Requirements	56
Sizing Requirements	57

CHAPTER 6 Installing the Visual Modeler.....59

Installation Overview	60
<i>Preparing to Install</i>	61
<i>Installing the Software Development Kit</i>	61
<i>Installing the Visual Modeler Using the SDK</i>	61
<i>Deploying the Sterling Web Application</i>	61
<i>Database Server Steps</i>	61
Preparing to Install	61
Configuring the Transactional and Segmentation Databases	64
<i>Database Configurations</i>	65
<i>Determining Configuration Type</i>	66
Determining the Oracle Configuration	66
Determining the SQL Server 2005 Configuration	66
Oracle and SQL Server Properties	66
SQL Server 2005 Replication Properties	67
SQL Server 2005 Migration Properties	68
<i>Database Communication Setup</i>	68
Oracle Setup	69
SQL Server 2005 Setup	71
SQL Server 2005 Setup - Migration with Two Servers	72
Installing the Software Development Kit.....	74
Installing the Visual Modeler Using the SDK.....	75
<i>Email Addresses</i>	81
Configuration Files	82
Minimal Data	82
<i>ObjectMap Settings</i>	82
Deploying the Sterling Web Application.....	83
<i>XML Parser Settings</i>	84
<i>Tomcat Releases</i>	85
Notes on Using Apache Tomcat	86
<i>WebLogic Releases</i>	87
Pre-Compiling JSP Pages	89
XML Parsing	90
<i>Solaris and Oracle OCI Driver</i>	90
<i>Further Deployment Steps</i>	90
Matrix Reference Segments Setup	91
Installing the Reference Visual Modeler	91

Default XML Identity Setup	102
<i>Default XML Identity Configuration</i>	103
<i>Trusted IP Address Configuration</i>	104
<i>Default XML Identity Request Authentication</i>	105
AuthenticationAPI	107
Database Server Steps	107
<i>Support for Oracle Server</i>	107
<i>Support for SQL Server</i>	108
Managing Database Connections	109
<i>Configuration Files</i>	109
<i>Connection Pooling</i>	109
Common Problems	110
What is the purpose of connection pooling?	110
How does Sterling's connection pooling work?	110
Why are there separate query and update connection pools?	110
How do I validate connections prior to reuse?	111
How can I limit the number of connections used?	111
How can I free up connections when demand drops?	111
What happens when connection limits are reached?	111
Why are the connection limits on the data source?	112
<i>Common Problems</i>	112
My database requests fail with a "connection reset by peer" message	112
My database connections are not being released when traffic drops	112
I share a database with other applications. I cannot allow the Visual Modeler to use more than n connections	112
Pagination Settings	112
Setting the Session Timeout	113
Modifying the URL for the Web application DTD	114
Managing Memory	114
Configuring Ehcache	115
High Availability and Clustering	115
Sharing Directories	115
Directory and File Organization	116
Cron Job Setup	118

Setting Up Apache as a Front-end to Tomcat.....	119
Prerequisites	119
Overview	119
Configuring Apache to Use mod_jk.....	120
Configure Tomcat to Use mod_jk.....	121
Starting Apache and Tomcat.....	121
Setting up Apache to Support SSL	122
Keep Alive Settings 123	
Filtering Static Content.....	123
Setting up Apache to Serve Static Content.....	123
Creating a NSAPI Filter	124
Compressing Output From the Visual Modeler.....	125
CHAPTER 7 Creating and Populating the	
 Knowledgebase.....	127
Gathering the Database Information	127
Creating the Knowledgebase Schema	128
Creating the Schema	128
Locales and Loading Data Using the XML Loader 129	
Populating the Knowledgebase	131
XML Data Format	131
XMLLoader Script	132
Encryption.....	132
Defining the Knowledgebase as the Data Source.....	133
MsSqlDataSources Syntax 133	
OracleDataSources Syntax 133	
Internationalization and Support for Locales.....	134
Creating Locales 134	
Updating Data using XMLLoader 134	
Database Server-Specific Steps	135
SQL Server Steps 135	
Oracle Steps 135	
Data Sets	135
Removing Locales.....	137
Logging into the Visual Modeler.....	137

CHAPTER 8 Troubleshooting and Backing Up the Visual Modeler..... 139

Troubleshooting	139
<i>Testing with the Administration URL</i>	139
<i>Email Server</i>	139
General Troubleshooting Tips.....	140
<i>Tomcat Server</i>	140
Common Problems.....	140
<i>Errors at Startup Time</i>	141
<i>Errors at Runtime</i>	144
Backing Up the Visual Modeler.....	145

CHAPTER 9 Managing Visual Modeler Logging 147

Logging	147
<i>Logging Preferences and Configuration</i>	147
Logging to the Console	148
Changing Logging Level for a Package	148
Formatting Logging	149
Logging File Size	149
<i>Making Transient Logging Configuration Changes</i>	149
<i>Logging File Locations</i>	150

CHAPTER 10 Localization Concepts..... 151

Localization Concepts	152
<i>Built-in Localization Support</i>	152
<i>Locale Specification</i>	152
<i>Using Locales</i>	153
User Locales	153
Default Locales for Languages	153
<i>Sorting in Locales</i>	153
Localization Pack Installation Overview	154
Localization Pack Installation Steps: New Implementation	155
Localization Pack Installation Steps: Existing Implementation..	161

CHAPTER 11 General Security Considerations 169

General Architectural Concerns	169
Administration Model	170
Networks	170
Servers	170
Roles	171
Data Center Roles	172
Securing Users	173
SSL support	173
Setting Up Secure Message Types.....	173
Example Usages.....	175
Protecting the Authenticated Environment	175
Protecting the Enterprise Environment	175
Protecting Credit Card Information	175
Protecting User Administration Pages	176
Installing Certificates for SSL	176
Overview	176
Storing Data in Encrypted Form.....	177
General Setup	178
Symmetric Encrypter	179
Digester	181
Default Symmetric Encrypter and Digester	182
Changing Encryption Algorithms	182
Two-Way Encrypted Data	182
One-Way Encrypted Data	183
Key Stores and System Initialization.....	184
Wrapper Classes for Standard Algorithms	185
SymmetricEncrypter Class	185
Digester Class	185
Key Rotation	185
Key Rotation Procedure	185
Background	185
Why Rotate?	186
Basic Process	186
Detailed Process	186
Changing the Default Encrypter	187

Updating Existing Database Ciphers	188
Password Policies	189
Cross-Site Request Forgery Filter	189

CHAPTER 12 Testing the Visual Modeler Server 191

Starting the Visual Modeler Server	191
Troubleshooting	191
<i>Error Messages on Startup</i>	192
<i>Runtime Troubleshooting</i>	194
<i>Communication Between Enterprise Servers</i>	195

CHAPTER 13 Installing a Clustered Implementation 197

General Steps	197
<i>Terminology and Overview</i>	197
Administration Servers	198
Shared Files	199
<i>Load Balancer</i>	200
<i>General Installation Instructions for Clustered Deployment</i>	200
Setting Up a WebLogic Cluster	203
<i>Web Server</i>	203
<i>Administration and Managed Servers</i>	203
Preparation to Deploy the Visual Modeler Web Application	204
Deploying the Visual Modeler Web Application	204
SQL Server	205
Cron Jobs	206
Common Directories	206
SharedPublicServlet Class	207
Session Sharing	208
<i>Reloading Files</i>	209
<i>Running a Clustered WebLogic Installation</i>	209
Setting up a Database for Caching	210
<i>Introduction</i>	210
Setting up JavaSpaces for Caching	210

Introduction	210
Install the Required Servers	210
Part 2: Implementation	215

**CHAPTER 14 Integrating the Visual Modeler with
Selling and Fulfillment Foundation ..
217**

Integration Overview	217
Configuring the Visual Modeler Properties	218
Configuring the Sterling Configurator Rules	219

CHAPTER 15 Introduction to J2EE Web Applications
221

Architecture	221
Web Applications	221
web.xml File	222
JSP Pages	223
Model 2 Architecture	224
Controllers	225
Model	226
View	227
Further Reading	227

CHAPTER 16 System Architecture.....229

Visual Modeler Web Application	230
Processing Requests	231
Overriding MessageType Definitions	233
Default Elements	234
Key Java Classes	234
Wrapper Classes	234
ComergentContext	234
ComergentDispatcher	234

ComergentRequest	234
ComergentResponse	234
ComerentSession	235
<i>Servlets</i> 235
<i>Controller Classes</i> 236
Custom Controllers	236
SimpleController	237
MessagingController	237
<i>DataBean Classes</i> 237
<i>ObjectManager and OMWrapper Classes</i> 238
Creating Objects	238
Mapping Object Names to Object Classes	238
Restrictions	239
Passing Parameters	240
Object Pooling	240
<i>AppExecutionEnv Class</i> 241
<i>AppsLookupHelper Class</i> 241
<i>ComergentAppEnv Class</i> 242
<i>Global Class</i> 243
<i>GlobalCache Interface</i> 243
<i>LegacyFileUtils Class</i> 244
<i>OutOfBandHelper Class</i> 244
<i>Preferences Class</i> 245
<i>PriceCheckAPI Class</i> 246
Transactions 247
Message Conversion Classes 247
<i>Converter Classes</i> 247
Message Categories	247
Converter Interface	247
Support for Lookup Codes 248
What lookup support does the Visual Modeler provide?	249
Are string values localized?	249
How do I define a code to string mapping?	249
Are lookups performed for XML messages?	249
How is the lookup cache loaded?	249

CHAPTER 17 Platform Modularity.....251

Overview	252
Platform Modules	253
Module Interfaces	253
<i>Invoking Interfaces</i>	254
Platform Module Descriptions	254
<i>Access Policy</i>	254
<i>Authentication</i>	254
<i>Base64</i>	254
<i>Classpath Appender</i>	254
<i>Cryptography Service</i>	254
<i>Data Services</i>	255
<i>Dispatch Authorization</i>	255
<i>Dispatch Framework</i>	255
<i>Email Service</i>	255
<i>Event Service</i>	255
<i>Exception Service</i>	255
<i>Global Cache Service</i>	255
<i>Help</i>	255
<i>Initialization Service</i>	256
<i>Internationalization</i>	258
<i>Logging</i>	258
Configuration 258	
Loggers 259	
Appenders 259	
Layouts 260	
<i>Memory Monitor</i>	261
<i>Message Type Entitlement</i>	261
<i>Object Manager</i>	261
<i>Out Of Band Response</i>	261
<i>Preferences Service</i>	261
<i>Tag Libraries</i>	262
<i>Thread Management</i>	262
API and Usage 263	
<i>XML Message Converter</i>	263
<i>XML Message Service</i>	264
<i>XML Services</i>	264

CHAPTER 18 Introducing Data Beans and Business Objects 265

What are Data Beans?	265
<i>Lifecycle of a Data Bean</i>	266
<i>Defining a Data Bean</i>	267
<i>Defining the Structure of a Data Object</i>	267
Extending Data Objects 267	
<i>Data Bean and Business Object Creation</i>	268
<i>DataContext</i>	268
What is the DataContext class? 269	
What behavior can be controlled? 269	
What are the Cache Id methods for? 269	
How do Max Results and Num Per Page work? 270	
How do I instantiate a DataContext instance? 271	
What are the Default Settings for a new DataContext? 272	
<i>List Data Beans</i>	272
Application, Entity, and Presentation Beans	273
Using Stored Procedures	274
Data Bean Methods	274
<i>IData Methods</i>	275
<i>IRd and IAcc Interface Methods</i>	276
<i>Restoring and Persisting Data</i>	276
restore() Method 277	
persist() Method 278	
<i>Miscellaneous Methods</i>	278
getBizObj() Method 278	
writeExternal() Method 279	
<i>Child Data Objects</i>	279
<i>Extending Data Objects</i>	280
Data Bean Example.....	281
DsElement Tree.....	286
<i>DsElements</i>	287
<i>DsElement MetaData</i>	289
BusinessObject Methods.....	289
restore() Method 289	
persist() Method 290	

CHAPTER 19 Using the Security Mechanisms.....293

Managing Message Types	293
<i>Checking for Entitlement</i>	294
Managing User Types	295
<i>Adding a Role to a User Type</i>	295
<i>Creating a User Type</i>	295
Managing Access to Data Objects Using Access Policies	296
<i>Overview</i>	297
Inheritance 297	
<i>AccessPolicy.xml Configuration File</i>	297
Principal Qualifiers 297	
Access Policies 298	
Access Checkers 298	
Access Services 298	
Boolean Expressions 299	
<i>Example</i>	299
Password Policies	302
<i>Configuration</i>	302
<i>Creating a Custom Password Policy</i>	304
Passing Login Data Through a URL	304

CHAPTER 20 Logging307

Overview	307
<i>log4j.debug System Property</i>	308
Auditing Changes to Data Objects	309

CHAPTER 21 Modularity and Generated Interfaces... 311

Overview	311
Modules	312
Module Interfaces	313
<i>Invoking Interfaces</i>	314
Using the Object Manager 314	
Using Factory Classes 315	
Generated Interfaces	315

Example of a Generated Interface 316

CHAPTER 22 Implementing Logic Classes 319

Key Concepts 319
 Application Logic Classes 320
 Business Objects 321
 XML Schema 321
Naming Service 321
 NamingService Example 322

CHAPTER 23 Software Development Kit 323

Project Organization 323
 Project File and Directory Locations 324
 Java Source Files 324
 JSP Pages 324
 Schema Files 325

CHAPTER 24 Visual Modeler Localization 327

Overview 327
Supporting Locales 328
 Presentation and Session Locales 328
 JSP Pages and Properties Files 329
Notes 331
Debugging 332
 Failover Behavior 332
Resource Bundles 332
JSP Pages 333
 Methods to Retrieve Locales 333
 Using Properties Files in Code 334
Data for Internationalization 334
Email Templates 335
HTML Pages 336
Images 336
Javascript 337

JSP Pages.....	337
<i>Calendar Widget</i>	338
Style Sheets.....	339
System Properties	339
Resource Bundles and Formats	339
<i>PropertyResourceBundles and Properties Files</i>	339
<i>ResourceBundles</i>	339
<i>NumberFormats and DateFormats</i>	340
CHAPTER 25 Exceptions.....	343
ComergentException Hierarchy	343
<i>Exception Root</i>	343
ComergentException	343
ICCException	343
ComergentRuntimeException	344
<i>Subsystem Grouping</i>	344
<i>Subsystem by Subsystem Exception Policy</i>	345
Exception Chaining	345
Throwing, Catching, and Logging Exceptions.....	346
<i>When to Throw Exceptions</i>	346
<i>Throwing Runtime or Compile Time Exceptions</i>	346
<i>Catch Clauses and Throws Declarations</i>	346
<i>Logging Exceptions</i>	347
Displaying Exceptions.....	347
CHAPTER 26 Implementing Cron Jobs.....	349
Overview	349
<i>CronManager and CronScheduler</i>	350
<i>CronJob Interface</i>	350
CHAPTER 27 Filters	353
Filters Overview	353
Available Filters.....	354
<i>DosFilter</i>	354

<i>WSDLFilter</i>	355
-------------------------	-----

CHAPTER 28 Managing and Displaying Constrained Fields 357

Options	357
Criteria	358

CHAPTER 29 Security Best Practices 361

Introduction.....	361
Role Definition and Security Policies	363
<i>Administration Model</i>	363
Networks 363	
Servers 363	
Roles 364	
<i>Data Center Roles</i>	365
Information Assets	366
<i>Encryption of Persistent Data</i>	366
<i>Information Assets</i>	367
Account profile 367	
Transaction and System Log 367	
Key Store File 367	
WAR local key store file 367	
User passwords 368	
<i>Roles Schematic</i>	368
Protection Mechanism for Information Assets	369
<i>Credit Card Information</i>	369
<i>User Passwords</i>	370
Protection of Critical Functions	370
<i>Setting Application's Database User and Password</i>	370
Assertions 371	
<i>Storing Sensitive Data in the Database</i>	371
Assertions 371	
Threat Scenarios.....	372
<i>Transport</i>	372
<i>Restores from Backup</i>	372

<i>Log Files</i>	372
<i>Bogus Account to Access Customer Records</i>	372
<i>Credit Card Number Theft</i>	373
<i>DBA Password Theft</i>	373
HTTP Sessions	373

CHAPTER 30 Backup and Recovery Best Practices....377

Introduction	377
Deployment Architecture Overview	378
Infrastructure	378
Backup Strategies	380
<i>Database Recovery</i>	380
<i>Application Server and Web Server Recovery</i>	381

Part 3: Best Practices 383

CHAPTER 31 Database Management Best Practices..385

Introduction	385
Archiving Data	386
Monitoring Database Tables	386
<i>Key Tables To Monitor</i>	386
<i>Purging Data</i>	387
<i>Creating and Using History Tables</i>	387
Updating Statistics	388
<i>Updating Statistics For an Oracle Database</i>	389
<i>Updating Statistics For a SQL Server Database</i>	389

CHAPTER 32 JVM Tuning and Log Analysis.....391

Introduction	391
JVM Memory and Tuning Guidelines	391
<i>Adjusting JVM Memory Settings</i>	392
<i>Additional Performance Tuning</i>	393
<i>Tracing Garbage Collection Activities</i>	393
Log Analyzer Tool	394
<i>Setting Up Log Analyzer Daily Reports</i>	397

Daily Reports Workflow	397
Setting Up the Daily Reports	398
Recommended directory layout	398
Configuration	400
Part 4: Administration	403

CHAPTER 33 Introduction 405

Terminology	405
Using Storefronts	406
Storefront Administrator Tasks	406
Storefront Hierarchy	407
Skins	407
Storefront Data	408
Storefront Partners	409
Users, Roles, and Functions	409
Organizational Functions	410
Creating Users	410
Assigning Functions	411
Pre-defined Functions	412
Managers	414
User Statuses	414
Inheriting Status	414
User Preferences	415
Configuring the Visual Modeler	415
Site System Administration	416
Enterprise System Administration	416
Business Rules	416
Job Scheduling	416

CHAPTER 34 User Administration 419

Managing Users	419
Defining Functions and Roles	422

CHAPTER 35 Channel Administration.....423

Profile Detail Page..... 424

- Info Tab* 424
- Addresses Tab* 426
- Detail Tab* 426
- Business Tab* 427
- Hierarchy Tab* 428
- Commerce Tab* 428

Pricing Options 429

- Assigned To* 430
- Pricelists Tab* 430
- Product Entitlements Tab*..... 430
- Attributes Tab*..... 430
- Notes Tab* 430

Profile Administration Tasks 430

Storefront Administration 440

Managing the Enterprise Profile..... 441

- Info Tab* 441
- Commerce Tab* 441
- Current Accounts* 441

CHAPTER 36 Using the Visual Modeler.....443

Visual Modeler Interface 443

Working with Model Groups 448

Working with Models 453

Working with Option Classes and Option Items 461

Working with Option Class Groups and Option Item Groups..... 469

Including Sub-Models in Models 483

Special Characters Encoding 484

- Testing a Model 485
- Compiling a Model 486
- Searching the Product Catalog for a Product ID 488
- Working with a Tabbed User Interface 488

CHAPTER 37 Advanced Visual Modeler Concepts..... 491

Properties	492
<i>Working With Properties</i>	493
<i>Using Worksheets</i>	502
<i>Properties as Variables</i>	504
<i>Visual Modeler Properties</i>	505
Lists	506
<i>Working With Lists</i>	507
Rules.....	510
<i>Working With Rules</i>	510
<i>Rule Firing</i>	520
<i>Controlling Rule Firing</i>	522
Fragments	523
<i>Working With Rule Fragments</i>	523
Foreach 526	
<i>Working with Rule Actions</i>	537
Example Uses of Expand 544	
Option Constraints	545
<i>Working With Constraints</i>	545
Importing and Exporting Models	553
<i>Importing Model Groups and Models</i>	553
<i>Exporting Model Groups and Models</i>	554
Using Dynamic Instantiation	555
Searching.....	556
Reporting.....	558

CHAPTER 38 Visual Modeler UI Concepts 561

UI Properties	561
<i>Working with Display Properties</i>	562
<i>Visual Modeler UI Properties</i>	562
Display Properties	571
Tabular Display of Properties.....	576
Image Properties	579
<i>Models and Option Classes</i>	579
<i>Option Items</i>	579
User-Entered Values.....	580

UI Control Reset Behavior	581
CHAPTER 39 Enterprise System Administration	583
System Administration Tasks	583
Configuration Properties	584
<i>Locale Settings</i>	584
Job Scheduler Settings.....	585
Frequently Used System Administration Settings.....	586
<i>Commerce Manager</i>	586
Are comergent applications rendered as part of a frameset?	586
Availability Data Access Method	587
SMTP Host Machine	587
<i>Application Settings</i>	587
Allowed Decimal Places for displaying extended prices	587
Allowed Decimal Places for displaying list prices	588
Lines Per Page in List Displays	588
CHAPTER 40 Business Rules Administration	589
Business Rules Administration Tasks.....	589
CHAPTER 41 Job Scheduling Administration	591
Enterprise and Storefront Cron Jobs.....	591
Job Scheduling Tasks.....	592
Cron Jobs	595
<i>Cache Cleanup</i>	595
<i>Maintain Configuration</i>	596
CHAPTER 42 Site System Administration	597
Overview	597
System User Administration.....	598
System Profile Administration	599

System Property Administration	599
System Cron Jobs	599
System Status	600
Part 5: Tutorial	601

CHAPTER 43 Storefront Administration 603

Creating a Storefront	603
Creating a Storefront Administrator	605
Creating Additional Storefront Administrators	606
Setting Default Storefront Preferences	607
Setting Storefront Business Rules	608
Exercise	609
Creating a Storefront Partner	610
Creating a Storefront Partner Administrator	611
Creating a Storefront Partner User	612

CHAPTER 44 Creating Product Models..... 613

Create the Model	614
Properties	617
<i>Defining Properties for the MXWS-7700 Model</i>	<i>618</i>
<i>Attaching Properties</i>	<i>619</i>
Rules	620
<i>Creating a Rule</i>	<i>623</i>
<i>Using Rules to Control Display of Option Items</i>	<i>625</i>
<i>Creating a Constraint Table</i>	<i>629</i>
UI Controls	631
<i>Display Properties</i>	<i>631</i>
Pre-Pick Guiding Text	631
Ignore in Quote	632
<i>Tabular Displays</i>	<i>632</i>
<i>Calculated Property Values</i>	<i>633</i>
<i>User-Entered Values</i>	<i>636</i>
Restricting User Entered Values	638
<i>Images</i>	<i>639</i>

<i>Layout Management</i>	640
--------------------------------	-----

This chapter describes a checklist of tasks that you must perform to ensure that the Visual Modeler application is functioning appropriately.

Tasks Checklist

- Install the Selling and Fulfillment Foundation and perform the necessary configurations. For more information, refer to the *Selling and Fulfillment Foundation: Installation Guide*.
- Create products and define their details using the Business Center application. Alternatively, you can load the products using the data load functionality. For more information about creating products using the Business Center application, refer to the *Business Center: Item Administration Guide*.
- Create storefronts using the Applications Manager and perform storefront administration tasks. For more information about creating an organization, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*. For more information about using storefronts, refer to CHAPTER 33, "Introduction".
- Configure the models by performing the following tasks:

- Install the Visual Modeler application and integrate it with the Selling and Fulfillment Foundation to enable them to exchange information. For more information about integrating the Visual Modeler with the Selling and Fulfillment Foundation, refer to CHAPTER 14, "Integrating the Visual Modeler with Selling and Fulfillment Foundation".
- Configure the Selling and Fulfillment Foundation appropriately. For more information, refer to CHAPTER 14, "Integrating the Visual Modeler with Selling and Fulfillment Foundation".

Note that you must extract the sic_properties.zip file located in the <INSTALL-DIR>/repository/external folder and copy the .properties files to the same location defined in the where the properties file are located. configured in Applications Manager.

- In the Visual Modeler application, create a storefront with the same Skin ID as the Organization Code of the catalog organization defined in the Applications Manager. For more information, refer to CHAPTER 35, "Channel Administration".
- In the Visual Modeler application, create configuration models. While creating the models, associate the products to the option items appropriately.
- Compile and test the models.
- Create configurable products using the Business Center application and associate models with those products. For more information, refer to the *Business Center: Item Administration Guide*.
- Generate the search index. For more information about generating the search index, refer to the *Business Center: Item Administration Guide*.
- Install Sterling Web and configure it appropriately to enable a user to create a cart and place an order. For more information, refer to the *Sterling Selling and Fulfillment Suite: Applications Installation Guide* and the *Sterling Web: Implementation Guide*.
- Access the Sterling Web application and verify that orders can be placed and processed by performing the following tasks:
 - Browse the catalog to find a configurable product.
 - Configure the product and add the configuration to a cart.
 - Place an order for the cart.

- Check the order and shipment status.

Part 1:

Installation

The chapters in this section of the guide provide information required for you to install and implement the Visual Modeler at your enterprise.

Purpose

This guide provides step-by-step instructions to install and implement the Visual Modeler. On completion, you should be able to verify that the system is up and running, and that you can perform the basic administration tasks.

Audience

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

Conventions

Throughout this guide, we will use the following conventions shown in Table 1, "Conventions", on page 36:

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>

Architecture and Configuration Overview

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

Architecture

The Visual Modeler is designed to conform to the Java 2 Platform, Enterprise Edition (J2EE) architecture as defined in *Java 2 Platform Enterprise Edition Specification, v 1.2* published by Sun Microsystems, Inc. The Visual Modeler server architecture is illustrated schematically in "Logical Representation of the Visual Modeler Server Architecture" on page 38.

The Visual Modeler is deployed as a Web application that comprises a set of Java classes together with accompanying configuration files, HTML templates, and JSP pages. It must be installed into a servlet container that conforms to the J2EE standard. You can use an existing servlet container that conforms to the standard or deploy the Visual Modeler using the servlet container that we provide as part of the distribution software. See CHAPTER 5, "Installation Requirements" for further information.

The Visual Modeler is designed to conform to the Model 2 architecture. In this architecture, three functional components referred to as the Model, View, and

Controller (MVC) partition the functionality of the server into logically distinct components.

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

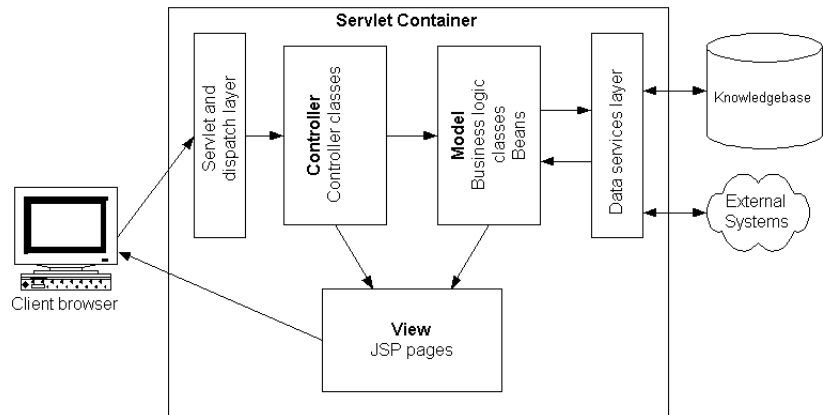


FIGURE 1. Logical Representation of the Visual Modeler Server Architecture

The Visual Modeler is designed to be flexible and extensible. You tailor the following components of the Visual Modeler as part of the implementation of your system.

TABLE 2. Implementation Components

Component	Function
JSP pages	Customize the JSP pages that determine the look and feel of the Web pages for end-users.
XML schema and data objects	Define the data object schema as a set of XML files. These specify the structure of the data objects and the data sources that provide their content.

TABLE 2. Implementation Components

Component	Function
Business logic and BizAPI classes	These Java classes determine the business logic that processes requests and messages.
Controller classes	These Java classes handle incoming requests from customer browsers and determine how the responses are displayed.
Configuration files	Use the configuration files to determine the properties of the Visual Modeler and control how incoming requests and messages are processed.

Implementation details are covered in the following chapters.

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

Implementation Tasks

This section describes a suggested methodology for implementing the Visual Modeler and an outline of the implementation steps.

Implementation Methodology

The Visual Modeler implementation methodology consists of phases that ensure that implementation can be planned and tracked through to completion. Table 3, "Visual Modeler Implementation Methodology", on page 42 provides a summary of the phases and the activities to complete in each phase.

A standard set of documents can be used to track each phase.

TABLE 3. Visual Modeler Implementation Methodology

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

Steps to Implementation

The main tasks you perform in implementing Visual Modeler are:

- *Project analysis*: agree to a schedule for the implementation project that sets a timeline. Identify milestones to measure the progress of the implementation and identify dependencies and risks that might prevent the implementation from completing on time.
- *Configuration analysis*: determine a suitable Visual Modeler configuration (the number of machines to be used and their location on internal networks in relation to firewalls and proxy servers). See "High Availability and Load Balancing" on page 45 for further details about a clustered implementation.
- *Integration analysis*: identify integration points with existing e-commerce systems.
- *Requirements analysis*: check hardware and software requirements to make sure that the machines are sufficiently powerful to support the anticipated traffic and response times required. See CHAPTER 5, "Installation Requirements" for more information.

- *Installation of Visual Modeler*: install the Visual Modeler on the designated machine(s). See CHAPTER 6, "Installing the Visual Modeler" for more information.
- *Knowledgebase setup*:
 - a. Installation of Knowledgebase: installing the Knowledgebase schema in the designated database server.
 - b. Knowledgebase setup: checking connectivity to the Knowledgebase database server and populating it with all your e-commerce-related information. This must include the partner profiles for your partners, your product catalog, and price list information.

See CHAPTER 7, "Creating and Populating the Knowledgebase" for more information.

- *Visual Modeler configuration*: modify configuration files to define the system configuration in your production environment.
- *Role and security definition*: define groups and roles and modify configuration files and ACL scripts accordingly. These determine the security privileges for your enterprise server users.
- *Schema creation*: create the business object schema to provide data source information. The data layer manages access between the enterprise server and the external systems.
- *Customizing BLCs and controllers*: modify business logic and controller classes to support your business logic. In some cases, you need to modify the Java classes in order to implement business processes specific to your organization.
- *Customizing JSP pages*: modify templates to meet your "look-and-feel", search, and static page requirements. The JSP pages provided by the Visual Modeler are used to display the browser pages and may be customized to meet the needs of your organization.
- *Product integration*: import product information into the Knowledgebase or provide punch-out integration. If your implementation is to support product ordering from a non-Sterling product, then you need to provide a means of integrating the product data with the Visual Modeler.
- *Testing server configuration*: before you deploy the Visual Modeler, thoroughly test the system. We provide a number of scripts to test the chief functional components.

- *Testing enterprise to partner communication*: send test messages from the enterprise server to other enterprise servers.
- *Testing partner to enterprise communication*: send test messages from other enterprise servers to your enterprise server.
- *Assess and enhance*: once the Visual Modeler is deployed you must plan for an ongoing process of analyzing its usage and performance.

Implementing the Visual Modeler Integration

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

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

TABLE 4. Implementation Tasks

Implementation phase	Task
Plan	Project analysis
Analyze	Configuration analysis
	Integration analysis
	Requirements analysis
Design and configure	Preparation of servlet container environment
	Installation of Sterling Commerce Manager
	Installation of Knowledgebase
	Knowledgebase setup
	Visual Modeler configuration
	Role and security definition
	System administrator authentication
	XML schema creation
	Customizing of BizAPIs, BLCs, and controllers
	Customizing JSP pages

TABLE 4. Implementation Tasks (Continued)

Implementation phase	Task
Testing and deployment	Product integration
	Testing server configuration
	Testing enterprise to partner communication
	Testing partner to enterprise communication
	Release to production systems
Improve	Assess and enhance

High Availability and Load Balancing

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

Integration Security Issues

Take special care to address security issues. Begin implementation only after you have addressed how users of the Visual Modeler will access data provided by you and your partners.

This discussion should cover:

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

See CHAPTER 29, "Security Best Practices" for more information.

This chapter presents a worksheet to help you gather the information that you need to install and configure the Visual Modeler.

Attention: If you do not have this information, then you will not be able to install and run the Visual Modeler.

- Which servlet container are you going to use for the Visual Modeler?
What release is this servlet container?

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

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

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

- What is the database server to be used for the Visual Modeler Knowledgebase?

- What JDBC URL will you use to connect to the Visual Modeler Knowledgebase database server?
 - You must connect to an Oracle Server using an Oracle JDBC driver.
 - You must connect to a Microsoft SQL Server using a Microsoft SQL Server JDBC driver.

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

- What name will you choose for the servlet context to be used for the Visual Modeler?

This chapter presents a description of the hardware, software, and network requirements to install the Visual Modeler. Make sure that your system meets these requirements before you begin installing the Visual Modeler. See:

- "Hardware Requirements" on page 49
- "Software Requirements" on page 50
- "Network Requirements" on page 52
- "Browser Requirements" on page 52
- "Database Server Requirements" on page 54

Pay special attention to performance requirements and what requirements that will drive in terms of hardware needs. See "Sizing Requirements" on page 57 for more information.

Hardware Requirements

This section provides a description of the minimum hardware requirements of the Visual Modeler.

Windows 2000

- 512 MB of RAM

- Single or dual Intel processors rated at 400 MHz or faster

UNIX

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

Software Requirements

This section provides a description of the software requirements of the Visual Modeler.

Operating Systems

Hewlett-Packard UNIX

- HP-UX 11.iv3.

Before deploying the Visual Modeler on HP-UX, you must apply the JavaOOB bundle provided by Hewlett-Packard. See the following URL for further information:

```
http://www.hp.com/products1/unix/java/java2/outofbox/infolibrary/  
release\_notes\_java\_oob.html
```

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

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

```
>ulimit -Sn 1028
```

Linux

- Red Hat Enterprise Linux 5.4.

Microsoft Windows

- Windows 2008 Server with Service Pack 1 or Windows 2000 Server or Professional with Service Pack 4.

Sun SPARC Solaris

- Sun SPARC Solaris 10 operating environment or subsequent compatible version.

Java Development Kit

You must use either JDK 6 or the most recent version of JDK 1.4.2. If you use JDK 1.4.2, your version must be at least 1.4.2_06; however, you should use the most current version. Problems have been reported using 1.4.2_03 because it uses incompatible keystores.

SDK

You must use JDK 6 to use the SDK to install the Visual Modeler and to create customizations using the SDK.

Servlet Containers

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

TABLE 5. Servlet Container Support

Servlet Container	Vendor	Release	Servlet Specification Support
Tomcat	Open Source	6.0.14 with JDK 6 On Windows installations, you must set the JVM used to the client jvm.dll	2.3
WebLogic	BEA Systems	10.3 with JDK 6	2.3

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

Servlet Container Clustering

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

See "High Availability and Clustering" on page 115 for more information on implementing a clustered solution. See CHAPTER 13, "Installing a Clustered Implementation" for more information about setting up clustered implementations:

- See "Setting Up a WebLogic Cluster" on page 203 for information relating to WebLogic clustering.

Network Requirements

The Visual Modeler machine(s) must also be able to establish a JDBC or an ODBC connection to the database server that is used in conjunction with the Visual Modeler. You must ensure that the appropriate database client software is installed on the Visual Modeler machine. See CHAPTER 6, "Installing the Visual Modeler" for more information.

Browser Requirements

To access and use the enterprise administration pages, users must run Internet Explorer 7 or subsequent compatible versions, or Firefox 3.5.x and subsequent compatible versions. This requirement includes partner users performing administrative tasks on the Visual Modeler.

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

Security Settings

You must enable your browser to support scripting.

Firefox

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

4. Click **OK**.

Internet Explorer

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

Mozilla

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

Character Sets

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

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

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

Database Server Requirements

This section lists the general set-up requirements for Oracle and SQL Server databases. See CHAPTER 6, "Installing the Visual Modeler", for details.

The Visual Modeler requires one of the following database servers to act as the Knowledgebase database:

- Oracle 11g
- Microsoft SQL Server 2005
- Microsoft SQL Server 2008

<p>Attention: We recommend that you run the database server on a separate machine from the Visual Modeler.</p>

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

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

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

Database Owner Requirements

The database owner must have the following privileges:

- CREATE TABLE
- CREATE VIEW
- CREATE SYNONYM

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

The database owner must have the following roles:

- CONNECT
- RESOURCE

Microsoft SQL Server Requirements

This section describes requirements for running the Visual Modeler with Microsoft SQL Server.

You must have a Microsoft SQL Server Release 2005 or subsequent compatible version running on Windows 2008 Server. You must connect to this SQL server using the JDBC drivers provided by Microsoft. You can download the JDBC drivers from: <http://msdn.microsoft.com/data/ref/jdbc/> (note that this URL is subject to change).

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

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

General Requirements

When you set up the SQL Server, you must specify that the database character set is Unicode. You must also set up the SQL Server client software on the Visual Modeler machine to use Unicode. On the servlet container machine:

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

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

SQL Server 2005 Communication Requirements

To enable communication between the transactional and segmentation databases on two separate servers, follow the steps specified in CHAPTER 6, "Installing the Visual Modeler", in the section "SQL Server 2005 Setup" on page 71.

Oracle Requirements

This section describes the requirements for running the Visual Modeler with Oracle 11g using the Oracle 11g JDBC driver.

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

<http://www.oracle.com/technology/index.html>

Search for "JDBC driver".

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

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

You should see:

```
NLS_CHARACTERSET UTF8
```

UNIX

To use the Oracle OCI driver to connect from the Visual Modeler to the database server, set the following environment variable:

```
NLS_LANG=AMERICAN_AMERICA.UTF8
```

Oracle Communication Requirements

To enable communication between the transactional and segmentation databases on two separate servers, follow the steps specified in CHAPTER 6, "Installing the Visual Modeler", in the section "Oracle Setup" on page 69.

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

See CHAPTER 6, "Installing the Visual Modeler", in the section "Oracle Setup" on page 69, for details.

Sizing Requirements

The Visual Modeler provides a sizing tool to help select hardware and software that will best meet your implementation needs.

This chapter describes how to install the Visual Modeler and deploy either the reference implementation or the minimal implementation. This chapter does not cover the customization work necessary to meet the needs of your implementation.

Most implementations of the Visual Modeler use the SDK to perform installation, and you must use the SDK to manage the customizations to your implementation. You can also install the reference Web application without using the SDK: this is useful for initial sanity testing and to verify the proposed deployment environment. See "Installing the Reference Visual Modeler" on page 91.

You can install Release 9.0 as a full install.

<p>Attention: The Visual Modeler is a complex product. Follow these instructions carefully.</p>
--

This chapter covers the following topics:

- "Installation Overview" on page 60
- "Preparing to Install" on page 61
- "Configuring the Transactional and Segmentation Databases" on page 64
- "Configuring the Transactional and Segmentation Databases" on page 64
- "Installing the Software Development Kit" on page 74

- "Installing the Visual Modeler Using the SDK" on page 75
- "Deploying the Sterling Web Application" on page 83
- "Matrix Reference Segments Setup" on page 91
- "Installing the Reference Visual Modeler" on page 91
- "Default XML Identity Setup" on page 102
- "Database Server Steps" on page 107
- "Managing Database Connections" on page 109
- "Pagination Settings" on page 112
- "Setting the Session Timeout" on page 113
- "Modifying the URL for the Web application DTD" on page 114
- "Managing Memory" on page 114
- "Configuring Ehcache" on page 115
- "High Availability and Clustering" on page 115
- "Sharing Directories" on page 115
- "Directory and File Organization" on page 116
- "Cron Job Setup" on page 118
- "Setting Up Apache as a Front-end to Tomcat" on page 119
- "Filtering Static Content" on page 123
- "Compressing Output From the Visual Modeler" on page 125

Installation Overview

Installing the Visual Modeler involves these stages:

- Preparing to Install
- Installing the Software Development Kit
- Installing the Visual Modeler Using the SDK
- Deploying the Sterling Web Application
- Database Server Steps

Once you complete these stages, continue to CHAPTER 7, "Creating and Populating the Knowledgebase".

Preparing to Install

This stage covers how to prepare for an efficient installation: identifying known issues, determining how you will configure your transactional and segmentation databases, identifying the database information you will need, identifying the servlet container root directory and the destination directory for the **Sterling.war** file, and so on. See "Preparing to Install" on page 61.

Installing the Software Development Kit

This stage covers how to install the Visual Modeler Software Development Kit (SDK). After you install the SDK, you can manage the installation of the Visual Modeler using targets provided by the SDK. See "Installing the Software Development Kit" on page 74.

Installing the Visual Modeler Using the SDK

This stage covers how to use the SDK to install the Visual Modeler Web application. After you complete this stage, you deploy the **Sterling.war** to a servlet container as a web application. See "Installing the Visual Modeler Using the SDK" on page 75.

Deploying the Sterling Web Application

This stage covers how to deploy the **Sterling.war** file as a Web application. How you deploy depends on which servlet container you are using. See "Deploying the Sterling Web Application" on page 83.

Database Server Steps

This stage covers the preliminary configuration steps you perform to prepare for the steps covered in CHAPTER 7, "Creating and Populating the Knowledgebase". The preparation steps require some knowledge of the database server that you plan to use for the Knowledgebase. See "Database Server Steps" on page 107.

Preparing to Install

1. Determine that your servlet container supports the Java Servlet Specification 2.3.

2. Determine the transactional and segmentation schema configurations. The system determines the configuration that you are using by examining properties that you set in the project properties file.

The out-of-the-box Visual Modeler supports segmentation using Oracle and SQL Server database servers.

See "Configuring the Transactional and Segmentation Databases" on page 64 for detailed information before proceeding.

3. Ensure that your databases and database servers are tuned appropriately for your implementation. Default settings may not always work, for example, you may have to increase settings such as the DEFAULT CURSORS setting. Consult your DBA to ensure proper database and database server tuning is complete before setting up the Visual Modeler application.
4. Determine the database connection information used to connect the Visual Modeler to the Knowledgebase database server:
 - a. For an Oracle database server:
 - Determine the configuration of your transactional and segmentation databases.
 - Set up access to the transactional database and to the segmentation database from the SDK machine: set up a TNS alias for the transactional database and, if you are using a separate server for the segmentation database, set up another TNS alias for the segmentation database.
 - To run the transactional and segmentation databases on separate servers, set up the link between the two database servers. You specify the name of the link as part of specifying database connection information.
 - b. For a SQL Server 2005 database server using JDBC to access the database:
 - Ensure that your SQL Server JDBC driver is at least version 1.1 or higher.
 - To run the transactional and segmentation databases as separate databases on separate servers, set up the link between the two database servers. The transactional database must have permission to access the segmentation database, and you must specify the transactional database server's login to the segmentation database server. For this configuration, you must also set up data replication between the two SQL Servers. See "Configuring the Transactional and Segmentation Databases" on page 64 for details.

- To run the transactional and segmentation databases as separate databases on the same SQL Server, the transactional database must have permission to access the segmentation database, including login permission.
5. You may add custom (or auxiliary) price types to your implementation.
 6. Identify any known issues.
 7. Identify the location of the servlet container root directory, *container_home*.
In a typical installation on Windows 2008, the location is as follows:

TABLE 6. Servlet Container Homes on Windows

Servlet Container	Home Location
Tomcat	C:\Program Files\Apache Software Foundation\Tomcat 6
WebLogic	C:\bea\weblogic103

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

TABLE 7. Servlet Container Homes on UNIX

Servlet Container	Home Location
Tomcat	/usr/local/tomcat/
WebLogic	/apps/bea/weblogic103/

8. Identify the destination directory location, *debs_home*, of the Visual Modeler. This is usually a sub-directory of *container_home*, but its precise location can vary from one servlet container to another.

In a typical installation on Windows 2008, the location is as follows:

Tomcat	C:\Program Files\Apache Software Foundation\Tomcat 6\webapps\
WebLogic	C:\bea\weblogic103\user_projects\domains\mydomain\applications\

9. Remove any existing deployment of the Visual Modeler Web application from the *debs_home* directory before starting the installation procedure. This requires using the servlet container's administrative console to remove the Web application, and then physically deleting the directories and files.
10. If you plan to implement Sterling Configurator, create an environment variable to specify the location of your JDK on the servlet container machine. For Windows systems, at the command line, enter:

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

For example:

```
set JDK_HOME=c:\jdk6
```

For UNIX systems, enter:

```
setenv JDK_HOME <path_to_jdk>
```

For example:

```
setenv JDK_HOME /usr/java/jdk6
```

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

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

Configuring the Transactional and Segmentation Databases

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

Segmentation is supported on Oracle and SQL Server 2005.

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

TABLE 8. Acceptable and Unacceptable Database Combinations

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

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

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

Database Configurations

The possible database configurations are:

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

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

Determining Configuration Type

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

Determining the Oracle Configuration

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

Determining the SQL Server 2005 Configuration

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

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

Oracle and SQL Server Properties

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

- ***DBTYPE_URL***: the location of the transactional database.
- ***DBTYPE_USERNAME***: the username for logging into the transactional database. This user is the owner of the database.

- **DBTYPE_PASSWORD:** the password for logging into the transactional database.
- **DBTYPE_SEGMENT_URL:** the location of the segmentation database.
- **DBTYPE_SEGMENT_USERNAME:** the username for logging into the segmentation database. This user is the owner of the database.
- **DBTYPE_SEGMENT PASSWORD:** the password for logging into the segmentation database.

SQL Server 2005 Replication Properties

Set the following replication properties in the *project_dev.properties* file. These properties are in addition to the properties listed in "Oracle and SQL Server Properties" on page 66.

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

- **MSSQLJDBC_SA_USERNAME:** the SQL Server system administrator user of the transactional database server for setting up replication. This is needed for setup only: you can delete this property after setup is complete.
- **MSSQLJDBC_SA_PASSWORD:** the SQL Server system administrator password of the transactional database server for setting up replication. This is needed for setup only: you can delete this property after setup is complete.
- **MSSQLJDBC_SA_DISTRIBUTOR_DATABASE:** a unique name for the SQL Server distribution database. Ensure that this name is unique and that this database does not already exist on the transactional database server.
- **MSSQLJDBC_SA_DATA_FOLDER:** the full pathname of the folder to contain replication data. This folder must already exist and reside on the transactional database server machine.

For Windows systems: The SDK considers the backslash character, "\", to indicate an Escape character. To specify a pathname such as C:\Replication, you must enter two backslashes. For example, C:\\Replication.

- **MSSQLJDBC_START_DATE:** the date on which to start data replication. The format is YYYYMMDD.

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

Note that the replication properties are required for all SQL Server 2005 transactional/segmentation database configurations. However, replication actually

occurs only for the separate databases/same server configuration or the separate databases/separate servers configuration.

SQL Server 2005 Migration Properties

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

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

When migrating data from an earlier release and using a configuration of two separate servers, the transactional server must be set up as a linked server on the segmentation server, see "SQL Server 2005 Setup - Migration with Two Servers" on page 72 for details.

Database Communication Setup

The following types of transactional and segmentation communication occurs:

- The transactional database pushes data to the segmentation database, which processes the data and resolves segment criteria into segment membership lists.
- The Visual Modeler Web application pulls data from the segmentation database for marketing activity purposes.
- The Visual Modeler Web application uploads third-party data to the segmentation database. The third-party data generally consists of files containing list of users and related information for segmentation and marketing purposes.

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

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

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

Oracle Setup

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

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

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

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

```
GRANT CREATE MATERIALIZED VIEW TO TRANSACTIONAL_DB_OWNER
GRANT CREATE PROCEDURE TO TRANSACTIONAL_DB_OWNER
GRANT CREATE SEQUENCE TO TRANSACTIONAL_DB_OWNER
GRANT CREATE SESSION TO TRANSACTIONAL_DB_OWNER
GRANT CREATE TABLE TO TRANSACTIONAL_DB_OWNER
GRANT CREATE TRIGGER TO TRANSACTIONAL_DB_OWNER
GRANT CREATE VIEW TO TRANSACTIONAL_DB_OWNER
GRANT UNLIMITED TABLESPACE TO TRANSACTIONAL_DB_OWNER
GRANT "CONNECT" TO TRANSACTIONAL_DB_OWNER
GRANT "RESOURCE" TO TRANSACTIONAL_DB_OWNER
```

3. To allow communication between the transactional database server and the segmentation database server:
 - a. Set up a TNSNAMES.ora entry on the transactional database server for the segmentation database server, then

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

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

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

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

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

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

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

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

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

The TNSNAME must be enclosed in single quotes.

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

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

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

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

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

SQL Server 2005 Setup

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

Note	If you are <i>migrating</i> from an earlier release and using a <i>two server configuration</i> , additional steps are required to establish communication between the two servers. See "SQL Server 2005 Setup - Migration with Two Servers" on page 72 for details
-------------	---

1. Set up the *project_dev.properties* file properties:

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

# SQL Server 2005 Segmentation DB properties file
# Tokenized files ODBCDataSources.xml/MSSQLSegCreateSchema.bat
MSSQLJDBC_SEGMENT_URL=jdbc:sqlserver:<full network address to SQL
Server instance>;DatabaseName=<dbname>
MSSQLJDBC_SEGMENT_USERNAME=<username>
MSSQLJDBC_SEGMENT_PASSWORD=<password>
MSSQLJDBC_SEGMENT_DATABASE=<dbname>
```

```
MSSQLJDBC_SEGMENT_SERVERNAME=<machine name only: not the network
address>

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

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

The MSSQLJDBC_SEGMENT_LINK and MSSQLJDBC_SCHEMA_NAME properties need to be set *only* when migrating data from an earlier release. Otherwise MSSQLJDBC_SEGMENT_LINK can be left blank and the default value for MSSQLJDBC_SCHEMA_NAME can be left in place. See "SQL Server 2005 Migration Properties" on page 68 for further information.

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

SQL Server 2005 Setup - Migration with Two Servers

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

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

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

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

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


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

```
USE [master]
GO
EXEC master.dbo.sp_addlinkedserver @rmtsrvname =
N'TRANSACTONAL_SERVER_NAME', @locallogin = N'SEGMENT_USER', @use-
self = N'False', @rmtuser = N'TRANSACTONAL_USER', @rmtpassword =
N'TRANSACTONAL_PASSWORD'
GO
```

4. Set up the *project_dev.properties* file properties:

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

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

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

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

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

The `MSSQLJDBC_SERVERNAME` property is used to specify the machine on which the SQL Server 2005 instance is running as well as to specify the name of the database server. In most cases the machine running the SQL Server 2005 instance

and the database server are the same, but they can be different. In your SDK environment, you must ensure that the machine name by itself is enough to enable the SQL client application sqlcmd to access the SQL Server 2005 instance, and that this is the name of the SQL server instance.

Installing the Software Development Kit

You can use the Software Development Kit (SDK) to install the Visual Modeler and to manage customizations to your implementation. You must use JDK 6 and Version 3.5.4 of the SDK to install the Visual Modeler.

This chapter covers only those SDK functions used to install the Visual Modeler. The Software Development Kit provides a comprehensive guide to the SDK.

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

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

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

For example:

```
set JAVA_HOME=c:\jdk6
```

For UNIX systems, enter:

```
setenv JAVA_HOME <path_to_jdk>
```

For example:

```
setenv JAVA_HOME /usr/java/jdk6
```

For the Bourne shell, enter:

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

For example:

```
export JAVA_HOME=/usr/java/jdk6
```

2. Set the COMERGENT.SDK_HOME environment variable to point to the *sdk_home* directory.
3. Version 3.5.4 of the SDK is delivered as a JAR file. Unjar the SDK framework JAR file in the *sdk_home* directory.
4. At the command line, navigate to the *sdk_home* directory, and enter:

```
sdk setup
```

5. For Windows systems: if your system directory is not **c:\winnt\system32**, edit the *sdk_home/my_sdk.properties* file and update the windows.system.dir property.

To continue with the installation, proceed to "Installing the Visual Modeler Using the SDK" on page 75.

Installing the Visual Modeler Using the SDK

This section describes how to use the SDK to install the Visual Modeler on either a Windows 2000/2008 or a UNIX operating system. Successful installation creates a **Sterling.war** file on your SDK machine.

Attention: Please install your servlet container before attempting to install the Visual Modeler.
--

Follow the steps described in "To Install the Visual Modeler as a Full Release" on page 75

Attention: If your release number is not Release 9.0 (for example, it is Release 9.0.2), then substitute the string for the release (for example, "8.0.2") wherever these instructions refer to "Release 9.0".

To Install the Visual Modeler as a Full Release

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

1. Locate the release **Sterling.jar** file for this release (called something like **Sterling-9.0-amberOne-9_0-RCx.jar**) and copy it to a temporary location on your system.
2. At the command line, navigate to the **sdk_home/** directory.
3. Edit the **sdk_home/my_sdk.properties** file to specify the value of the **container.home** and **app.name** properties. The SDK uses these properties to determine the values of other properties as follows:
 - **deploy.home** is set to **container.home/apps.dir**. The **deploy.home** property is used to specify the servlet container deployment directory.
 - The **app.name** is the name of the Web application. The **app.name** is usually the same as the Web application's directory under the deployment directory. In this guide, we assume that the value of this property is "Sterling". If you want to change the name of the Web application (for example, to change the name of the generated WAR file), modify the **app.name** entry in the **my_sdk.properties** file.

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

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

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

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

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

4. Run the `install` target twice, specifying the location of the **Sterling.jar** file set in Step 1. For example:

```
sdk install /tmp/Sterling-9.0-amberOne-9_0-RCx.jar
```

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

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

```
sdk newproject matrix
```

This target can take a few minutes to run.

6. The `newproject` target creates properties files for the new project in the `sdk_home/projects/project/` directory. The default name for the properties file is `project_dev.properties`. Note that the values of properties (such as `container.home`) set in the properties file override the values set in the `local-sdk.properties` and `my_sdk.properties` files.

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

You can also set other properties such as the logging level. Values you set here are automatically merged into the `prefs.xml` configuration files under the `sdk_home/builds/project/` directory. See "Email Addresses" on page 81 for information about the email addresses set in the properties files.

7. Database targets:
 - a. If you plan to run the Knowledgebase on Oracle, run the `installOracle` target, specifying the location of the Oracle JDBC JAR file, to copy the

Oracle JDBC drivers JAR file to the project files. The name and location of the JAR file will vary from installation to installation. For example:

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

or

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

For example:

```
sdk installOracle /tmp/Oracle_jdbc.jar
```

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

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

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

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

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

```
sdk env.setDBType Oracle
```

or

```
sdk env.setDBType MSSQLJDBC
```

9. To set the database password to be encrypted:

- a. Set the Encrypted flag to true:

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

- b. Encrypt the database password string:

```
sdk encryptVal <password>
```

The result is an encrypted version of the password.

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

```
ORACLE_PASSWORD=<encrypted_password>
```

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

10. If you plan to support locales other than the default U.S. English (en_US) locale, perform the localization installation and customization steps. See CHAPTER 24, "Visual Modeler Localization" for instructions.

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

```
sdk merge
```

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

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

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

```
sdk distWar
```

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

13. Alternatively, you can run the `dist` target. This creates a JAR file that contains the WAR file along with JAR files that provide the SQL scripts and XML data files. You can install the Visual Modeler from this JAR file by following the instructions provided in "Installing the Reference Visual Modeler" on page 91. Note that you can skip the steps to set the database properties information because your **prefs.xml** file already has this information.
14. In the `sdk_home/dist/time_stamp/` directory, rename the generated **prefs_env.xml** file to **prefs.xml** file. This is the basic configuration file that must be copied under the home directory of the user running the servlet container.

15. Add any custom (auxiliary) price types to your implementation.

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

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

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

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

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

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

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

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

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

The format is as follows:

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

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

```
</PriceType>
</PriceTypeList>
</PriceTypeListData>
```

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

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

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

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

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

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

- f. Merge the customized files into the project:

```
sdk merge
```

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

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

```
sdk createDB
```

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

- b. Check the results of the createDB target by looking at log files in the *sdk_home/logs/projects/project_name* directory.
17. Run either the loadDB target (to load the minimal data set) or the loadMatrixDB target (to load the full Matrix reference data set) into the Knowledgebase.


```
sdk loadDB
```

or

```
sdk loadMatrixDB
```

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

Check for an error similar to the following:

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

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

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

18. Create the segmentation database:

```
sdk createSegDB
```

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

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

```
sdk loadSegDB
```

or

```
sdk loadSegMatrixDB
```

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

Next, deploy the Web application into the servlet container. Follow the steps for your servlet container provided in "Deploying the Sterling Web Application" on page 83.

Email Addresses

As part of implementing the Visual Modeler, set up the email addresses used by the system. They reside in one of the following locations:

- Configuration Files
- Minimal Data

Configuration Files

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

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

Minimal Data

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

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

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

ObjectMap Settings

Sterling Configurator uses rules to determine some of the behavior of models used to configure products. These rules are compiled into Java classes when the rule is first fired. The rules may be compiled using either an external compiler (that is, using javac) or an internal compiler (that is, the com.sun.tools.javac.Main class). An element of the **ObjectMap.xml** configuration file is used to specify the one you

want to use. Bear in mind that there are differences between using these two compilers as described here:

- external compiler: this is spawned as a new process. On UNIX, this can cause the creation of a copy of the current process running the Visual Modeler and so may require a large memory allocation, and fail if this is not available.
- internal compiler: this is generally faster, but may be prone to memory leaks.

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

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

to:

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

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

Deploying the Sterling Web Application

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

You must identify the operating system user that runs the servlet container. You must copy the **prefs.xml** configuration file created in Step 12 on Page 78 to a sub-

directory of the home directory of this user. The sub-directory is called ***user_home/cmgt/debs/conf/***.

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

TABLE 9. Operating System Home Directories

Operating System	Standard Home Directory
Windows 2008	C:\Documents and Settings\ <i>username</i>
Solaris 10	/export/home/ <i>username</i>
Linux 2.6	/home/ <i>username</i>

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

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

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

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

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

We provide deployment steps for the following servlet containers:

- "To Deploy the Sterling Web Application on Apache Tomcat" on page 85
- "To Deploy the Sterling Web Application on WebLogic 10.3" on page 88

XML Parser Settings

To ensure that the correct Java classes are used for the XML processing performed by the Visual Modeler, you must ensure that the Java Virtual Machine settings specify the correct classes. In general, you can either set the classes as additional

parameters in the command line that starts the servlet container or you can specify them as parameters for the Web application.

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

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

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

Tomcat Releases

To Deploy the Sterling Web Application on Apache Tomcat

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

1. Ensure that your JDK **bin** directory is defined in the system PATH environment variable. For example:
C:\Program Files\Java\jdk1.6.0_08\bin
2. On Windows installations of Tomcat, you must use the client version of the Java VM DLL. Open **Start -> All programs -> Apache Tomcat 6.0 -> Configure Tomcat**, and on the Java tab, set the Java Virtual Machine to the location of the client JVM DLL (for example:
C:\Program Files\Java\jdk1.6.0_07\jre\bin\client\jvm.dll).
3. Modify the Tomcat startup parameters to set the following Java parameters:
 - -Xms128m

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

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

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

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

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

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

- b. If you are running Tomcat as a service, set the Java parameters as part of the Tomcat service configuration. Open **Start -> All programs -> Apache Tomcat 6.0 -> Configure Tomcat**, then click the **Java** tab. The fields map as follows:
 - Initial memory pool: `-Xms`
 - Maximum memory pool: `-Xmx`
4. On Windows systems with Tomcat installed as a service, you must set the login information through the service. To ensure that the Visual Modeler works correctly and locates the correct **prefs.xml** file, set the login to the user that owns **prefs.xml**.

For example, if the **prefs.xml** file is located in:

```
C:\Documents and Settings\adminuser.DOMAIN\cmgt\debs\conf
```

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

- Login: `DOMAIN\adminuser`
- Password: adminuser's password

Notes on Using Apache Tomcat

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

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

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

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

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

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

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

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

Continue with the steps described in "Database Server Steps" on page 107.

WebLogic Releases

Deployment Considerations For WebLogic 10

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

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

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

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

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

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

If you are running the Selling and Fulfillment Foundation in HTTPS mode, ensure that you pass the following argument when you run the target to start the WebLogic server:

```
-DlocalPost.ssl.noCheck=true
```

To Deploy the Sterling Web Application on WebLogic 10.3

Deployment of the Visual Modeler into WebLogic Release 10.3 has been simplified from earlier releases of WebLogic. However, because the Visual Modeler must run as an “expanded” Web application (as opposed to as a WAR file), these instructions ensure that the Web application WAR file is expanded as part of the deployment process.

1. In your WebLogic installation, identify the *domain_home* directory that you plan to use for your deployment of the Visual Modeler. The default location is *container_home/user_projects/domains/mydomain/*.
2. In the *container_home/user_projects/domains/mydomain/* directory, create a directory in which to deploy your applications called **applications**. In the **applications** directory, create the directory in which to deploy your Visual Modeler application, **Sterling**.
3. Expand the Sterling.war file into the *container_home/user_projects/domains/mydomain/applications/Sterling* directory using a tool such as WinZip. This process can take a few minutes. Verify that the Sterling directory structure is in place as you expand the Sterling.war file.
4. Log in to the WebLogic server as the user used to install WebLogic and start the BEA WebLogic Administration Console.
5. Install the Visual Modeler as a WebLogic application:
 - a. In the Domain Configurations section, click Deployments, then click the Lock & Edit button in the left-hand panel.
 - b. In Deployments, click the Install button. Using the Install Applications Assistant, locate the *container_home/user_projects/domains/mydomain/applications* directory, select Sterling, and click Next.

- c. Use the Install Applications Assistant to configure your Sterling application, or accept the defaults and click Finish.
 - d. In the left-hand panel, click the Activate Changes button. The Sterling application appears in the Deployments list with a State of Prepared.
6. Click Start to start the Sterling application and choose Servicing All Requests. The application's State will update to Start/Running.
 7. Your Sterling application is now installed. Click the Release Configuration button.
 8. Verify that you can log into the Visual Modeler by pointing your browser to the standard URL:

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

For WebLogic servers, the default port number is 7001.

Pre-Compiling JSP Pages

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

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

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

```
        <param-value>
            Comergerent_jsp
        </param-value>
    </jsp-param>
</jsp-descriptor>
</weblogic-web-app>
```

XML Parsing

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

1. Add the following:

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

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

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

Continue with the steps described in "Database Server Steps" on page 107.

Solaris and Oracle OCI Driver

Note that if you are using an Oracle database server for the Knowledgebase, then you can use the Oracle OCI JDBC driver to connect from the Visual Modeler to the Oracle database server. See "Support for Oracle Server" on page 107 for further information.

Further Deployment Steps

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

- "Database Server Steps" on page 107
- "Setting the Session Timeout" on page 113

- "Modifying the URL for the Web application DTD" on page 114
- "Managing Memory" on page 114
- "High Availability and Clustering" on page 115

Matrix Reference Segments Setup

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

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

1. Log in to the Visual Modeler as an admin user with access to the job scheduler.
2. Click **Job Scheduler** in the System Administration panel on the Visual Modeler home page.
3. Click the Nightly Segments Build cron job.
The details of the cron job display.
4. Start the Nightly Segments Build cron job.

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

Installing the Reference Visual Modeler

This section describes the steps to install Visual Modeler Release 9.0 without using the SDK. This provides you with a relatively quick verification of the basic deployment of our reference Web application. Before you start the process of customizing the Visual Modeler, install the Visual Modeler into the SDK and work in that environment to create your customized deployment.

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

These instructions use three logically distinct machines: the developer machine, the servlet container machine, and the database server machine. Depending on your

situation, these machines may actually all be the same physical machine or different ones: we identify what steps are performed on which logical machine.

Make sure that you have the Java Development Kit (JDK) 6 installed on all three machines. The servlet container machine and the database server machine should have the database client tools installed to connect to the database server. You will need to know the database connection information required to connect from the servlet container machine to the database server. In addition, you will need to know how to configure your database setup to take advantage of the segmentation feature. See "Configuring the Transactional and Segmentation Databases" on page 64 for more information before proceeding with the reference installation.

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

```
jar -xvf Sterling-Aries.jar
```

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

This unpacks the release JAR file and creates several sub-directories under *cmgt_home*.

4. You must now set the values of system properties that the Web application will need, such as the location of the database. At the command line, execute:

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

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

5. The initial settings dialog box displays.

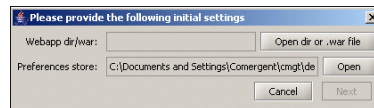


FIGURE 2. Initial Settings Dialog Box

6. Click the Open dir or .war file button to navigate to the release WAR file, then select the file named Sterling-Aries-def-RC-1.war (or something very similar

to this). The release WAR file name appears in the File Name field. Click Open. The Preferences Store Open File dialog box should open up in the correct directory and you should be able to leave the value of the Preferences store with its current value.

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

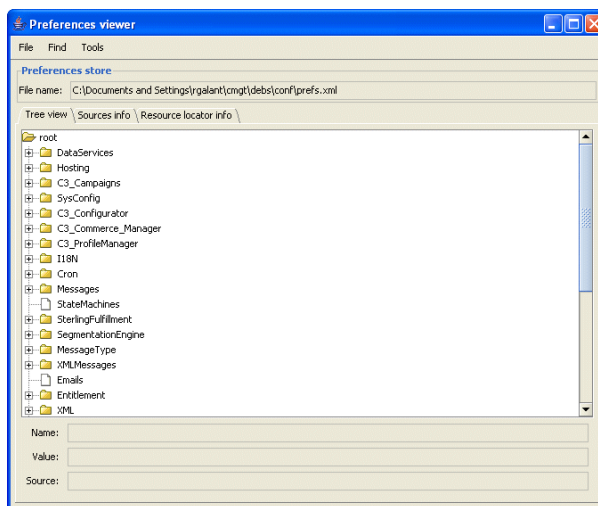


FIGURE 3. Preferences Viewer Window

8. Using this window, set the values for the following properties as follows:
 - a. Navigate to the `DataServices.DataSource.ENTERPRISE.ConnectionString` property. You should see the lower property panel display the name of the property, its current value, and where the value is stored:

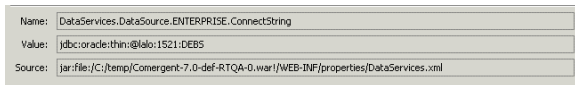


FIGURE 4. Property Panel

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

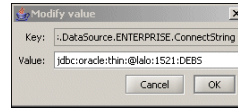


FIGURE 5. Modify Value Dialog Box

- c. In the Modify Value dialog box, enter the correct value for the connection string property: this is the URL used by the data services layer to connect to the Knowledgebase database server. The form of this URL depends on the database type as follows:
 - For Oracle: “jdbc:oracle:thin:@<machine>:1521:<sid>”
 - For SQL Server: “jdbc:sqlserver://<sqldb_name>; DatabaseName=<dbname>”
- d. Repeat these steps for the following properties:
 - DataServices.DataSource.ENTERPRISE.DataService:
 - For Oracle: “JdbcService”
 - For SQL Server: “JdbcService”
 - DataServices.DataSource.ENTERPRISE.SrvSubType:
 - For Oracle: “ORACLE”
 - For SQL Server: “MS”
 - DataServices.DataSource.ENTERPRISE.UserId
 - DataServices.DataSource.ENTERPRISE.Password
 - DataServices.DataSource.SEGMENT.ConnectionString:
 - For Oracle: “jdbc:oracle:thin:@<machine>:1521:<sid>”
 - For SQL Server: “jdbc:sqlserver://<sqldb_name>; DatabaseName=<dbname>”
 - DataServices.DataSource.SEGMENT.Dataservice:
 - For Oracle: “JdbcService”
 - For SQL Server: “JdbcService”
 - DataServices.DataSource.SEGMENT.SrvSubType:

- For Oracle: “ORACLE”
 - For SQL Server: “MS”
 - DataServices.DataSource.SEGMENT.UserId
 - DataServices.DataSource.SEGMENT.Password
 - DataServices.General.JdbcDriver1:
 - For Oracle: “oracle.jdbc.driver.OracleDriver”
 - For SQL Server: “com.microsoft.sqlserver.jdbc.SQLServerDriver”
 - DataServices.General.DsKeyGenerators: set to the database-specific value:
 - For Oracle: “OracleKeyGenerators.xml”
 - For SQL Server: “MSSQLJDBCKeyGenerators.xml”
 - You can set any other properties that you wish using the Preferences Editor, but these should be enough to get your reference deployment up and running.
9. When you have finished setting property values, click **File** -> **Save**.
 10. Click **File** -> **Exit**.

The new property values will be saved to the file: **cmgt/debs/conf/prefs.xml** in your home directory, referred to as *user_home* (for example: **C:\Documents and Settings\username** or **/export/home/username/**).
 11. If the database server is inaccessible from your current network location, then transfer the appropriate sql-data file (depending on your DB_TYPE) to a temporary location, *cmgt_data_home*, on a machine that can access the database server machine:
 - **cmgt_home/sql/MSSQLJDBCsql-data-Aries-def-RC-1.jar**
 - or
 - **cmgt_home/sql/Oraclesql-data-Aries-def-RC-1.jar**
 12. On this machine, at the command line navigate to *cmgt_data_home*.
 13. Unpack the JAR file by executing:

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

14. Set up your database users, privileges, and, if you are using an Oracle database server, the Oracle database link. See "Configuring the Transactional and Segmentation Databases" on page 64 for details.
15. If you are using an Oracle database server, edit the following files:
 - ***cmgt_data_home/WEB-INF/sql/Oracle/setup/oracle_indexes.sql*** file: change the value of `@ORACLE_INDEX_TABLESPACE@` name, if necessary, to `TABLESPACE <tablespace_name>`. The `TABLESPACE` name specifies the location of your Oracle database server's index files.
 - ***cmgt_data_home/WEB-INF/sql/Oracle/setup/Oracle_privileges.sql*** file: replace `@ORACLE_USERNAME@` with the name of the transactional database user name and `@ORACLE_SEGMENT_USERNAME@` with the segmentation database user name. If your configuration is transactional and segmentation databases within the same schema on the same server, you can leave `@ORACLE_SEGMENT_USERNAME@` blank.
 - ***cmgt_data_home/WEB-INF/sql/Oracle/segment/oracle_mvviews1.sql*** file: replace `@ORACLE_USERNAME@` with the transactional database user name and `@ORACLE_SEGMENT_USERNAME@` with the segmentation database user name. If your configuration is transactional and segmentation databases within the same schema on the same server, you can leave `@ORACLE_SEGMENT_USERNAME@` blank.
 - ***cmgt_data_home/WEB-INF/sql/Oracle/segment/oracle_mvviews.sql*** file: If your configuration is transactional and segmentation schema within the same server (either in the same schema or separate schema), you can skip this step. Otherwise, replace `@ORACLE_SEGMENT_LINK@` with the name of the Oracle database link that was created on the segmentation database to access the transactional database.
16. If you are using a SQL Server 2005 database server, edit the following files:
 - ***cmgt_data_home/WEB-INF/sql/MSSql/setup/mssql_create_replication.sql*** file: replace the following variables with appropriate values. See "SQL Server 2005 Setup" on page 71 for details.
 - `@MSSQLJDBC_SERVERNAME@`: This is the actual name of the server machine, not the network name
 - `@MSSQLJDBC_SA_PASSWORD@`
 - `@MSSQLJDBC_DISTRIBUTOR_DATABASE@`

- @MSSQLJDBC_DISTRIBUTOR_DATA_FOLDER@
 - @MSSQLJDBC_DATABASE@
 - @MSSQLJDBC_SEGMENT_SERVERNAME@: This is the actual name of the server machine, not the network name.
 - @MSSQLJDBC_SEGMENT_DATABASE@
 - @MSSQLJDBC_SEGMENT_USERNAME@
 - @MSSQLJDBC_SEGMENT_PASSWORD@
 - @MSSQLJDBC_START_DATE@
- *cmgt_data_home/WEB-INF/sql/MSSql/setup/mssql_drop_replication.sql* file: replace the following variables with appropriate values. See "SQL Server 2005 Setup" on page 71 for details.
 - @MSSQLJDBC_SA_PASSWORD@
 - @MSSQLJDBC_DISTRIBUTOR_DATABASE@
 - @MSSQLJDBC_DISTRIBUTOR_DATA_FOLDER@
 - @MSSQLJDBC_DATABASE@
 - @MSSQLJDBC_SEGMENT_SERVERNAME@: This is the actual name of the server machine, not the network name.
 - @MSSQLJDBC_SEGMENT_DATABASE@
 - @MSSQLJDBC_SEGMENT_USERNAME@
 - @MSSQLJDBC_SEGMENT_PASSWORD@
17. Edit the following batch scripts to add your connection information. The scripts reside in *cmgt_data_home*.
- For Oracle database servers:
 - **OracleCreateSchema.bat** or **OracleCreateSchema.sh**
 - **OracleCreateSegmentSchema.bat** or **OracleCreateSegmentSchema.sh**
 - If the transactional and segmentation databases are on separate Oracle servers (the database URL's have different values), edit **OracleCreateSegmentSchema1.bat** or **OracleCreateSegmentSchema1.sh**

- If the transaction and segmentation databases are on the same Oracle server (the database URL's have the same values) and their user names are different, edit **OracleCreateSegmentSchema2.bat** or **OracleCreateSegmentSchema2.sh**
 - **OracleCreateSegmentSchema3.bat** or **OracleCreateSegmentSchema3.sh**
 - For SQL Server 2005 database servers:
 - **MSSQLJDBCCreateSchema.bat**, **MSSQLJDBCCreateSegmentSchema.bat**
 - If the transactional and segmentation databases are on separate SQL Servers or different databases on the same server (the database URL's have different values), edit **MSSQLJDBCCreateSegmentSchema1.bat**.
 - **MSSQLJDBCCreateSegmentSchema3.bat**
18. Run the following scripts in the order specified:
- Oracle:
 - a. **OracleCreateSchema.bat** or **OracleCreateSchema.sh**
 - b. **OracleCreateSegmentSchema.bat** or **OracleCreateSegmentSchema.sh**
You must use the TNS alias name of the Oracle database server as seen from the current machine.
 - SQL Server 2005:
 - a. **MSSQLJDBCCreateSchema.bat**
 - b. **MSSQLJDBCCreateSegmentSchema.bat**
You must use the ODBC source name of the SQL Server database as seen from the current machine.
19. Run the appropriate scripts for your transactional/segmentation database configuration:
- For Oracle:
 - If the transactional and segmentation databases are on separate Oracle servers (the database URL's have different values), run **OracleCreateSegmentSchema1.bat** or **OracleCreateSegmentSchema1.sh**

- If the transaction and segmentation databases are on the same Oracle server (the database URL's have the same values) and their user names are different, run **OracleCreateSegmentSchema2.bat** or **OracleCreateSegmentSchema2.sh**
 - Always run **OracleCreateSegmentSchema3.bat** or **OracleCreateSegmentSchema3.sh**
 - For SQL Server 2005:
 - If the transactional and segmentation databases are on separate SQL Servers (the database URL's have different values), or the transactional and segmentation databases are on the same SQL Server but in two different databases, run **MSSQLJDBCCreateSegmentSchema1.bat**.
 - Always run **MSSQLJDBCCreateSegmentSchema3.bat** or **MSSQLJDBCCreateSegmentSchema3.sh**.
20. Copy the following files to a temporary location, *cmgt_app_home*, on the servlet container machine:
- *cmgt_home*/Sterling-Aries-def-RC-1.war
 - *user_home*/cmgt/debs/conf/prefs.xml
 - *cmgt_home*/data/Sterling-xmldata-Aries-def-RC-1.jar
 - *cmgt_home*/data/cmgt-xmlloader-tool.jar
 - *cmgt_home*/install/cmgt-cryptography-tool.jar
 - *cmgt_home*/install/cmgt-jspResource.jar
 - *cmgt_home*/install/xmlClient-tool.jar
21. On the servlet container machine, copy the *cmgt_app_home*/prefs.xml file to the following directory (which you may have to create): *user_home*/cmgt/debs/conf/prefs.xml. Note that this should be the *user_home* for the user that is used to run the servlet container.
22. If you are running against Oracle or SQL Server 2005, then install the appropriate JDBC JAR file into the servlet container so that it can be used by the Visual Modeler web application when deployed. In the Tomcat Application Server, install these JAR files by placing them in the *container_home*/common/endorsed/ directory. In WebLogic application servers, this step is probably unnecessary, since WebLogic servers are deployed with an extensive collection of JDBC clients.

23. Make sure that you are logged in as the user who is running the application server, and at the command line, navigate to *cmgt_app_home*.

24. Unpack the **Sterling-xmldata-Aries-def-RC-1.jar** file by executing:

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

25. Add any custom (auxiliary) price types to your implementation. The following lists the general steps.

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

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

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

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

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

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

</PriceTypeList>
</PriceTypeListData>
```

Where *price_type_code* is the unique numeric code associated with the price type and which matches the lookup code, such as 1000, 2000, or 3000; *group_code* is the price type group code with which this price type is associated, such as 20 for one-time prices; *property_name* is the

name of the price type property, is uppercase, and always begins with PRICE:, such as PRICE: ACTIVATION; and the *UpdateDate* and *CreateDate* dates are timestamps.

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

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

26. Load the data.

- a. On UNIX, run:

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

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

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

```
SELINUX=disabled
```

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

```
loadDBFromXML full
```

- c. For Windows configurations with JDBC, run:

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

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

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

27. Load the full segmentation data:

- a. On UNIX, run:

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

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

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

```
SELINUX=disabled
```

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

```
loadSegDBFromXML full
```

- c. For Windows configurations with JDBC, run:

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

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

28. Check the results by examining the log files in the directory in which you ran the data load command.
29. Rename the `cmgt_app_home/Sterling-Aries-def-RC-1.war` file to **Sterling.war** and deploy it to the servlet container using the steps described in "Deploying the Sterling Web Application" on page 61.
30. Restart the application server.
31. You should now be able to point your browser to the standard Visual Modeler URL and log in as the enterprise administrator user admin/admin.

Default XML Identity Setup

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

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

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

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

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

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

The general steps are:

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

You set up the default XML identity from the Visual Modeler UI as the storefront administrator. You edit the **web.xml** configuration file to configure the trusted IP addresses.

Default XML Identity Configuration

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

1. Log in to your storefront as a storefront administrator.
2. Create a new storefront user of type Default XML Identity.
3. From the System Services page, set the following XML Messages properties:
 - Set *Should the system enable a default xml user identity?* to true. This sets the system property XMLMessages.EnableDefaultXMLIdentity to true. The default value is false.
 - Enter a user name in the *Username of default identity for XML messages* field. This sets the default XML identity user name in the system property XMLMessages.DefaultXMLIdentity_UserName. The default value is null.
4. Log out and restart the server. This allows the system to recognize the new default XML identity and user name.

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

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

The minimal data contains a default XML user: XMLGuest.

Trusted IP Address Configuration

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

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

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

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

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


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

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

Default XML Identity Request Authentication

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

The following diagram illustrates the authentication sequence.

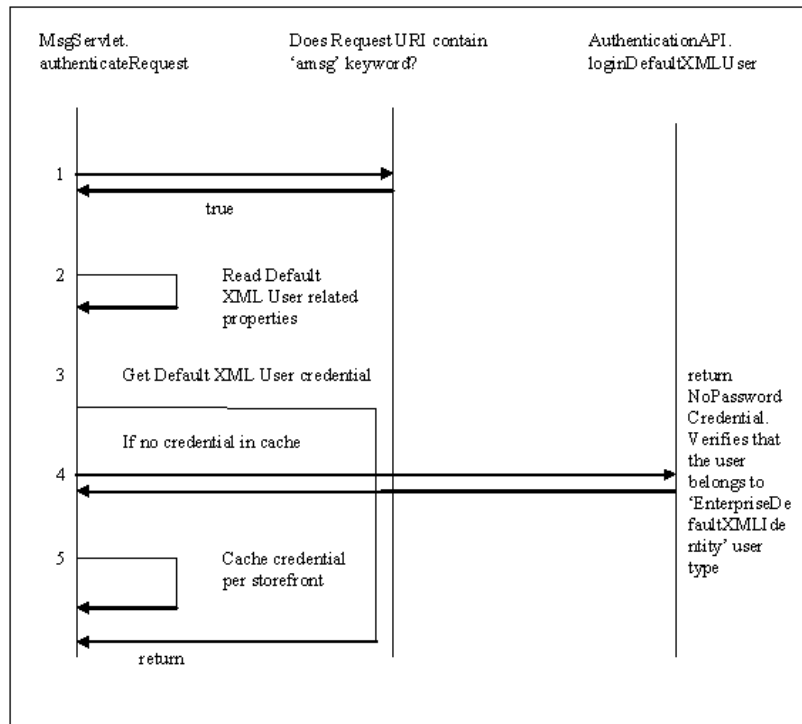


FIGURE 6. Authentication Sequence

The authentication sequence is as follows:

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

AuthenticationAPI

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

Methods:

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

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

Database Server Steps

Depending on which database server you use with the Visual Modeler, perform the required steps in this section. See "Managing Database Connections" on page 109 for information about connection pooling.

Support for Oracle Server

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

If you plan to use the OCI driver to connect from the Visual Modeler machine to the Oracle Server, then you must make sure that the OCI driver is set up correctly. On a UNIX system:

1. Make sure that the OCI **liboci*jdbc.so** file is installed on the Visual Modeler machine. The "*" in the name of the file is the version number of the OCI library.
2. Make sure that the servlet container scripts ensure that the LD_LIBRARY_PATH includes the location of the OCI **liboci*jdbc.so** file and that ORACLE_HOME is set to point to the location of the Oracle client tools.

3. Make sure that the Oracle JAR file matches the version of the OCI library file:

TABLE 10. OCI library and JDBC Driver Files

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

Support for SQL Server

If you are running the Visual Modeler on a Windows system and plan to use SQL Server 2003 for your database server, then you must perform the following steps.

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

Set LowerCase to "LOWER(" and UpperCase to "UPPER("; for example:

```
<UpperCase controlType="text" runtimeDisplayed="true"
ChangeOnlyAtBootTime="true" visible="true" boxsize="45"
displayQuestion="UpperCase SQL Function" defaultChoice=""
help="Enter the SQL function that converts strings to uppercase
for the selected database.">UPPER(</UpperCase>
```

Note that the use of the UPPER and LOWER functions in searches means that indexes on the tables are not used and this can result in reduced performance.

3. If more than one deployment of the Visual Modeler is accessing the same Knowledgebase on SQL Server, then you must set a two-digit server ID for each deployment.
 - a. If the machines are not clustered, then set the ServerId element in the **prefs.xml** file so that each has a unique integer value: 01, 02, and so on.
 - b. If the machines are clustered, then you must modify the servlet container command or script that starts the servlet container on each machine so that

a Java system property is set: `Comergent.DataServices.General.ServerId`. This should be set on each machine so that each has a unique value: 01, 02, and so on.

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

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

Managing Database Connections

This section describes how to manage the connections between the Visual Modeler and the Knowledgebase.

Configuration Files

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

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

Connection Pooling

This section answers some common questions about connection pooling.

- What is the purpose of connection pooling?
- How does Sterling's connection pooling work?
- Why are there separate query and update connection pools?
- How do I validate connections prior to reuse?
- How can I limit the number of connections used?
- How can I free up connections when demand drops?
- What happens when connection limits are reached?
- Why are the connection limits on the data source?

Common Problems

- My database requests fail with a “connection reset by peer” message
- My database connections are not being released when traffic drops
- I share a database with other applications. I cannot allow the Visual Modeler to use more than n connections

What is the purpose of connection pooling?

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

How does Sterling’s connection pooling work?

The Visual Modeler maintains these logical pools of connections:

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

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

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

Why are there separate query and update connection pools?

The use of separate pools for Query and Update Connections allows the Visual Modeler to optimize connections for read-only access.

How do I validate connections prior to reuse?

The following properties control connection timeout and validation:

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

How can I limit the number of connections used?

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

How can I free up connections when demand drops?

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

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

What happens when connection limits are reached?

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

Why are the connection limits on the data source?

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

Common Problems

My database requests fail with a “connection reset by peer” message

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

My database connections are not being released when traffic drops

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

I share a database with other applications. I cannot allow the Visual Modeler to use more than n connections

Setting the `MaxConnections` property on the data source will allow you to limit the maximum number of database connections used by the Visual Modeler.

Pagination Settings

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

If your implementation of the Visual Modeler uses a cluster of servlet containers, then the location of the pagination directory must be accessible to all members of the cluster. See "High Availability and Clustering" on page 115.

Setting the Session Timeout

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

You can set the session timeout period in the Visual Modeler **web.xml** configuration file using the `session-timeout` element. For example, to timeout sessions after 30 minutes, set the element to:

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

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

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

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

Modifying the URL for the Web application DTD

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

By default, the validating DTD is at the URL:

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

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

Our suggested solution is to use a relative URL to reference the DTD within the Visual Modeler context. For example:

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

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

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

Managing Memory

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

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

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

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

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

Configuring Ehcache

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

High Availability and Clustering

The Visual Modeler can be deployed in a distributed environment in which more than one individual instances of the Visual Modeler run as a cluster. This provides support for ensuring high availability of the Visual Modeler and to support fail-over of individual machines. See CHAPTER 13, "Installing a Clustered Implementation" for more information.

Sharing Directories

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

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

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

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

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

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

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

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

```
/DEBS_shared
```

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

```
/DEBS_shared/lw
```

and set that value in the configuration file. For example:

```
<loadable ...>/DEBS_shared/lw</loadable>
```

Directory and File Organization

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

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

- A locale directory for each supported locale. Each locale may be expressed as `<la>_<CO>`, where *la* is one of the standard language codes and *CO* is one of the standard country codes, for example: `en_US` or `fr_CA`. For a locale, the corresponding directory is ***debs_home/Sterling/la/CO***. This directory contains:
 - **css/**: holds the cascading style sheets used by the Visual Modeler.
 - **images/**: holds common images used by the Visual Modeler.
 - **js/**: holds Javascript libraries used by the Visual Modeler.
 - **htdocs/**: holds the HTML templates, images, and online help for the Visual Modeler.
- **dXML/**: holds the DTDs for the dXML message types.
- **htdocs/**: a directory for content that can be served up directly by the servlet container or Web server. Content stored here should not be locale-specific.
- **j2ee/**: a directory to hold local copies of the J2EE DTDs. See "Modifying the URL for the Web application DTD" on page 114 for more information.
- **WEB-INF/**: holds all the configuration files used by the server. It contains the following subdirectories:
 - **bizobjjs/**: holds the business object DTDs. These DTDs are used to validate XML messages. The DTDs can be generated automatically by the generateDTD target provided by the SDK.
 - **classes/**: holds the Visual Modeler Java classes.
 - **commerceone/**: used as part of Commerce One integration.
 - **converters/**: holds the configuration files used in message conversion.
 - **data/**: holds data provided as part of the reference implementation.
 - **extralib/**: holds class libraries that are needed for implementation work, but that should not be used at runtime.
 - **integrator/**: holds the configuration files for Sterling Integrator.
 - **lib/**: holds the Java class libraries used by the Visual Modeler Web application.
 - **lib/winnt/**: this holds any Windows-specific DLL files that are required.
 - **messages/**: holds the DTDs for the Sterling message family.

- **properties/**: holds the **Comergent.xml** file and the other configuration files used to set the configuration of the server.
- **reports/**: holds the files required for Sterling Analyzer.
- **rosettanet/**: contains the DTD and XML files that define the RosettaNet messages supported by the Visual Modeler.
- **schema/**: holds the XML files that specify the schema for your implementation.
- **stylesheets/**: holds the XSL files used to translate messages from one message family to another.
- **templates/**: holds the text templates used to generate messages such as email notifications.
- **web/**: holds most of the JSP pages, HTML pages and support files for the applications. It has the following structure:

/locale/ directories for each of the Visual Modeler applications. Each locale supported by your implementation of the Visual Modeler must have its own set of JSP pages in its corresponding locale directory.

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

Cron Job Setup

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

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

- Maintain Indexsets
- Product Sync

- User Sync

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

- Nightly Segments Build
- Reprocess Segments

Setting Up Apache as a Front-end to Tomcat

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

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

Prerequisites

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

Overview

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

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

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

1. The browser connects to Apache's port 80 and submits its request.

2. Apache determines if the incoming URL needs to be managed by the JK 1.2 connector, `mod_jk`.
3. If so, then Apache initiates an AJP 1.3 connection to Tomcat's port 8009. The request is now sent to Tomcat.
4. Tomcat processes the request and returns the response through the same AJP 1.3 connection.
5. Apache in turn relays the same response to the browser.

Configuring Apache to Use `mod_jk`

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

- a. Add the following line in the LoadModule section:

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

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

- b. Add an IfModule element to force Apache to set up the Tomcat servlet container connection and access to the Visual Modeler web application:

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

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

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

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

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



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

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

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

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

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

Configure Tomcat to Use mod_jk

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

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

Edit Tomcat's `tomcat_home/conf/server.xml` file:

1. Add the following line to the Listeners section:

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

Starting Apache and Tomcat

1. Start up Apache, then start up Tomcat.

2. Try:

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

to verify that you can access the Visual Modeler through Apache.

Setting up Apache to Support SSL

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

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

```
LoadModule ssl_module modules/mod_ssl.so

and

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

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

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

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

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

Then use the key to generate the certificate:

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

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

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

Keep Alive Settings

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

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

```
KeepAlive On
```

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

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

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

Filtering Static Content

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

Setting up Apache to Serve Static Content

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

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

Follow these steps:

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

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

2. Add the following expires rules:

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

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

3. Restart the Apache Web server.

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

Creating a NSAPI Filter

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

1. Locate the **ctrans.c** file under *debs_home/*.
2. Compile it to a dynamic library.

For Solaris, the compile command is:

```
gcc -DXP_Unix -DMCC_HTTPD -DNET_SSL -DSOLARIS -D_REENTRANT -Wall -
c module.c * ld -G module.o -o module.so
```

For Windows, the compile command is:

```
cl -LD -MD -DMCC_HTTPD -DXP_WIN32 -DNET_SSL module.c -
Insapl\include /link nsapi\examples\libhttpd.lib
```

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

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

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

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

- b. Find the block:

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

- c. Add the line:

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

By default, use the following values for the parameters:

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

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

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

Compressing Output From the Visual Modeler

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

These steps assume that you have set up Apache as a front-end to the servlet container in which the Visual Modeler is deployed. For example, see "Setting Up Apache as a Front-end to Tomcat" on page 119.

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

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

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

```
<Location /Sterling>  
  
# Insert filter  
SetOutputFilter DEFLATE  
  
# Netscape 4.x has some problems...  
BrowserMatch ^Mozilla/4 gzip-only-text/html  
  
# Netscape 4.06-4.08 have some more problems  
BrowserMatch ^Mozilla/4\.0[678] no-gzip
```

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

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

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

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

</Location>
```

If necessary, change the context string “/Sterling” to the name of the context used for the Visual Modeler.

Creating and Populating the Knowledgebase

This chapter describes how you create the Knowledgebase using the standard Visual Modeler schema. You must run the schema creation and data scripts to create and populate the Knowledgebase in your designated database server.

These steps enable you to test that the installation of the Visual Modeler has been successful. By the end of this chapter you will be able to log in to the Visual Modeler and verify that the basic functionality works.

If you are upgrading your installation of the Visual Modeler from an earlier release, then you can migrate the data to a Release 9.0 Knowledgebase.

Gathering the Database Information

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

1. Identify the connection information required to connect from the Visual Modeler machine to the database server.
 - For an Oracle Server, this is the machine name or IP address of the database server, the port at which the database server is listening, and the SID (name of the database instance) of the database server. You must create a TNS alias for the database server on the Visual Modeler machine.
 - For a SQL Server, this is the machine name or IP address of the database server.

2. Establish a database userid on the database server which is used by the Visual Modeler.

Use this userid to perform all the Visual Modeler-related calls to the database. This userid must have sufficient privileges to create and modify database tables.

You may use one of your existing database userids. However, we suggest that you set up a database userid that is dedicated to Visual Modeler-related tasks.

3. Check that you can connect to the database server using the userid and connection information:
 - For Oracle, you must be able to connect from the Visual Modeler machine to the database server using SQL*Plus and the TNS alias.
 - For SQL Server, you must be able to connect using OSQL and the JDBC data source name and userid.

Creating the Knowledgebase Schema

You run the schema creation script to create the Knowledgebase in your designated database server. You can run the schema creation script as a batch file as described in this section or from the SDK. See "To Run the Schema Creation Script with the SDK" on page 130.

Attention: Running the schema creation scripts directly is no longer supported. You must run the createDB target as described in "Installing the Visual Modeler Using the SDK" on page 75.

Before using the schema creation scripts and XML Loader scripts, you must decide what locales your implementation will support. You should remove any locales from the scripts that you do *not* intend to support. To begin with, you can try just creating the "en_US" locale and adding more as the implementation progresses.

Creating the Schema

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

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

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

By default, the Oracle schema creation script creates indexes for tables in a separate tablespace called **INDX**. You can choose to create the indexes in the same tablespace as the main schema or in a tablespace of your choice. To do either of these two choices, you must edit the **oracle_indexes.sql** SQL script.

- If you are running against a Microsoft SQL Server database server, then run **MssqlCreateSchema.bat** from the *debs_home/Sterling/* directory. Make sure the database is the default database for the userid you use.

Locales and Loading Data Using the XML Loader

The current schema creation script creates seven locales as part of the table creation script. You should review this part of the table creation script and modify it to remove locales if need be before running the script.

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

TABLE 11. Database Sorting Settings

Database	Locale	Value
Oracle	en_US	BINARY
	de_DE	XGERMAN
	fr_FR	XFRENCH
	fr_CH	BINARY
	de_CH	BINARY
	ja_JP	BINARY
	zh_TW	BINARY

TABLE 11. Database Sorting Settings (Continued)

Database	Locale	Value
SQL Server	en_US	Latin1_General_BIN
	de_DE	FRENCH_CS_AS
	fr_FR	FRENCH_CS_AS
	fr_CH	SQL_Latin1_General_CP1_CI_AS
	de_CH	SQL_Latin1_General_CP1_CI_AS
	ja_JP	SQL_Latin1_General_CP1_CI_AS
	zh_TW	Chinese_Taiwan_Stroke_CS_AS

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

To Run the Schema Creation Script with the SDK

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

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

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

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

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

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

2. Check that the appropriate database client software has been installed on your machine and that it is in your path. For example, if you are using an Oracle server for the Knowledgebase, then make sure that SQLPLUS is in your path. Make sure also that the **tnsnames.ora** file includes an alias as specified as the ORACLE_DATABASE parameter. You should be able to successfully run:

```
tnsping <TNS alias>
```

3. Run the createdDB target from the SDK.

4. Check the generated log files to verify that the script ran without error.

Populating the Knowledgebase

Having created the Knowledgebase, you need to load data into it, in order to run the Visual Modeler. Release 9.0 loads data into the Knowledgebase using a set of XML definitions of each data object. Data loading is invoked from the SDK using the loadDB and loadMatrixDB targets: these load the minimal and reference data sets respectively. These targets invoke the XMLLoader scripts as described in "XMLLoader Script" on page 132.

XML Data Format

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

The top-level element has these attributes:

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

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

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

The files must be in one of two forms:

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

```
<PromotionData>
  <!--Record 1 ----->
  <Promotion state="INSERTED" type="BusinessObject">
    <PromotionKey state="INSERTED">126</PromotionKey>
    <PromoCode state="INSERTED">ID 360</PromoCode>
```

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

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

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

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

XMLLoader Script

The XML data loading script invokes a Java class that uses the Visual Modeler DataManager to load each object.

The script is called **XMLLoader.bat** (or **XMLLoader.sh** on UNIX systems) and it is located in *debs_home*/Sterling/WEB-INF/scripts/. The Visual Modeler provides two sets of data that can be loaded: see "Data Sets" on page 135 for further information.

Encryption

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

Attention: You <i>must</i> determine which fields are to be encrypted before loading any data into the Knowledgebase.
--

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

Defining the Knowledgebase as the Data Source

To run the data loading script, you must configure the Visual Modeler to access the Knowledgebase. You do this using the configuration files **prefs.xml**, and **DataServices.xml**. You must also ensure that the DsKeyGenerators element of the **DataServices.xml** file points to the correct key generator file:

MsSqlKeyGenerators.xml, or **OracleKeyGenerators.xml**. Note that if you use the SDK to install the Visual Modeler, then the correct values will be set up automatically in these files.

Attention: Please read the database server-specific instructions below. Set the logging level to INFO before running the data loading script.

The following sections provide a brief description of the syntax of the **DataSources.xml** file for the supported database servers.

MsSqlDataSources Syntax

```
<Primary DataService="MsSqlService" SubType="MS"
  ConnectionString="MSSQL_MACHINE"
  UserId="MSSQL_USERNAME" Password="MSSQL_PASSWORD" />
```

- MSSQL_MACHINE is the machine name or IP address of the machine on which SQL Server is running.
- MSSQL_USERNAME and MSSQL_PASSWORD are the username and password used to create the Visual Modeler schema. Note that the default database for this user must be the database in which the schema was created.

OracleDataSources Syntax

You can use either the OCI JDBC driver or the Oracle thin client JDBC driver to connect from the Visual Modeler machine to the Oracle Server. Use:

- For Oracle 8i:

```
<Primary DataService="JdbcService" SubType="ORACLE"
  ConnectionString="jdbc:oracle:oci8:@ALIAS"
  UserId="ORACLE_USERNAME" Password="ORACLE_PASSWORD"/>
```

- For Oracle 9i:

```
<Primary DataService="JdbcService" SubType="ORACLE"
  ConnectionString="jdbc:oracle:oci:@ALIAS"
  UserId="ORACLE_USERNAME" Password="ORACLE_PASSWORD"/>
```

or

```
<Primary DataService="JdbcService" SubType="ORACLE"
ConnectionString="jdbc:oracle:thin:@ORACLE_MACHINE:ORACLE_PORT:SID"
  UserId="ORACLE_USERNAME" Password="ORACLE_USERNAME" />
```

- ALIAS is the TNSNAMES alias set up on the Visual Modeler machine.
- ORACLE_USERNAME and ORACLE_PASSWORD are the userid used to create the Visual Modeler schema.
- ORACLE_MACHINE is the machine name or IP address of the machine on which the Oracle Server is running.
- ORACLE_PORT is the port number at which the Oracle Server listener is listening for connections.
- SID is the Oracle SID of the database.

Internationalization and Support for Locales

Creating Locales

You must edit the file **LocaleDataList** (located in *debs_home*/Sterling/WEB-INF/**xmldata**) to specify the locales that you want the Knowledgebase to support. Each installation of the Visual Modeler can support one or more locales.

Make sure that the server locale (defined by the defaultSystemLocale element of the **Internationalization.xml** file) is included in the list of locales defined in **LocaleDataList**.

Updating Data using XMLLoader

When you load locale-specific data into the Knowledgebase, you can make use of the XMLLoader's ability to modify data objects. In each data object element and child elements, set the state attribute to "Modified". This will update business objects rather than inserting a new business object.

This feature is particularly useful when you are adding locale-specific information to an existing implementation of the Visual Modeler. For example, the following data object can be used to update a product category business object:

```
<ProductCategory state="MODIFIED">
  <ProductCategoryKey state="MODIFIED">1002</ProductCategoryKey>
  <ParentCategoryKey state="MODIFIED">-1</ParentCategoryKey>
  <SequenceId state="MODIFIED">5</SequenceId>
  <ResourceKey state="MODIFIED">3</ResourceKey>
  <StartDate state="MODIFIED">2000-10-06 17:20:28.0</StartDate>
  <EndDate state="MODIFIED">2100-10-06 17:20:28.0</EndDate>
  <OwnedBy state="MODIFIED">1</OwnedBy>
  <AccessKey state="MODIFIED">2006</AccessKey>
```

```
<ProductCategoryLocale state="MODIFIED">
  <Locale state="MODIFIED">de_DE</Locale>
  <Name state="MODIFIED">Software</Name>
  <Description state="MODIFIED">
    Alle Anwendungspakete, die auf unserer Site vorhanden
    sind, werden auf allen unsere Qualitätscomputersysteme geprüft und
    bestätigt.
  </Description>
</ProductCategoryLocale>
</ProductCategory>
```

Database Server-Specific Steps

When running the data loading scripts, the steps vary a little from one database server to another. This section covers the supported database servers.

SQL Server Steps

1. If you are using MS SQL Server as your database server, then make sure that you have copied the **MsSqlJNI.dll** file to the **Winnt\system32** directory on the Visual Modeler machine.
2. Edit the **DataServices.xml** file so that all the **JdbcDriver** elements are commented out. Use the **<!--** and **-->** tags to comment out each element.
3. In **MsSqlDataSources.xml**, make sure that the connection information sets the same **UserId** and **Password** as were used to create the schema.

Oracle Steps

1. Edit the **DataServices.xml** file to specify the **OracleDataSources.xml** file and **OracleKeyGenerators.xml** file.
2. Make sure that the **JdbcDriver** element takes the value of the name of the Oracle JDBC driver.

Data Sets

The Visual Modeler provides two sets of XML data objects:

- Reference implementation: this set populates the Knowledgebase with the complete reference implementation (Matrix Solutions) data set. You use this set if you want to deploy our reference implementation in order to familiarize yourself with the Visual Modeler.
- Minimal implementation: this set populates the Knowledgebase with the minimal data required to get the Visual Modeler up and running. You use this set when you want to deploy your production system using your data.

See "Email Addresses" on page 81 for information about email addresses set in the minimal data set.

Attention: You must always install the minimal data set; you can optionally layer the reference data on top. Use the SDK loadDB target to load the minimal data only; use the loadMatrixDB target to load both.
--

To Edit and Run the XML Data Loading Script

1. Configure the **Comergent.xml**, **DataServices.xml**, and appropriate **DataSources.xml** configuration files to point to the database server to be used for the Knowledgebase.
 - a. Make sure that the DataServices element of the **Comergent.xml** file points to the correct location of the **DataServices.xml** file.
 - b. Make sure that the DsDataSources element of the DataServices.xml file points to the correct location of your **DataSources.xml** file.
 - c. Make sure that the appropriate connection information has been entered in the **DataSources.xml** file.

2. If your environment does not have a JAVA_HOME environment variable set, then edit the **XMLLoader.bat** to set the JAVA_HOME environment variable to point to your installation of the JDK. For example:

```
>set JAVA_HOME=C:\JDK1.2.2
```

If you are using Oracle as the database server, then make sure that the classpath is set to include the location of the appropriate JDBC driver class. Typically, you must ensure that the XMLLoader script includes a line of the following form:

```
CP=%CP%;%DH%/lib/oracle816_jdbc2.jar
```

3. Save the edited file to **debs_home/Sterling/**.
4. Run the XML data loading script from **debs_home/Sterling/**.

- The syntax to load the reference data is:

```
>XMLLoader persist
```

- The syntax to load the minimal data is:

```
>XMLLoader persist minimal
```


Removing Locales

Out of the box, the schema creation scripts and the minimal and reference data sets create data for several locales. Before going live with your implementation, you should remove from the Knowledgebase any locales not supported by your implementation.

To remove a locale, you must remove references to it from the following tables:

- CMGT_ANALYZER_TEXT: for example

```
DELETE FROM CMGT_ANALYZER_TEXT WHERE LOCALE = 'de_DE';
```

- CMGT_CURRENCIES: for example

```
DELETE FROM CMGT_CURRENCIES WHERE LOCALE = 'de_DE';
```

- CMGT_LOCALE: for example

```
DELETE FROM CMGT_LOCALE WHERE LOCALE_NAME = 'de_DE';
```

- CMGT_LOCALE_CURRENCY: for example

```
DELETE FROM CMGT_LOCALE_CURRENCY WHERE LOCALE_NAME = 'de_DE';
```

- CMGT_LOCALE_NAMES: for example

```
DELETE FROM CMGT_LOCALE_NAMES WHERE LOCALE_NAME = 'de_DE';  
DELETE FROM CMGT_LOCALE_NAMES WHERE EFFECTIVE_LOCALE = 'de_DE';
```

- CMGT_LOOKUPS: for example

```
DELETE FROM CMGT_LOOKUPS WHERE LOCALE = 'de_DE';
```

- CMGT_<OBJECT>_LOCALE: there are multiple tables that store locale-specific strings for data objects such as products, features, and so on. You must remove the references to the deleted locale from each such table. For example

```
DELETE FROM CMGT_<OBJECT>_LOCALE WHERE LOCALE_NAME = 'de_DE';
```

Logging into the Visual Modeler

Point your browser to the standard login page. The standard URL to access this page is:

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

Irrespective of whether you have generated the reference data set or minimal data set, you can log in as the enterprise administrator whose username and password are “admin” and “admin”.

You can now administer the Visual Modeler through the standard browser interface.

<p>Attention: Before going live with your implementation of the Visual Modeler, you <i>must</i> change the passwords of the admin and ERPAdmin users. Failure to do so presents a security breach.</p> <p>Do not use the ERPAdmin user for administration tasks. It is intended only for integration with an ERP system. You should not use this user to log in to the system through the Web.</p>

When you have successfully completed your installation, proceed to the next chapter. Otherwise, use CHAPTER 8, "Troubleshooting and Backing Up the Visual Modeler" to troubleshoot your installation.

Troubleshooting and Backing Up the Visual Modeler

Troubleshooting

Testing with the Administration URL

You can make sure that the various parts of the installation are functioning by pointing your browser to the URL used to access the administration pages:

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

Email Server

You must make sure that the SMTP Mail Server used to send email from the Visual Modeler is up and running. Make sure that you can ping the SMTP Mail Server from the Visual Modeler machine using the machine name specified in the SMTPHost element of the **Comergent.xml** file. Storefront administrators can configure the SMTP host by setting the SMTP Host Machine system property to the appropriate value. To set the SMTP Host Machine system property, navigate from the System Administration panel of the home page to System Services, then the Commerce Manager category, then the SMTP category, and enter the appropriate value in the SMTP Host Machine field.

Certain UTF-8 characters may not display well in the subject lines of email sent from the Visual Modeler to users. This is due to email clients that are not configured to display UTF-8 characters correctly. If the problem persists, review

the characters being used in the subject lines of the email and provide information to users about suitable email clients.

General Troubleshooting Tips

This section includes general diagnosis approaches that can help to quickly pinpoint the source of your problem.

Tomcat Server

The following considerations apply to running Apache Tomcat:

- `SESSIONS.ser`: when Tomcat is shut down, the server saves the current session information to a file called `container_home/work/Standalone/localhost/Sterling/SESSIONS.ser`. You should delete copies of this file before restarting Tomcat.
- By default, Tomcat does not recompile JSP pages if it determines that its compiled version has a timestamp newer than the corresponding JSP page. If you see errors relating to `MethodNotFound` exceptions, then the likely cause is an old compiled page. You can solve this problem by deleting the `container_home/work/Standalone/localhost/Sterling/` directory to force re-compilation of all the JSP pages.

Common Problems

This section covers problems that commonly occur during startup and runtime. You can use the `messageTypeValidate` element to validate all the message types as the system starts. The element is set to `TRUE` by default. You should set this element to `FALSE` once the system has passed its acceptance tests.

Errors at Startup Time

When the Visual Modeler is started by the servlet container, it logs its progress through initialization. Look for these errors in the console window or event log:

TABLE 12. Startup Errors

Error	Cause and Solution
<pre>InputStream: 24, 384: "</Comergent>" expected.</pre>	<p>A syntax error in one of the configuration files has caused the DataManager to fail to initialize. You must correct the syntax error. The InputStream line provides the exact location of the error.</p>
<pre>java.io.FileNotFoundException: C:\jakarta- tomcat\webapps\sterling\WEB-INF\proper- ties\Comergent.xml (The system cannot find the file specified)</pre>	<p>The main Comergent.xml file is not in the correct location as specified by the propertiesFile element of the web.xml file.</p>
<pre>Env/main:W1:DATASERVICES Primary Connection Failed: Io exception: Connection refused(DESCRIPTION =(TMP=) (VSN- NUM=135290880) (ERR=12505) (ERROR_STACK=(ERROR =(CODE=12505) (EMFI=4))))</pre>	<p>The server has failed to connect to the database server. Perform the following checks:</p> <p>Ping the database server machine; there may be a network failure. Enter the IP address of the machine to see whether the host name can be resolved.</p> <p>If you can ping the database server machine, then check that the database connection information that you have entered in the DataSources.xml file is correct. If possible, use an alternate database connection method (such as SQLPLUS or SQL Server's Enterprise Manager). If this fails, then either the database is down or you have incorrect connection information.</p>
<pre>Primary Connection Failed: No suitable driver</pre>	<p>The server has failed to find a valid Driver class in its classpath. Check that any JDBC driver specified in the DataServices.xml file lies in one of the classpath directories or archive files. In particular, check that an appropriate JDBC driver has been specified to connect to the database server: For Oracle Servers, you may use <code>oracle.jdbc.odb.OracleDriver</code>.</p>
<pre>Cannot find file=web-inf/HostedPartner.xml</pre>	<p>The initialization servlet has failed to find one of the properties files referred to in the main Comergent.xml file. Check the names and paths to the properties files. In a UNIX installation, check for case-sensitive file names.</p>

TABLE 12. Startup Errors (Continued)

Error	Cause and Solution
<p>Failed to create a NameKeyTable. com.comergent.dcms.util.ICCException: [CMGT_E_SCHEMA_KEY_GEN_NOT_FOUND] error: "Schema Error - DataObject: Promotion ExternalObject: PromotionControl specifies KeyGenerator: ControlKey which does not exist."</p>	<p>A problem lies in your definition of the XML schema. On startup, the Visual Modeler reads the schema files and attempts to load the schema as an internal data structure. Exceptions are most commonly thrown when the definition of an element is omitted from the schema.</p> <p>In this example, the Visual Modeler has read the DsRecipes.xml file, and attempted to load the Promotion DataObject. This DataObject includes in its definition file, Promotion.xml, a reference to a KeyGenerator element called "ControlKey". Inspection of the DsKeyGenerators file shows that no KeyGenerator element called ControlKey is declared.</p>
<p>java.net.BindException: Address in use</p>	<p>A process is already bound to one of the ports that the Visual Modeler is attempting to use. You must either stop the existing process that is using the port or use a different port.</p>
<p>com.comergent.dcm.util.ICCException: [CMGT_E_UNKNOWN_ELEMENT] error: "Element: Comment of DataObject: Comment is not in the DCMS schema"</p>	<p>An error has occurred while the DataManager initializes the schema. Check the definition of business and data objects. Check that a header DsElement has been declared for the data object and that DsElements are declared for each data element.</p>
<p>Comergent Init Servlet: DataManager NOT initialized com.comergent.dcm.util.ICCException: [CMGT_E_SCHEMA_KEY_GEN_NOT_FOUND] error: "Schema Error - DataObject: OrderAddress ExternalObject: OrderAddress specifies KeyGenerator: OrderAddressKey which does not exist."</p>	<p>You have declared a KeyGenerator in the data object definition, but it is not defined in the KeyGenerators.xml file. Make sure that you have modified the correct KeyGenerators.xml file: this is usually OracleKeyGenerators.xml or MsSqlKeyGenerators.xml. Also make sure that your DataServices.xml file points to the correct KeyGenerators.xml file.</p>
<p>2002.10.22 13:33:03:459 Env/Thread-2:ER:DATASERVICES JDBCService.restore Error: ORA-00600: internal error code, arguments: [ttcgcsynd-1], [0], [], [], [], [], [] java.sql.SQLException: ORA-00600: internal error code, arguments: [ttcgcsynd-1], [0], [], [], [], [], [] at oracle.jdbc.dbaccess.DBError.throwSQLException(DBError.java:168) at oracle.jdbc.ttc7.TTIoer.processError(TTIoer.java:208) ...</p>	<p>There is a mismatch between the Oracle database version (usually 8i or 9i) and the JDBC driver that is being used to connect from the Visual Modeler. Check the classpath that the servlet container is using and remove any references to JDBC driver JAR files that come before the oraclejdbc.jar file in <i>debs_home/Sterling/WEB-INF/</i>.</p>

TABLE 12. Startup Errors (Continued)

Error	Cause and Solution
Env/main:ER:MSGT Illegal Unresolved Reference to a MessageType: contentFrame	A MessageTypeRef element references a message type definition that does not exist.
Env/main:ER:MSGT com.comergent.api.dcm.messageType.MessageTypeInstantiationException: Failed instantiating or locating com.comergent.apps.partnerMkt.blc.MissingBLC in MessageType GenericLoginDisplay	A message type failed validation: typically, this means that one of its elements is missing such as a missing BLC, controller class, or JSP page.
2003.08.05 15:35:28:359 Env/Thread-6:ER:CORE java.lang.NoClassDefFoundError java.lang.NoClassDefFoundError at com.comergent.dcm.authentication.UserPasswordCredentials.verify(UserPasswordCredentials.java:38) at com.comergent.dcm.authentication.LoginController.execute(LoginController.java:59)	On startup, the Visual Modeler has tried to re-instantiate a session stored when the servlet container was last stopped. Before starting the servlet container, make sure that you have deleted any stored sessions. For example, in Tomcat 4.1, check the <i>container_home/work/Standalone/localhost/Sterling/</i> directory, and delete any files called SESSIONS.ser .

Errors at Runtime

Some errors are listed here that have occurred infrequently in running instances of the Visual Modeler.

TABLE 13. Runtime Errors

Error	Cause and Solution
<p>Assertion failed: 1 == pConnectionObject->fCallCheck, file q:\SPHINX\NETLIBS\nt\sock\src\ntssockc.c, line 1039</p>	<p>This error has been observed when the Visual Modeler is run with SQL Server. Make sure that you have applied the latest Windows and SQL Server Service Packs to the machine on which SQL Server is running, and make sure that the client SQL Server software installed on the Visual Modeler machine matches your version of SQL Server.</p>
<p>On Solaris, the servlet container cannot find a certain servlet or URL.</p>	<p>First make sure that you did not make a typo. If you are certain that there was no mistake, then do the following:</p> <ol style="list-style-type: none"> 1. Run the following command on web.xml: <pre>java com.comergent.dcm.util.CheckWebXML web.xml > newWeb.xml</pre> 2. Edit the file newWeb.xml. Look for the following string <pre><!-- (8192) XXX BOUNDARY BREAK --></pre> <p>The start of the comment <!-- is the start of a 8192 boundary break. If it falls within a value for an XML node, then that node will get truncated.</p> <p>A work around is to pad the web.xml file such that the boundary break will fall inside a comment. For more information, see the comments at the start of file CheckWebXML.java.</p>

TABLE 13. Runtime Errors (Continued)

Error	Cause and Solution
<p>You see parser errors such as:</p> <pre>java.lang.NoSuchMethodError at org.apache.xpath.DOM2Helper.getNamespaceOfNode (DOM2Helper.java:348) at org.apache.xml.utils.TreeWalker.startNode (TreeWalker.java:281) at org.apache.xml.utils.TreeWalker.traverse (TreeWalker.java:119) at org.apache.xalan.transformer.TransformerIdentityImpl.transform (TransformerIdentityImpl.java:320)</pre>	<p>Check that you have followed the instructions to copy the XML parser-related JAR files to the servlet container's lib/ directory, and that you have removed any default parser.jar files.</p>
<p>Running iPlanet, you see the following in your browser:</p> <pre>GX Error (GX2GX) socket result code missing!!!</pre>	<p>There is a mismatch between the web.xml and ias-web.xml files. All servlets mentioned in web.xml must have a corresponding entry in the ias-web.xml file. Use the kguidgen utility to generate a GUID for the servlet.</p>

Backing Up the Visual Modeler

It is good practice to plan for the possibility of a catastrophic failure that renders the Visual Modeler machine unusable. In this eventuality, you need to be able to restore the Visual Modeler as rapidly as possible.

We suggest taking the following steps:

1. Replicate the servlet container: keep the installable for the exact release of the servlet container, together with any patches applied. Back up any changes to archive files, or startup scripts that might affect the order of class-loading for example. A copy of the JDK used to run the servlet container would also be useful.
2. Back up the Visual Modeler itself: that is, create a WAR file of the running Sterling Web application directory. This will capture any changes to system properties, business rules, as well as the XML model files, resource files (product images and so on) and other files (such as uploaded GIF files).

Note that the Visual Modeler enables a fair amount of customization that can place files outside of the Visual Modeler Web application directory. If you take advantage of this capability, then you have to backup these directories too. In particular, a clustered installation of the Visual Modeler requires the creation of a shared location that will be external to the Web application directory on any particular machine.

3. Take a snapshot of the database at regular intervals: verify that the Visual Modeler Knowledgebase can be restored from this backup.
4. Make a copy of the **dcmsKey.ser** file and put this in a very safe place. Encrypted data will be unrecoverable if this file is lost. For customers with Release 6.3 or higher, this instruction needs to be modified depending on your choice of encryption scheme and key management policy.

This chapter provides a description of the logging service used to manage logging messages in the Visual Modeler.

Logging

Use the logging settings of the Visual Modeler to monitor activity of the Visual Modeler and to help diagnose problems.

Logging Preferences and Configuration

The log4j API handles logging and uses the Preferences API to retrieve logging configuration properties. The basic configuration file for the log4j API is **log4j.properties**. A copy of this file with default logging properties is included in the **WEB-INF/lib/cmgt-logging.jar** JAR file packaged with the Visual Modeler and is also placed in the **WEB-INF/classes/** directory. To override the default properties permanently, you must modify the **log4j.properties** file. Any values that you specify in this file will overwrite the corresponding values in the **log4j.properties** file in the **cmgt-logging.jar** file. You can override the default properties on a transient basis for testing purposes by logging in to the System Administration site of the Visual Modeler as a site administrator and modifying the System Logging properties. See "Making Transient Logging Configuration Changes" on page 149 for more information.

The following sections describe some typical changes you may want to make:

- "Logging to the Console" on page 148
- "Changing Logging Level for a Package" on page 148
- "Formatting Logging" on page 149
- "Logging File Size" on page 149

Logging to the Console

If you want logging output to the standard output stream, rather than to a logging file, specify the use of the STDOUT appender:

```
log4j.rootCategory=info, STDOUT
```

Depending on the configuration of the servlet container, the logging output will be directed to the standard destination of the System.out output stream. Note that when you specify a different appender, then you must include the appender's properties in the custom **log4j.properties** file too. For example:

```
log4j.appender.STDOUT=org.apache.log4j.ConsoleAppender
log4j.appender.STDOUT.layout=org.apache.log4j.PatternLayout
log4j.appender.STDOUT.layout.ConversionPattern=[%t] (%c{2}) - %m%n
```

Note that you can also force logging to be output to the System.out output stream by specifying `-Dcomergent.console.logging=true` as part of the command line that starts the servlet container. This overrides any logging properties specified in the **log4j.properties** configuration file.

Changing Logging Level for a Package

If you want to see more detailed logging information from just one Java package as it is executed, then you can specify this by overriding the root logging level. By default, all logging is done at the INFO level, because the rootCategory is defined as follows:

```
log4j.rootCategory=info, CMGT
```

For example, to specify DEBUG level logging for the visualModeler API package, enter:

```
log4j.logger.com.comergent.api.apps.visualModeler=DEBUG
```

The specification of logging is hierarchical following the package organization of the Visual Modeler. Thus if you specify:

```
log4j.logger.com.comergent.api.apps=WARNING
```

Then, all logging at the API level will be done at the WARNING level except for the visualModeler package.

Formatting Logging

You can format the logging output to suit your needs. For example, the following will provide a more compact logging format than the standard default layout:

```
log4j.appender.CMGT.layout.ConversionPattern=[%r] [%t] (%c{1}) - %m%n
```

Logging File Size

If log files get too large, then consider modifying the logging preferences to rotate the log files. For example, you can specify that log files are rotated once they reach 10MBytes in size as follows:

After the line:

```
log4j.appender.CMGT=com.comergent.logging.ComergentRollingFileAppender
```

add:

```
log4j.appender.CMGT.MaxFileSize=100KB
```

Alternatively, to specify that log files should be rotated daily, change:

```
log4j.appender.CMGT=com.comergent.logging.ComergentRollingFileAppender
```

to:

```
log4j.appender.CMGT=com.comergent.logging.ComergentDailyRollingFileAppender
```

Making Transient Logging Configuration Changes

If you want to change the logging configuration settings for troubleshooting or other testing purposes, then make the changes as a site administrator using the **System Logging (log4j dynamic)** page of the site administration's System Properties page. Changes that you make remain in effect until you restart the Visual Modeler. If you are working in a clustered environment, then the logging configuration changes will be propagated to all the nodes in the cluster, and will also remain in effect until you restart the Visual Modeler.

To make permanent logging level changes, you must modify the **log4j.properties** file, as described in "Logging Preferences and Configuration" on page 147.

To make transient logging configuration changes:

1. Navigate to the System Administration URL and log in as a site administrator. The System Administration URL is similar to:

`http://server:port/Sterling/en/US/enterpriseMgr/admin`

The System Administration home page displays.

2. Click **System Services**, then click **System Logging (log4j dynamic)**.

The **Configure log4j** page displays, similar to the following figure.

3. Copy the name of the logger you want to change and paste it into the **Logger name:** field, then choose a logging level from the **Threshold** drop-down list.

Logging levels range from trace (logging all activity) to fatal (logging only fatal errors). You can also use the drop-down list to turn logging off.

4. Click **Update** to update the logging level. The new logging level displays in the Level column for the logger.

Logging File Locations

The location of logging files varies from one servlet container to another: here are some standard locations:

TABLE 14. Servlet Container Log Files

Servlet Container	Log File Location
Tomcat	<i>container_home/logs/</i>
WebLogic	<i>container_home/user_projects/domains/mydomain/</i>

You can provide your customers with an e-commerce experience in their preferred language and location, or locale. This chapter describes how to add support for locales other than United States English to an implementation of the Visual Modeler.

Individual locales are provided as localization pack JAR files that you install into your release using the SDK.

Attention: There are known issues with the Visual Modeler using SQL Server to support locales other than en_US.
--

This chapter covers the following topics:

- "Localization Concepts" on page 152
- "Localization Pack Installation Overview" on page 154
- "Localization Pack Installation Steps: New Implementation" on page 155
- "Localization Pack Installation Steps: Existing Implementation" on page 161

Localization Concepts

This section introduces localization concepts and the Visual Modeler support for localization.

Built-in Localization Support

The Visual Modeler has built-in support for:

- multiple currencies
- multiple languages
- number and date formats
- character sets

You can manage other aspects of localization for specific markets, such as:

- local laws and regulations
- currency processing
- shipping and export information
- tax

Locale Specification

You manage support for internationalization using locales. Each locale identifies a language and country. By identifying the locale to use when displaying information to a user, you ensure that the user sees information that is specific to their locale: they see your site's Web pages in their preferred language, with numbers and dates in their expected format.

A locale comprises a language and a country: for example, "English and United States" or "Italian and Switzerland". The same language may be used in more than one country: French in France, Switzerland, and Canada for example. In one country there may be speakers of more than one language: French, German, Italian, and Romansch in Switzerland for example.

The ISO standards 639 and 3166 specify a list of standard abbreviations for languages and countries that you must use. Some common language abbreviations are: Arabic (ar), Chinese (zh), English (en), French (fr), German (de), Hindi (hi), Japanese (ja), and Spanish (es).

Some common country abbreviations are: Canada (CA), China (CN), France (FR), Germany (DE), India (IN), Indonesia (ID), Japan (JA), United Kingdom (GB), and United States (US).

By combining a language and a country, you can uniquely specify a locale. For example: en_US (English-United States), it_CH (Italian-Switzerland), and zh_TW (Chinese-Taiwan). Locales are stored in the Visual Modeler using this representation.

Using Locales

Each installation of the Visual Modeler defines a *system default locale* in its **Internationalization.xml** file using the defaultSystemLocale element.

User Locales

When a user works in the Visual Modeler, their *current locale* determines the look-and-feel of Web pages and the locale-specific data (such as product descriptions) to use to display business object data to the user.

Each user has a *preferred locale* specified in their user profile. When a user first enters the Visual Modeler, their current locale is set to their preferred locale. If they change their current locale as they work, they see the Web pages in the new locale.

Users change locale by selecting a new locale from a drop-down list of available locales in their user profile. The display names for each locale in the drop-down list depend on the user's current locale. For example, if the user's current locale is en_US, the display name for fr_FR can be "French-France", and if the user's current locale is fr_FR, then the display name for en_US can be "Anglais-Etats Unis".

Default Locales for Languages

The Visual Modeler enables you to specify a default locale for each language so that if JSP pages are not available for the specific locale, the language's default locale's JSP pages are used instead. You specify the language's default locale using the defaultCountry elements of the Languages element of the **Internationalization.xml** configuration file.

The defaultSystemLocale element determines which JSP pages to serve if they are not provided in the language's default locale directory.

Sorting in Locales

The Visual Modeler enables users to sort displayed data in a number of ways while they perform their tasks. For example, they can sort the display of partners by name or inquiry lists by inquiry list ID. When a column is sorted based on a String value,

you can specify whether the sorting is performed using the binary value of the String or whether a locale-specific sort is used. This switch is set at the system level, so the same method is used for all such sorts.

<p>Note: SQL Server support for user-locale sorting is limited. You can set only a single collating sequence which is then used for all users.</p>

The sorting behavior is controlled by the UseLocalizedSort element of the **DataServices.xml** configuration file. The types of sorting behavior are:

- Binary: sorting is based on the binary value of the String value
- Locale specific

By default, the value of the UseLocalizedSort element is “false”: binary sorting is used. To use locale-specific sorting, set UseLocalizedSort to “true”. Note that binary sort is always fastest.

You can change the value of the UseLocalizedSort element by editing the **DataServices.xml** configuration file.

Site administrators can also change the UseLocalizedSort value as follows:

1. Navigate to the Visual Modeler administration site as a site administrator
2. Click **System Services** to display the System Properties page, then click **DataServices**
3. Scroll to Use Localized Sorting, then select the **false** or **true** radio button as required for your site

Note that you must stop and restart the Visual Modeler to enable the change.

The sorting behavior must be supported by the knowledgebase on which the Visual Modeler runs. See your database administrator for information about the type of sorting behavior supported for your implementation.

Localization Pack Installation Overview

The localization pack installation instructions describe how to install a Visual Modeler Release 9.0 localization pack using SDK 3.5.4. You can install into a new implementation or add support for a locale to an existing implementation. You install the localization pack into your release, not into your project.

These instructions assume that you are adding support for one or more locales to your new or existing implementation, but the default locale remains en_US.

What's contained in a localization pack:

- The resource bundle properties files (*.properties) for the locale.
- The locale's version of the look-up information from the **xmlloader** file, such as Keytype property names and values (descriptions).
- All the Javascript (.js) files.
- All the properties files with the locale appended to the file names, for example, **AttribMgrAGGENResources_fr.properties**.
- The .js files translated into the locale. The .js files retain their original names but are placed in locale-specific directories, such as fr\FR\js.

Note that the Matrix reference data, online help and online help images are not localized.

Localization Pack Installation Steps: New Implementation

These steps apply to a new implementation of the Visual Modeler.

These steps assume that you completed the Release 9.0 Visual Modeler set-up steps, including:

- Database configuration, privileges grants, and so on
- Installation of the Release 9.0 JAR file into your SDK
- Creation of a project using the `sdk newproject` target
- Installation of the appropriate database driver using the `sdk installDB` target, where *DB* is Oracle or MSSQLJDBC

1. Install the localization pack JAR file. For example:

```
sdk install SterlingSellingSuite-Locale-DEBS-9.0-fr-FR.jar
```

This installs the localization pack in the **releases\debs-9.0** directory.

2. Modify the **web.xml** file:

- a. Enter the following in a command window:

```
sdk customize web.xml
```

- b. Open ***sdk_home*\projects\project-name\WEB-INF\web.xml** in a text editor.

- c. Copy the section that begins with the following:

```
<!-- Start of English US mapping -->
```

And ends with the following:

```
<!-- End of English US mapping -->
```

- d. Paste the entire section after the English US mapping section.
- e. Modify the new section's comments to refer to the locale name you are adding. For example:

```
<!-- Start of French France mapping -->
```

- f. Modify the new section's file path references to refer to the locale name file path, such as **/fr/FR**. For example:

```
<servlet-mapping>
    <servlet-name>DispatchServlet</servlet-name>
    <url-pattern>/fr/FR/catalog/*</url-pattern>
</servlet-mapping>
```

3. Configure the **Internationalization.xml** file.

- a. Enter the following in a command window:

```
sdk customize Internationalization.xml
```

- b. Open **projects\project-name\templates\WEB-INF\properties\Internationalization.xml** in a text editor.

- c. Add the locale designation to the Presentation element's supportedLocales field. For example, for the French France locale:

```
<supportedLocales controlType="text" runtimeDisplayed="true"
ChangeOnlyAtBootTime="true" visible="true" boxsize="60"
displayQuestion="Presentation Locales"
displayOptions="en_US,en_US (English-United States),
zh_TW,zh_TW (Chinese-Taiwan),fr_FR,fr_FR
(French-France),fr_BE,fr_BE (French-Belgium),
de_DE,de_DE (German-Germany)" defaultChoice="en_US"
help="Supported presentation locales.">en_US,fr_FR</supportedLo-
cales>
```

- d. Add the following to the Languages element. Replace **la** with the language code and **CO** with the country code for your locale:

```
<la visible="false">
<defaultCountry controlType="text" runtimeDisplayed="true"
ChangeOnlyAtBootTime="true" visible="false" boxsize="60"
displayQuestion="URL for the Help Files" defaultChoice="US"
help="This is the default country for a specific language">CO
```

```
</defaultCountry>
</la>
```

For example, for the French France locale:

```
<fr visible="false">
<defaultCountry controlType="text" runtimeDisplayed="true"
ChangeOnlyAtBootTime="true" visible="false" boxsize="60"
displayQuestion="URL for the Help Files" defaultChoice="FR"
help="This is the default country for a specific language">FR
</defaultCountry>
</fr>
```

4. Add the locale table entry.

The locale table entry includes the DB_SORT_LOCALE_NAME field, which specifies how to sort data that your users are viewing. Possible values are BINARY (sort by the binary value of the string data value) and LATIN_GENERAL_BIN (perform locale-specific sorting). See "Sorting in Locales" on page 153 for information about determining sorting behavior for your implementation. Check with your Database Administrator to ensure that the sorting behavior that you select is supported for your database implementation.

For Oracle-based implementations:

- a. Enter the following in a command window:

```
sdk customize oracle_tables.sql
```

- b. Open **projects\project-name\WEB-INF\sql\Oracle\setup\oracle_tables.sql** in a text editor.
- c. Search for the text "INSERT INTO CMGT_LOCALE"
- d. Add an INSERT statement for the locale you are installing after the INSERT INTO CMGT_LOCALE statement. The format is:

```
INSERT INTO CMGT_LOCALE
(LOCALE_KEY, LOCALE_NAME, LOCALE_DESCRIPTION, ACTIVE_FLAG, DB_SORT_LO
CALE_NAME )
VALUES ( key, 'la_CO', 'Country Language', 'Y', 'BINARY')
/
```

- *Key* is the locale key. The locale key must be a unique numeric value.

- *la_CO* is the language code and country code. Use the appropriate language_COUNTRY encoding for the locale you are installing, for example, fr_FR for French France, jp_JP for Japanese Japan, and so on.

For example, the following INSERT statement is for supporting the fr_FR (French France) locale:

```
INSERT INTO CMGT_LOCALE
(LOCALE_KEY, LOCALE_NAME, LOCALE_DESCRIPTION, ACTIVE_FLAG, DB_SORT_LO
CALE_NAME )
VALUES ( 2, 'fr_FR', 'France French', 'Y', 'BINARY' )
/
```

- a. Save and close the file.

For SQL Server 2005-based implementations:

- a. Enter the following in a command window:

```
sdk customize mssql_schema.sql
```

- b. Open **projects\project-name\WEB-INF\sql\MSSql\setup\mssql_schema.sql** in a text editor.
- c. Search for the text "INSERT INTO CMGT_LOCALE"
- d. Add an INSERT statement for the locale you are installing after the INSERT INTO CMGT_LOCALE statement. The format is:

```
INSERT INTO CMGT_LOCALE
(LOCALE_KEY, LOCALE_NAME, LOCALE_DESCRIPTION, ACTIVE_FLAG,
DB_SORT_LOCALE_NAME ) VALUES ( key, 'la_CO', 'Country Lan-
guage', 'Y', 'LATIN_GENERAL_BIN' )
GO
```

- *Key* is the locale key. The locale key must be a unique numeric value.
- *la_CO* is the language code and country code. Use the appropriate language_COUNTRY encoding for the locale you are installing, for example, fr_FR for French France, jp_JP for Japanese Japan, and so on.

For example, the following INSERT statement is for supporting the fr_FR (French France) locale:

```
INSERT INTO CMGT_LOCALE
(LOCALE_KEY, LOCALE_NAME, LOCALE_DESCRIPTION, ACTIVE_FLAG,
DB_SORT_LOCALE_NAME )
```

```
VALUES ( 2, 'fr_FR', 'France French', 'Y', 'LATIN_GENERAL_BIN')
GO
```

e. Save and close the file.

5. Add the country translations:

a. Enter the following in a command window:

```
sdk customize LocaleNameDataList
```

b. Open

projects\project-name\WEB-INF\xml\data\LocaleNameDataList in a text editor.

c. The **LocaleNameDataList** file should contain only English and the language mappings for the locale(s) that you support. Remove all other Locale Mappings from the **LocaleNameDataList** file.

d. At the end of the **LocaleNameDataList** clause, add lines to supply translations for the United States and the country for which you are installing a locale. For example, to provide French translations for the United States and France:

```
<LocaleNameData state="INSERTED">
  <DisplayName state="INSERTED">France</DisplayName>
  <EffectiveLocale state="INSERTED">en_US</EffectiveLocale>
  <LocaleName state="INSERTED">fr_FR</LocaleName>
</LocaleNameData>
<LocaleNameData state="INSERTED">
  <DisplayName state="INSERTED">La France</DisplayName>
  <EffectiveLocale state="INSERTED">fr_FR</EffectiveLocale>
  <LocaleName state="INSERTED">fr_FR</LocaleName>
</LocaleNameData>
<LocaleNameData state="INSERTED">
  <DisplayName state="INSERTED">Les Etats-Unis</DisplayName>
  <EffectiveLocale state="INSERTED">fr_FR</EffectiveLocale>
  <LocaleName state="INSERTED">en_US</LocaleName>
</LocaleNameData>
```

If your implementation supports other locales, follow the same pattern so that each supported locale has translations for each country name.

6. Add loading of the **LightWeightLookupList** to the minimal data load.

a. Enter the following in a command window:

```
sdk customize LightWeightLookupList.lst
```

b. Open **projects\project-name\WEB-INF\scripts\LightWeightLookupList.lst** in a text editor.

- c. Add a line specifying the language_COUNTRY code:

```
WEB-INF/xmldata/I18N/la_CO/LightWeightLookupList
```

For example, to add French France:

```
WEB-INF/xmldata/I18N/fr_FR/LightWeightLookupList
```

- d. Save and close **LightWeightLookupList.lst**.
7. Configure the SearchConfigurationProperties.xml file.
 - a. Enter the following in a command window:

```
sdk customize SearchConfigurationProperties.xml
```

- b. Open **projects\project-name\WEB-INF\properties\SearchConfigurationProperties.xml** in a text editor.

- c. Add the following section to the the <Locales> element. Replace **la_CO** with the appropriate language_COUNTRY code for the locale you are installing, for example, fr_FR for French France.

```
<Locale id="la_CO" queryParserClass="com.comergent.api.appservices.search.queryParser.standard.CmgtQueryParser">
```

```
<Analyzers>  
<Analyzer analyzerClass="com.comergent.api.appservices.search.analysis.CatalogSearchAnalyzer" description="CatalogAnalyzer" id="search"/>  
<Analyzer analyzerClass="com.comergent.api.appservices.search.analysis.CatalogSearchAnalyzer" description="CatalogAnalyzer" id="build"/>  
</Analyzers>  
<DictionaryFile file="CatalogDictionary.mappings"/>  
</Locale>
```

- d. Save and close the file.

8. Rebuild the project:

```
sdk merge -clean
```

9. Load the database with the new locale information:

```
sdk createDB  
sdk loadDB or sdk loadMatrixDB  
sdk createSegDB  
sdk loadSegDB or sdk loadSegMatrixDB
```

10. Build the deployable (.war file) image:


```
sdk distWar
```

11. Deploy the .war file to your servlet container.
12. Navigate to the *sdk_home\dist\timestamp-WAR* directory, where *timestamp* has the form YYYYMMDD and is the date on which you issued the `sdk distWar` command. Rename the **prefs_dev.xml** file to **prefs.xml**, then copy it to the home directory of the user who is running the servlet container: *user_home/cmgt/debs/conf/* directory.
13. Restart your servlet container.
14. Verify that the installation succeeded:
 - a. Navigate to your implementation home page. The URL is similar to:

```
http://<server>:<port>/Sterling/en/US/enterpriseMgr/matrix
```
 - b. Log in as an administrator user, then click My Account.
The User Detail page displays.
 - c. The Preferred Locale drop-down list in the User Locale panel should include the locales that you just installed.
 - d. Choose a locale from the drop-down list, click **Save**, log out, and log back in.

The administrator user's home page should display with localized text. You can now create users who use the new locales.

Localization Pack Installation Steps: Existing Implementation

These steps apply to an existing implementation of the Visual Modeler.

Adding support for a locale to an existing implementation requires that you enter SQL commands directly to modify and populate the Knowledgebase. To complete these steps, you must have access to a SQL client such as Microsoft SQL Server Management Studio Express or SQLPlus for Oracle.

These steps assume that your implementation was installed using the SDK and that you have an existing release structure within which to work.

Before you begin, stop your servlet container instance.

1. Install the localization pack JAR file. For example:

```
sdk install SterlingSellingSuite-Locale-DEBS-9.0-de-DE.jar
```

This installs the localization pack in the **releases\debs-9.0** directory.

2. Modify the **web.xml** file:

- a. If you have not already customized your project's **web.xml** file for other locales, enter the following in a command window:

```
sdk customize web.xml
```

- b. Open **projects\project-name\WEB-INF\web.xml** in a text editor.

- c. Copy the section that begins with the following:

```
<!-- Start of English US mapping -->
```

And ends with the following:

```
<!-- End of English US mapping -->
```

- d. Paste the entire section after the English US mapping section.

- e. Modify the new section's comments to refer to the locale name you are adding. For example:

```
<!-- Start of German Germany mapping -->
```

- f. Modify the new section's file path references to refer to the locale name file path, such as **/de/DE**. For example:

```
<servlet-mapping>
    <servlet-name>DispatchServlet</servlet-name>
    <url-pattern>/de/DE/catalog/*</url-pattern>
</servlet-mapping>
```

3. Configure the **Internationalization.xml** file.

- a. If you have not already customized your project's **Internationalization.xml** file for other locales, enter the following in a command window:

```
sdk customize Internationalization.xml
```

- b. Open **projects\project-name\templates\WEB-INF\properties\Internationalization.xml** in a text editor.

- c. Add the locale designation to the Presentation element's supportedLocales field. For example, for the German Germany locale:

```
<supportedLocales controlType="text" runtimeDisplayed="true"
ChangeOnlyAtBootTime="true" visible="true" boxsize="60"
displayQuestion="Presentation Locales"
displayOptions="en_US,en_US (English-United States),
```

```
zh_TW, zh_TW (Chinese-Taiwan), fr_FR, fr_FR  
(French-France), fr_BE, fr_BE (French-Belgium),  
de_DE, de_DE (German-Germany)" defaultChoice="en_US"  
help="Supported presentation locales.">en_US, de_DE</supportedLo-  
cales>
```

- d. Add the following to the Languages element. Replace *la* with the language code and *CO* with the country code for your locale:

```
<la visible="false">  
<defaultCountry controlType="text" runtimeDisplayed="true"  
ChangeOnlyAtBootTime="true" visible="false" boxsize="60"  
displayQuestion="URL for the Help Files" defaultChoice="US"  
help="This is the default country for a specific language">CO  
</defaultCountry>  
</la>
```

For example, for the German Germany locale:

```
<de visible="false">  
<defaultCountry controlType="text" runtimeDisplayed="true"  
ChangeOnlyAtBootTime="true" visible="false" boxsize="60"  
displayQuestion="URL for the Help Files" defaultChoice="DE"  
help="This is the default country for a specific language">DE  
</defaultCountry>  
</de>
```

4. Modify the **I18NLookup.lst** file:

- a. Enter the following in a command window:

```
sdk customize I18NLookup.lst
```

- b. Open **projects\project-name\WEB-INF\xml\data\I18N\I18NLookup.lst** in a text editor.
- c. Add a line specifying the **LightWeightLookupList** file to load for the locale you are adding:

```
WEB-INF/xml\data/I18N/la_CO/LightWeightLookupList
```

For example, to load the **LightWeightLookupList** file for German Germany:

```
WEB-INF/xml\data/I18N/de_DE/LightWeightLookupList
```

5. Configure the **SearchConfigurationProperties.xml** file:

- a. If you have not already customized your **SearchConfigurationProperties.xml** file for other locales, enter the following in a command window:

```
sdk customize SearchConfigurationProperties.xml
```

- b. Open **projects\project-name\WEB-INF\properties\SearchConfigurationProperties.xml** in a text editor.
- c. Add the following section to the the <Locales> element. Replace *la_CO* with the appropriate language_COUNTRY code for the locale you are installing, for example, de_DE for German Germany.

```
<Locale id="de_DE" queryParserClass="com.comergent.api.appservices.search.queryParser.standard.CmgtQueryParser">
```

```
<Analyzers>  
<Analyzer analyzerClass="com.comergent.api.appservices.search.analysis.CatalogSearchAnalyzer"  
description="CatalogAnalyzer" id="search"/>  
<Analyzer analyzerClass="com.comergent.api.appservices.search.analysis.CatalogSearchAnalyzer"  
description="CatalogAnalyzer" id="build"/>  
</Analyzers>  
<DictionaryFile file="CatalogDictionary.mappings"/>  
</Locale>
```

- d. Save and close the file.
6. Set the location of the XML loader script and prefs.xml file:

- a. Enter the following in a command window:

```
sdk customize loadI18NFromXML.bat (For Windows systems)
```

or:

```
sdk customize loadI18NFromXML.sh (For Unix systems)
```

- b. Open **\projects\project_name\WEB-INF\scripts\loadI18NFromXML.bat** or **.sh** in a text editor.

- c. Search for the line containing:

```
set LOADER_JAR=%DEBS_RELEASE_DIR%/cmgt-xmlloader-tool.jar (For  
Windows systems)
```

```
LOADER_JAR=$DEBS_RELEASE_DIR/cmgt-xmlloader-tool.jar (For Unix  
systems)
```

- d. Replace %DEBS_RELEASE_DIR% (Windows) or \$DEBS_RELEASE (Unix) with the full pathname of your project's **image/data** directory location. Ensure that the directories are separated by forward slashes. For example: **C:/SDK351/releases/debs-9.0/image/data** (Windows) or **/debs/sdk351/releases/debs-9.0/image/data** (Unix).

- e. Specify the location of the `prefs.xml` preferences store file. Search for the line containing:

```
-DDataServices.General.ServerId=1
```

- f. Add the following parameter to the `JAVA_OPTS` parameter list:

```
-Dcomergent.preferences.store="prefs.xml_full_pathname"
```

Where `prefs.xml_full_pathname` is the location of the **prefs.xml** file in your project. For example, if your project name is `matrix` and your `sdk_home` is `/debs/sdk351`, the command is as follows:

```
$JAVA $JAVA_OPTS -DDataServices.General.ServerId=1  
-Dcomergent.preferences.store="/debs/sdk351/matrix/prefs.xml"  
-classpath $CP ${MAIN} dummy persist PARTNER_NAME=matrix  
$PHASE1_LIST MODE=QUIET
```

- g. Save and close the file.

7. Rebuild the project:

```
sdk merge -clean
```

8. Load the database with the new locale information:

- a. Use your SQL client tool to connect to the existing knowledgebase. For connection information, consult the values in the **projects\project_name\project_name-dev.properties** file.
- b. Update the `CMGT_LOCALE` table entry. In your SQL client tool, run the following SQL command:

```
INSERT INTO CMGT_LOCALE  
(LOCALE_KEY, LOCALE_NAME, LOCALE_DESCRIPTION, ACTIVE_FLAG, DB_SORT_LO  
CALE_NAME )  
VALUES ( key, 'la_CO', 'Country Language', 'Y', 'BINARY')
```

- *Key* is the locale key. The locale key must be a unique numeric value.
- *la_CO* is the language code and country code. Use the appropriate `language_COUNTRY` encoding for the locale you are installing, for example, `de_DE` for German Germany, `fr_FR` for French France, `jp_JP` for Japanese Japan, and so on.

For example, the following `INSERT` statement is for supporting the `de_DE` (German Germany) locale:

```
INSERT INTO CMGT_LOCALE
```

```
(LOCALE_KEY, LOCALE_NAME, LOCALE_DESCRIPTION, ACTIVE_FLAG, DB_SORT_LOCALE_NAME )
VALUES ( 3, 'de_DE', 'Germany German', 'Y', 'BINARY')
```

To check that your INSERT into the CMGT_LOCALE table is correct, run the following SQL statement:

```
select * from CMGT_LOCALE
```

9. Add the country name translations:

- a. Update the CMGT_LOCALE_NAME Table. In your SQL client tool, run the following SQL commands:

```
insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
DISPLAY_NAME, ACTIVE_FLAG) values ('la_CO', 'la_CO',
'Locale_Country_Name', 'Y')
insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
DISPLAY_NAME, ACTIVE_FLAG) values ('la_CO', 'defaultLA_defaultCO',
'default_Country_Name', 'Y')
insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
DISPLAY_NAME, ACTIVE_FLAG) values ('defaultLA_defaultCO', 'la_CO',
'Country', 'Y')
```

Where:

- *la_CO* is the language_COUNTRY combination for the locale you are adding, such as de_DE
- *Locale_Country_Name* is the name of the country in the locale's language, such as Deutschland
- *defaultLA_defaultCO* is the language_COUNTRY combination for the default locale, such as en_US
- *default_Country_Name* is the name of the default country in the language of the locale that you are adding, such as Vereinigte Staaten
- *Country* is the name of the country in the default locale's language, such as Germany

For example, to add country name translations for German:

```
insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
DISPLAY_NAME, ACTIVE_FLAG) values ('de_DE', 'de_DE', 'Deut-
schland', 'Y')
insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
DISPLAY_NAME, ACTIVE_FLAG) values ('de_DE', 'en_US', 'Vereinigte
```

```
Staaten', 'Y')
insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
DISPLAY_NAME, ACTIVE_FLAG) values ('en_US', 'de_DE', 'Germany',
'Y')
```

- b. If there are several locales installed on your system, add a country name translation for the new locale to each of the existing locales, and add a country name translation for each existing locale to the new locale. For example, if you already support the fr_FR locale and are adding support for the de_DE locale, run the following SQL:

```
insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
DISPLAY_NAME, ACTIVE_FLAG) values ('fr_FR', 'de_DE', 'L'Alle-
magne', 'Y')
insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
DISPLAY_NAME, ACTIVE_FLAG) values ('de_DE', 'fr_FR', 'Frankreich',
'Y')
```

To check the results of the INSERT commands, run the following SQL command:

```
select * from CMGT_LOCALE_NAMES
```

10. In a command window, navigate to your *sdk_home*\workspaces\project-name directory and run the following command:

```
WEB-INF\scripts\loadI18NFromXML.bat I18N jdbc_driver (for Windows
Systems)
```

or

```
WEB-INF\scripts\loadI18NFromXML.sh I18N jdbc_driver (for Unix Sys-
tems)
```

where *jdbc_driver* is the full pathname of your JDBC jar file.

For example:

```
WEB-INF\scripts\loadI18NFromXML.bat I18N oraclejdbc.jar
```

11. In a command window, navigate to your *sdk_home* directory and build the deployable (.war file) image:

```
sdk distWar
```

12. Deploy the .war file to your servlet container.

13. Navigate to the **projects\project_name\dist\timestamp-WAR** directory, where *timestamp* has the form YYYYMMDD and is the date on which you issued the `sdk distWar` command. Rename the **prefs_dev.xml** file to **prefs.xml**, then copy it to the home directory of the user who is running the servlet container: **user_home/cmgt/debs/conf/** directory.
14. Restart your servlet container.
15. Verify that the installation succeeded:
 - a. Navigate to your implementation home page. The URL is similar to:
`http://<server>:<port>/Sterling/en/US/enterpriseMgr/matrix`
 - b. Log in as an administrator user, then click My Account.
The User Detail page displays.
 - c. The Preferred Locale drop-down list in the User Locale panel should include the locales that you just installed.
 - d. Choose a locale from the drop-down list, click **Save**, log out, and log back in.
The administrator user's home page displays with localized text. You can now create users who use the new locales.

This chapter covers:

- "General Architectural Concerns" on page 169
- "Securing Users" on page 173
- "SSL support" on page 173
- "Installing Certificates for SSL" on page 176
- "Storing Data in Encrypted Form" on page 177
- "Password Policies" on page 189
- "Cross-Site Request Forgery Filter" on page 189

General Architectural Concerns

When you design your implementation environment, you should bear in mind the physical and network configuration of your data center, and your security policies to determine what people can perform what activities. In particular, you must distinguish carefully between what a person can do as an administrator in your data center environment, and what a person can do as a Visual Modeler user.

Administration Model

This section describes the entities assumed to be present in the administrative domain (the data center) in which the Visual Modeler resides, including networks, servers, and administrative roles. This is likely not an exhaustive list. It is likely that various network devices will exist within this environment, and perhaps other servers.

Networks

The following network zones are assumed to exist. These networks are connected to themselves as outlined below through gateways.

- **External network:** Directly visible from the internet. It hosts the Web servers and static content. The External network is accessible to the internet through a firewall. It is assumed that the firewall and appropriate standard security practices are sufficient to prevent shell level access from the internet. The External network has a gateway to the De-Militarized Zone (DMZ) that permits highly controlled access from the Web server to the application server(s).
- **DMZ:** This network is not directly visible from the Internet. A constrained gateway permits the Web server(s) residing on the External network to access the Application server(s) residing on this network and another, similar, gateway permits access from the DMZ network to the Internal network. The Web server routes messages to application servers through a dedicated port.
- **Internal network:** Not visible from the Internet, nor from the External network. Database resources reside here. Application servers in the DMZ connect to Database servers in this network through a constrained gateway.

Servers

The following servers are assumed to exist. The term server here indicates a software application that is more or less continuously listening on one or more network ports responding to requests received on the ports. Software servers, of course reside on computer hardware. Generally, though not necessarily, there will be a one-to-one relationship between a server software system, and a server hardware entity.

- **Web server** resides in the External network. It responds to HTTP (possibly using SSL) requests from the Internet or internal corporate Intranets.

- Application server resides in DMZ. Some http and https requests are delegated to the Application server for dynamically generated response. The Application server maintains connections with the Database server.
- Database server resides in the Internal network.

Roles

This section describes roles within the administrative context of the Visual Modeler. These are roles assigned to data center personnel acting as employees or agents of the Enterprise. They are distinguished from the roles of individuals who interact directly with the Visual Modeler (“Online Users”). Online users have capabilities managed directly by three Visual Modeler Entitlement services. Dispatch (or “MessageType”) Entitlement Service manages page flow privileges. The Access Policy Service and ACL Service together manage fine-grained data-level access.

- Database Administrator
 - Responsible for Database servers.
 - Can log into database server.
 - Can read, create, update, or delete databases, database tables, indexes, and other database resources.
 - Can create backups and restore from backups.
 - Can create Database users and manage them.
 - Does not have root level authority in server operating system.
 - Does not have direct access to Application server machine (or DMZ).
 - Does not have access as Visual Modeler user.
- System Administrator
 - Responsible for server hardware, and server software.
 - Has root access to server machines within his/her zone of responsibility.
 - Has the authority to start and stop server processes.
 - As root, can read, write, update, or delete files in file systems.
 - Can back up and restore files.
 - Can create operating system level users and manage them.
 - Does not have access to log in to database server.

- Does not have access as Visual Modeler user.
- Developer
 - Responsible for preparation of deployment Web archives (WAR files).
 - Has the authority to create Web archives representing the Visual Modeler executable.
 - Can set properties and business rules governing Visual Modeler operation, including properties that configure access to the database, properties that configure the JCE Key store, and so on.
 - Has the authority to create or modify the initial Visual Modeler dataset. This dataset is a part of the deployment archive.
 - Does not have any kind of access to the Production Database server or Application servers.
 - Does not have access to the production Visual Modeler as a Visual Modeler user.
 - Does not move code from development and QA environments to production.
- Network Administrator
 - Configures and manages network.
 - Has authority to create and assign network resources, including domain names, IP addresses, firewall policies, and so on.
 - Does not have Database server access.
 - Does not have access as Visual Modeler user.

Data Center Roles

The following are assumed about data center administrative roles:

1. Roles are segregated. System Administrators cannot be Developers, Network Administrators, nor Database Administrators. Similarly, Database Administrators cannot be Developers, System Administrators, nor Network Administrators, and so on.
2. System Administrator Roles should be partitioned on network boundaries. A system administrator for the DMZ should not be a system administrator for the Internal network.

3. Data center administrators do not have Visual Modeler userids with administration roles.

Securing Users

When you load either the minimal or reference data set into the Visual Modeler, you create two enterprise users: admin and ERPAdmin. Before permitting the Visual Modeler to go live in production, you must change the passwords of both users. If you do not do this, then it represents a serious security hole in your application.

You can change the passwords by logging in after the Visual Modeler is started using the administration interface.

SSL support

The Visual Modeler supports communication using the SSL protocol between a user's browser and the Visual Modeler. In particular, as part of your implementation of the Visual Modeler, you must consider which pages (if any) should be SSL-protected.

This section discusses SSL support in the Visual Modeler.

If you are not using SSL in your implementation of the Visual Modeler, then you do not need to change the out-of-the-box port settings for non-SSL and SSL ports. If you are using SSL to protect some or all access to the system, set port-related properties in the **Comergent.xml** template file, located in **templates/WEB-INF/Comergent.xml**. Use the following guidelines to set the port-related properties:

- If you are using the standard ports (80 for non-SSL access and 443 for SSL access, respectively) for both schemes, then set the port settings to "." for both.
- If you are using non-standard ports for one or the other scheme, then you must explicitly set the port number for each.

Setting Up Secure Message Types

You can choose whether a group of message types require SSL access or not. A particular web site can have mixed secure and non-secure access requirements. The access requirements follow the type of request received by the JSP page. If only a section of a web site requires SSL, you must set the relevant message type to

require SSL/HTTPS. You must also specify where users revert to non-SSL/HTTPS access.

When a user clicks a link on a Sterling web application JSP page, the JSP page looks up the link in a repository to see if the link requires SSL. If it does, the JSP page generates the link as "https://" and SSL handles all the necessary encoding.. Otherwise, the JSP page generates the link as "http".

You can ensure that pages require SSL access by setting their associated message types to a security level that requires "https". When a link is generated using the *link()* methods provided by the Visual Modeler, then the command parameter is used to identify the message type, and the SecurityLevel child element is used to ensure that the appropriate schema (http or https) is reflected in the URL.

The SecurityLevel element can be used at any level in the message type group and type hierarchy. Specifying it at a message type group level means that all message types that belong to the group inherit the security level unless it is overwritten at a lower level in the hierarchy.

The Level attribute of the SecurityLevel element can take the following values:

- any: both http and https can be used to access the message type. This is the default value if nothing is specified.
- useHttp: can be accessed by http and subsequent URLs will be generated with http.
- useHttps: the page may be accessed without https, but any URL generated by the Visual Modeler will specify https.
- requireHttps: any URL requesting this message type must use https, otherwise the request is rejected and an error page is displayed. Ensure that wherever this message type is used, it is used in the *link()* method to form the link to the message type, especially in any forms which use this message type.

For example, suppose that the following message type is declared in **MessageTypes.xml**:

```
<MessageType Name="SysUserDetailDisplay">
  <SecurityLevel Level="requireHttps" />
  ...
</MessageType>
```

Then link("partnerMkt", "SysUserDetailDisplay") will generate the URL:

```
https://<server>:<serverSSLPort>/Sterling/partnerMkt/
matrix?cmd=SysUserDetailDisplay
```

By default, that is if no `SecurityLevel` is specified for a message type or for any group to which the message type belongs, the `link()` methods will generate URLs using the same protocol used to access the referring page.

You can use the `SecurityLevel` element to specify that `https` is required to access a given page by setting the security level to “`requireHttps`”: the Visual Modeler verifies that each message type is being accessed using the appropriate protocol.

You must set the `C3_Commerce_Manager.General.ServerSSLPort` element in the **Comergent.xml** file to the appropriate value for your servlet container. If the servlet container is set up to use the standard SSL port (443), then you do not have to specify it. Consult your servlet container documentation for any steps that are required to set up a port to accept SSL connections.

Example Usages

In these examples, we suggest message groups and types that might be candidates to protect using the SSL protocol. In general, you need to determine the possible page flows that users can perform and identify the entry and exit message types that surround the area to be SSL-protected.

Protecting the Authenticated Environment

You should consider protecting the entire Web experience of the Visual Modeler presented to authenticated users. You can do this by adding:

```
<SecurityLevel Level="requireHttps" />
```

to the `EnterpriseHomeGroup` and `PartnerHomeGroup` message groups.

Protecting the Enterprise Environment

Suppose that you want to protect your enterprise administration pages behind the `https` schema. You can add:

```
<SecurityLevel Level="requireHttps" />
```

to the following message types:

- `LoginDisplay`
- `GenericLoginDisplay`
- `HomePageDisplay`

Protecting Credit Card Information

Users are highly sensitive to passing credit card information over the Web, and you may be required to ensure that any requests that include credit card information are SSL-protected. For example, users may enter credit card information when they

edit an order header. The URL used to submit the order header information uses the OILAddrChangeProcess message type. By adding:

```
<SecurityLevel Level="useHttps" />
```

to the OILDisplayGroup message group, you ensure that when URLs are formed with message types from this group that they specify the https schema.

Protecting User Administration Pages

You should consider using the SSL protocol to protect pages in which users enter personal information. For example, add:

```
<SecurityLevel Level="requireHttps" />
```

to the UserAdminGroup message group.

Installing Certificates for SSL

To support SSL communication, you must ensure that you have determined the level of security you want to support between users and your Visual Modeler and between the Visual Modeler and any of your partners' enterprise servers. You can enable SSL communication between users' browsers and the enterprise server and between the enterprise server and one or more of your partners' enterprise servers.

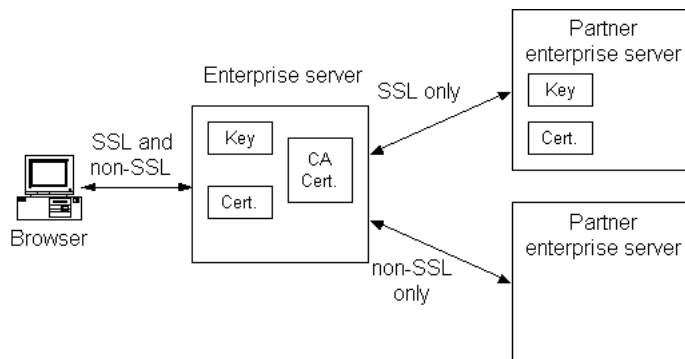


FIGURE 7. SSL Communication in the Visual Modeler

Overview

When two systems elect to use the SSL protocol to communicate, the entity that initiates the communication is referred to as the client and the other entity is the server. For example, if the enterprise server is set up to support SSL communication with users' browsers, then the browser acts as the SSL client and the enterprise

server is the SSL server. If the enterprise server and a partner's enterprise server use SSL to transfer price and availability requests, then the enterprise server is acting as the SSL client and the partner's enterprise server is the SSL server.

The SSL protocol requires that the SSL client maintains certifying authority certificates from certificate authorities from whom they accept certificates. When a client attempts to open an SSL communication session with an SSL server, the server must send to the client its certificate and key. The client may choose to verify against the certifying authority certificate, and then uses the key to encrypt messages sent back to the server.

A client makes a request to open an SSL communication by posting to the server using the https schema: that is, by using a URL of the form

```
https://myserver.com:<SSL port>
```

Consequently:

- If you want your installation of the enterprise server to support SSL communication from Web browsers, then you must obtain a server certificate and server key and use the appropriate servlet container mechanism to specify an SSL port.
- If you want to enable SSL communication between an enterprise server and a partner's enterprise server, then the partner's enterprise server must obtain a server certificate and server key. The partner's enterprise server must be configured to receive messages from your enterprise server on an SSL port. Your enterprise server must be configured to accept the SSL certificates offered by the partner server. In general, if the partner server's certificates match the domain of the partner server's URL, then the enterprise server accepts them automatically.

Storing Data in Encrypted Form

This section covers:

- "General Setup" on page 178
- "Changing Encryption Algorithms" on page 182
- "Key Stores and System Initialization" on page 184
- "Wrapper Classes for Standard Algorithms" on page 185
- "Key Rotation" on page 185
- "Password Policies" on page 189

The Visual Modeler lets you store sensitive business data in the Knowledgebase in an encrypted form. This is done by setting the Encryption attribute of the corresponding DataElement to “1-way” or “2-way”.

Attention: If you deploy the reference implementation of the Visual Modeler without making changes to the schema, then note the following:

Credit card numbers stored as part of a user’s profile are 2-way encrypted. This means that a **dcmsKey.ser** file is created on your system.

If you upgrade the Java Virtual Machine (JVM) at any time after creating encrypted data, then you should check that the data can be retrieved correctly. For example, if upgrading the JVM means that the **.jceKeystore** file is regenerated, then 2-way encrypted data will have to be recovered.

Securing data in the persistent storage system should be considered when the data travels over insecure internal or external networks or when access to the database server cannot be restricted to authorized individuals. In general, it is preferable to secure data using the facilities provided by the database server. These are likely to provide a more comprehensive and higher performance solution.

You must decide which fields are to be encrypted before loading the data and creating more data. In general, you cannot change to using encrypted data for a data field after any data objects have been created of that type.

Attention: You cannot encrypt data that is used in Sterling Analyzer reports without breaking the reports that use the data. You can identify which database columns are used in reports by reviewing the view creation scripts. All report data is accessed using views and so the view scripts provide a complete list of the columns accessed by reports.

General Setup

This section describes the basic steps to implement data encryption.

Note: Release 6.3 introduced more flexible encryption support than previous releases. If you are working on an earlier release, see "Password Policies" on page 189.

You should use JDK 6 or a subsequent compatible version because this has the Java Cryptography Extension (JCE) built in.

1. You should download the Unlimited Strength Jurisdiction Policy Files 1.5 available from the SUN Java Web site. Follow the instructions provided by the **Readme.txt** file to install the JCE jurisdiction policy JAR files into your Java environment.

2. To encrypt data that is to be stored in the Knowledgebase, you must specify the encryption method that is to be used:
 - For two-way encrypted fields this must be a symmetric encryption algorithm so that the data can be retrieved in its unencrypted form.
 - For one-way encrypted fields, you must use a digester. This must effectively provide a highly probable guarantee that if two source strings are digested and the digested strings are the same, then the source strings must have been the same to start with.

At any one time, the data services layer determines that only one active symmetric encryption algorithm and only one active digester can be in use.

- You can switch from one symmetric encryption algorithm to another as your encryption needs change. Data encrypted using an earlier encryption algorithm can be retrieved, and if it is re-saved, then it is persisted using the appropriate active symmetric encryption algorithm. See "Changing Encryption Algorithms" on page 182 for more information.
- Once you have selected your digester, then this cannot be changed. By its nature, data encrypted using a digester cannot be retrieved in order to re-encrypt it using a different digester.

You cannot change the status of data fields from encrypted to unencrypted or the other way round. In summary:

1. Decide which data fields are to be encrypted. You cannot add or remove fields from this list once it is set up and data objects have been persisted using the encryption methods.
2. Decide which of these fields are to be one-way encrypted and which are to be two-way encrypted.
3. Select a secure digester to be used for one-way encryption. You must keep this digester.
4. Select a secure symmetric encryption algorithm: you can change this later if your encryption needs change.

Symmetric Encrypter

You declare the active symmetric encryption algorithm using the `TwoWayEncrypter` element in the **DataServices.xml** file. For example:

```
<TwoWayEncrypter>DefaultEncrypter</TwoWayEncrypter>
```

The value of this element must match the Name attribute of an Alias element or a SymmetricEncrypter element declared in the **CryptographyService.xml** configuration file. For example:

```
<Alias Name="DefaultEncrypter" OriginalName="InlineDES">
  <Description>Alias to the default encrypter.</Description>
</Alias>
```

The OriginalName attribute points to the SymmetricEncrypter element that defines the encrypter:

```
<SymmetricEncrypter Name="InlineDES"
  Class="com.comergent.cryptography.JCESymmetricEncrypter"
  KeyManager="InlineKeyManager" KeyName="myDesKey" Tag="IDES">
  <Description>
    DES Encrypter using key from inline key store.
  </Description>
  <Algorithm Name="DES" />
</SymmetricEncrypter>
```

Alternatively, you can reference a symmetric encrypter directly by its name. For example:

```
<SymmetricEncrypter Name="JCE DES" Tag="DES"
  Class="com.comergent.dcm.cryptography.JCESymmetricEncrypter"
  KeyManager="JCEKeyManager" KeyName="myKey">
  <Algorithm Name="DES" Provider="SunJCE"/>
</SymmetricEncrypter>
```

Each SymmetricEncrypter element declared in the **CryptographyService.xml** file can be used to encrypt and decrypt data.

- The Name attribute in the element is used to identify the symmetric encrypter in the TwoWayEncrypter element. The Name attribute must be unique among all the SymmetricEncrypter and Digester elements declared.
- The Tag attribute of the SymmetricEncrypter element is used to prefix the encrypted strings in the persistent data store. The Tag attribute must be unique among all the SymmetricEncrypter elements declared, but the same Tag value can be used for a Digester. For example, if the string “ajones” is encrypted to “hg\$y&7606(7gfj)” by the symmetric encrypter whose tag is “DES”, then the value stored in the database is “DES:hg\$y&7606(7gfj)”. In this way, each stored encrypted value provides an indication of which symmetric encrypter can be used to decrypt it.

- The Class attribute of each SymmetricEncrypter element specifies the class to be used to perform the encryption and decryption: this class and its dependent classes must be in the Visual Modeler classpath. The specified class must implement the `com.comergent.api.dcm.cryptography.SymmetricEncrypter` interface.
- If the symmetric encryption algorithm requires a key manager, then the SymmetricEncrypter element also specifies its key manager using the KeyManager attribute. The value of this element must match the Name attribute of one of the declared KeyManager elements.

```
<KeyManager Name="JCEKeyManager"  
  Class="com.comergent.dcm.cryptography.JCEKeyManager">  
  <Algorithm Name="DES"/>  
</KeyManager>
```

Each KeyManager element in the **CryptographyService.xml** configuration file declares the class to be used to manage keys for a symmetric encrypter. Typically, these classes are used to access keys managed in a key store. The name of the key to be retrieved from the key store is specified by the KeyName attribute of the SymmetricEncrypter element. Thus, by having two SymmetricEncrypter elements declaring the same Class attribute, but different KeyName attributes, you can use different keys to encrypt data. See "Key Stores and System Initialization" on page 184 for more information about key stores and how keys can be retrieved when the Visual Modeler is starting up.

Digester

You declare the active digester using the OneWayEncrypter element in the **DataServices.xml** file. For example:

```
<OneWayEncrypter>MD5</OneWayEncrypter>
```

The value of this element must match the Tag attribute of a Digester element declared in the **CryptographyService.xml** configuration file.

Each Digester element declared in the **CryptographyService.xml** file can be used to encrypt data.

- The Name attribute in the element is used to identify the digester in the OneWayEncrypter element.
- The Tag attribute is used to prefix the encrypted strings in the persistent data store. The Tag attribute must be unique among all the Digester elements declared, but the same Tag value can be used for a SymmetricEncrypter element. For example, if the string "ajones" is encrypted to "Ta\$y&%lN7gL5" by the digester whose tag is "MD5", then

the value stored in the database is “MD5:Ta\$y&%\N7gL5”. In this way, each stored encrypted value provides an indication of which digester was used to encrypt it.

- The Class attribute of each Digester element specifies the class to be used to perform the encryption: this class and its dependent classes must be in the Visual Modeler classpath. The specified class must implement the `com.comergent.api.dcm.cryptography.Digester` interface.

Default Symmetric Encrypter and Digester

By default, Release 7.0 and higher of the Visual Modeler uses the one-way and two-way encryption schemes used in the Sun JCE implementation: these reference MD5 and DES respectively. Earlier releases used the legacy encryption schemes provided by the crysec packages. See "Password Policies" on page 189 for further information. These schemes are identified by empty Tag attributes (that is, Tag=“”).

Note:	You should consider replacing the legacy digester with either SHA or MD5 digesters. Both offer a higher level of security against cryptographic attack. However, you must make the decision to change to a different digester before implementing the Visual Modeler. See "One-Way Encrypted Data" on page 183.
--------------	---

Changing Encryption Algorithms

You can change the symmetric encryption algorithm used if your encryption needs change. If you do this, data that has previously been encrypted using the earlier symmetric encryption algorithm is not lost.

Note:	You must use JDK 6 or a subsequent compatible version to use AES encryption.
--------------	--

Two-Way Encrypted Data

Suppose that a data field of a data object is marked for two-way encryption and the active symmetric encryption scheme is set to “DES”. If a new data object is persisted, then the data field value is set to something like “DES:hfd8kUH9*”. Suppose that you decide to switch to using the symmetric encryption algorithm identified by the Tag value “AES”, and so you modify the

CryptographyService.xml file to declare a new default encrypter. Specifically, modify the Alias element whose Name attribute is `DefaultEncrypter` so that the `OriginalName` attribute is set to “JCE_AES”. For example:

```
<Alias Name="DefaultEncrypter" OriginalName="JCE_AES">  
  <Description>Alias to the default encrypter.</Description>
```

If new data objects are persisted, then their data is encrypted using the AES symmetric encryption scheme. If the earlier data object is restored, then the encryption service recognizes that the field was encrypted using the DES encryption schema, and invokes the corresponding symmetric encrypter class to decrypt it. If the restored data object is subsequently persisted, then the AES scheme is used to perform the encryption and the value of the data field will be something like “AES:8(HH\$DyGK” in the persistent data store.

You can also perform “key rotation” to update all your encrypted data to make use of a new key. See "Key Rotation" on page 185 for more details.

One-Way Encrypted Data

Data that is encrypted using a digester is not intended to be used to retrieve the original value. You cannot easily change digesters once data has been encrypted using your choice of digester.

For example, suppose that you choose to one-way encrypt user passwords in the Password field of the UserContact data object, and suppose that you have chosen to use the legacy digester for this purpose. If subsequently you decide to change digesters to the more secure SHA or MD5 digesters, then you would have to proceed along these lines:

1. Notify users that their passwords will be changed at a certain date.
2. At that date, stop the Visual Modeler, and set the OneWayEncrypter element to the new Digester name. Suppose that the Tag for the new digester is “SHA”.
3. Generate a new password value for each user: say, their username and a randomly selected integer: for example, “ajones67854”.
4. Offline, use the new digester to encrypt the new password for each user: suppose that “ajones67854” is encrypted to “hjkYF*&5NF0”.
5. Using a SQL script, update the CMGT_USER_CONTACT table to enter the encrypted form of their password for each user:

```
UPDATE CMGT_USER_CONTACTS SET PASSWORD = 'SHA:hjkYF*&5NF0' WHERE  
USER_NAME = 'ajones';
```
6. Restart the Visual Modeler.
7. Notify each user by email that their password has now changed, and give them the new unencrypted value. Ask each user to log in using their new password, and ask them to change their password immediately.

Note that any other data in the Visual Modeler that was also one-way encrypted (such as credit card numbers) would be rendered inaccessible and would have to be re-created if required.

Key Stores and System Initialization

Almost all encryption schemes use a key store to hold the keys used to encrypt and decrypt data. In symmetric schemes the same keys are used to both encrypt and decrypt data. Consequently, it is important that you take great care to protect your key stores and ensure that they are not corrupted or deleted.

Attention: Loss or corruption of your key store can lead to complete loss of your encrypted data.

Each encryption scheme make use of different types of keys and key stores, and you must consult the documentation that comes with your choice of encryption scheme carefully. Typically, the process is to create a key store, and then generate keys that you add to the key store. Each key has a name that is used to retrieve the key from the key store.

When the Visual Modeler is started or re-started, it must retrieve the appropriate keys from a key store. If it fails to do so, then the Visual Modeler fails to initialize and will not permit any logins.

Key stores and keys in the key store can be encrypted. If you choose to encrypt either, then as part of the initialization process, the cryptography service must decrypt them to retrieve the keys for the symmetric encrypters.

Attention: Take extreme care in encrypting key stores. A key store is effectively impossible to decrypt without the appropriate passwords. Data will be impossible to retrieve if the keys in the key store are inaccessible.

The cryptography service is initialized during the initialization of the InitServlet class. It attempts to decrypt the key stores and keys for each symmetric encrypter using the password “Comergent”. If it fails for one or more of the symmetric encrypters, then the initialization of the Visual Modeler stops at this point. Requests posted to the DispatchServlet (the main servlet class used to process requests), are sent a 503 response: Service Unavailable.

You can complete initialization of the cryptography service and hence of the whole Visual Modeler by posting a request to the InitServlet that includes the relevant parameters: typically, for each symmetric encrypter the passwords used to encrypt the key store and the key. For example if one symmetric encryption scheme requires a keyStorePassword parameter and a keyPassword parameter, and a

second scheme used just a storePassword parameter, then the following post would provide the initialization information:

```
http://<machine:port>/Sterling/init?keyStorePassword=password&key-  
Password=password&storePassword=password
```

Wrapper Classes for Standard Algorithms

SymmetricEncrypter Class

The JDK 6 provides implementations of standard symmetric encryption algorithms such as AES and DES. The Visual Modeler enables you to use these through the JCESymmetricEncrypter class. This class implements the SymmetricEncrypter interface and so may be specified in the SymmetricEncrypter element.

Digester Class

The JDK 6 provides implementations of standard digester algorithms such as MD5 and SHA. The Visual Modeler enables you to use these through the JCEDigester class. This class implements the Digester interface and so may be specified in the Digester element.

Key Rotation

Key Rotation Procedure

This procedure describes how to rotate encryption keys that protect data on the Visual Modeler. The procedure is designed so that it can be executed as needed or incorporated as an operating system level cron job. It must be executed from a shell on one of operation systems because it references files located on these systems.

Background

The Visual Modeler can be configured to efficiently encrypt selected persistent data using any of a variety of Symmetric Encryption algorithms. The purpose of this feature is to protect confidential data from internal users who may need access to the Visual Modeler Knowledgebase.

The encryption keys are stored in a password-protected keystore. A keystore is usually a file in the filesystem of the application server. The identical keystore must be present on all members of a clustered system. This can be arranged by placing the keystore on a shared filesystem or by copying the keystore by hand whenever it changes.

A Visual Modeler configuration file maintains a mapping between keystore keys and internal logical “encrypters”. This file, named, **CryptographyService.xml**, is

colocated with preferences files, usually in *user_home/cmgt/debs/conf/*. Again, it must be identical among members of a cluster.

The Visual Modeler tags encrypted values. By this means it knows which encrypter encrypted a particular value.

Why Rotate?

The Visual Modeler is designed to use a standard encrypter using an internal keystore during development and deployment. This standard encrypter encrypts using 56-bit DES encryption, which is adequate for those purposes. It is not, however recommended for a production system for two reasons: the keystores on the production systems should not be shared with development systems, because this compromises the data protected by the keys, and production systems should have stronger encryption.

Many customers may want to rotate keys at regular intervals, to limit the window of vulnerability from any compromise. If keys are rotated on a monthly bases, for example, then the payoff from acquiring a key is limited to that month.

Basic Process

The basic key rotation process has three steps:

1. Create a new encrypter by cloning an existing encrypter.
2. Change the default encrypter reference to point to the new encrypter.
3. Update values stored in the database to use the new encrypter. This step may take some time, depending on the number of encrypted values in the database, but can be accomplished over time, since any old values will continue to be decryptable with the old encrypter and new values will employ the new encrypter.

Detailed Process

Begin by creating a new key.

1. Log in as the user running the servlet container.
2. Open a shell window and change directory to the home directory. We refer to this directory as *user_home*.
3. You must select a new unique name for the new encrypter and key, and a new tag. These can all be the same. The tag should be short. We suggest these incorporate temporal information or sequence number. For example, “AESyymm”, where “yy” is the year and “mm” the month. You must also

determine the encryption algorithm: AES or DESede (“triple DES”) are standard for business applications.

4. Locate the tool **cmgt-cryptography-tool.jar** in the file system. If you have installed Release 9.0 into the SDK, the JAR file is in *sdk_home/releases/debs-9.0/image/install/*.

5. Copy this JAR file to *user_home*.

6. The base configuration provides a standard selection of encrypters that are likely to meet most needs. Therefore, creating a new encrypter means cloning an existing encrypter in most cases. To see the configured encrypters for your system, execute:

```
java -jar cmgt-cryptography-tool.jar list
```

7. To create a new encrypter, execute, for example:

```
java -jar cmgt-cryptography-tool.jar clone-encrypter JCE_AES  
AES0805
```

The above command will clone the JCE_AES encrypter and name it AES0805. The tag and key names will also be set to AES0805 in this example, but if a different tag or key name is desired it can be set on the command line.

8. You can verify the new encrypter by executing:

```
java -jar cmgt-cryptography-tool.jar info AES0805
```

You will see something like this:

```
AES0805: Name=AES0805 Tag=AES0805 initialized=true algorithm=AES  
keymanager=JCEKeyManager keyname=AES0805 provider=SunJCE
```

Changing the Default Encrypter

9. To change the default encrypter, we need set the alias to the new encrypter:

```
java -jar cmgt-cryptography-tool.jar alias DefaultEncrypter  
AES0805
```

If you are working with a clustered configuration with no shared filesystem, then the keystore and configuration files must be manually synchronized.

10. To do this, you must first activate the key. Do this by encrypting something:

```
echo "hello" | java -jar cmgt-cryptography-tool.jar encrypt
```

11. You must copy the keystore and configuration file to each of the cluster members. The cryptography service configuration file is colocated with Sterling preferences and is named **CryptographyService.xml**. By default, this

is in *user_home/cmgt/debs/conf/*, where *user_home* is the home directory of the user that starts the Visual Modeler Web application. The location of the keystore can be identified with the following command:

```
java -jar cmgt-cryptography-tool.jar info JCEKeyManager
```

Note that the location of the keystore may be specified by using the `KeyStorePath` attribute of the appropriate `KeyManager` element in the **CryptographyService.xml** configuration file. For example:

```
<KeyManager Class="com.comergent.cryptography.JCEKeyManager"
  Name="JCEKeyManager" KeyStorePath="cmgt/debs/conf/.keyStore">
</KeyManager>
```

The path may be specified as a path relative to the *user_home* directory or as an absolute path.

12. In order for this change to take effect it is currently necessary to cycle the servlet container. Use techniques appropriate to your servlet container to restart it.

Updating Existing Database Ciphers.

At this point, all new values for encrypted columns in the database will be encrypted with the new encrypter, but values stored with the old encrypter are unmodified. We need to update these values. In particular, passwords protected by encryption (`Encryption="2-Way"` in the schema file) will no longer work.

13. First identify all primary beans that have encrypted properties. This can be accomplished by identifying Elements in `DsDataElements` with `Encryption="2-Way"`, then searching the data object schema files for `<DataElement>` references to those elements. More than one bean may refer to the same encrypted column. For example, the `UserContactBean` and the `UserBean` both refer to the `CMGT_USER_CONTACTS` table and the `PASSWORD` column within it. In this case, we obviously need only update one of these bean types.
14. Locate the cipher-update script in your installation: **cipher-update.sh** or **cipher-update.bat**, depending on your shell. If you have installed Release 9.0 into the SDK and built your project, then you will find the script in *sdk_home/workspaces/project/WEB-INF/scripts/*.
15. Copy the script to the appropriate runtime location:
 - a. If you are working in the SDK, then copy the script to *sdk_home/builds/project/*.

- b. If you are working in a running deployment environment, then copy the script to *debs_home/Sterling/*.
16. For each primary bean that has encrypted properties, execute the cipher-update script. You must provide the name of the bean and the name of the key field used to retrieve each instance of the bean. For example:

```
cypher-update OrderBean ShoppingCartKey
```

or

```
cypher-update UserContactBean Userkey
```

This may take a long time if the database is very large, so this step is best executed during a period of low Web activity.

Password Policies

The Visual Modeler supports the ability to specify password policies. These are used to determine how passwords are created, criteria that passwords must satisfy (such as minimum lengths), and how often passwords must be changed.

Cross-Site Request Forgery Filter

The Visual Modeler supports the ability to prevent Cross-Site Request Forgery (CSRF) by configuring a servlet filter that invalidates the current session when it detects a potential CSRF request. You can configure the filter using the following init-parameters in the **web.xml** file located in the **WEB-INF** directory.

- `noop`: set the value of this parameter to true to disable the filter. The default value of this parameter is false.
- `checkOnly`: set the value of this parameter to true to log a potential CSRF request but not invalidate the current session. The default value of this parameter is false.
- `noCheckList`: set the value of this parameter to a comma-separated list of `messageTypes` you want to skip while checking.

Note: Ensure that the JSPs associated with the "white-list" commands do not contain any known holes, particularly those related to cross-site scripting.

This chapter provides a description of the tests that you can perform once implementation is complete.

Starting the Visual Modeler Server

In general, you can start the Visual Modeler by starting the servlet container in which the Visual Modeler is installed. The order in which the servlets load is specified in the Visual Modeler Web application **web.xml** file and you can read this file in any text editor.

As the Visual Modeler starts, the servlet console window displays preliminary logging information. Once the Visual Modeler has initialized its logging environment, then it uses the logging methods to record events.

Troubleshooting

This section covers some basic steps that you must perform to ensure that the system starts correctly. This list is not comprehensive; rather it covers some check points that are a common source of problems. In general, you should troubleshoot your installation using the SDK to ensure that any modifications you make are contained in your project directory.

To Perform Pre-startup Checks

1. Review the **prefs.xml** configuration file. Check that it is in the correct location as this is the most frequent cause of problems on startup. Remember:
 - a. By default, the location of this file is assumed to be **user_home/cmgt/debs/conf/** where **user_home** is the home directory for the operating system user running the servlet container.
 - b. This location can be overridden by:
 - Either: specifying the location of the file as a system property:
`-Dcomergent.preferences.store=/path/prefs.xml`
 - Or: specifying its location using the `comergent.preference.store` parameter in the Visual Modeler **web.xml** configuration file:

```
<init-param>
  <param-name>comergent.preferences.store</param-name>
  <param-value>/path/prefs.xml</param-value>
</init-param>
```
2. Review the **Comergent.xml** configuration file. Check that:
 - It contains the value of system properties that you expect to see (or that are overridden by the **prefs.xml** configuration file).
3. Using the SDK, run the `generateDTD` target.
 - If you get a series of lines of the form: "Writing DTD for ACL...done!", then the DTDs have been successfully generated. Look in the **debs_home/Sterling/WEB-INF/bizobjs/** directory to verify that a complete set of DTDs are there.
 - If you get an error message, then review the steps outlined above.
4. Using the SDK, run the `generateBean` target. This should generate all the beans specified by the data objects. If you see any error messages, then you should fix their cause before proceeding.
5. Using the SDK, run the `merge` target. If this runs successfully, then run the `dist` target to generate the Web application WAR file.

Error Messages on Startup

When the Visual Modeler starts, you can see initialization information in either the console window or the servlet container log file. See CHAPTER 8, "Troubleshooting and Backing Up the Visual Modeler" for a summary of the most likely error messages, together with their causes and how to resolve them.

To troubleshoot problems with message types, you can set the `messageTypeValidate` element in the **Comergent.xml** file to “TRUE”.

Runtime Troubleshooting

This section covers some problems identified during testing.

TABLE 15. Troubleshooting Problems and Solutions

Problem	Solution
<p>On Solaris, the servlet container cannot find a certain servlet or URL.</p>	<p>First make sure that you did not make a typo. If you are certain that there was no mistake, then do the following:</p> <ol style="list-style-type: none"> 1. Run the following command on web.xml: <pre>java com.comergent.dcm.util.CheckWebXML web.xml > newWeb.xml</pre> 2. Edit the file newWeb.xml. Look for the following string <pre><!-- (8192) XXX BOUNDARY BREAK --></pre> The start of the comment <code><!--</code> is the start of a 8192 boundary break. If it falls within a value for an XML node, then that node will get truncated. A work around is to pad the web.xml file such that the boundary break will fall inside a comment. For more information, see the comments at the start of file CheckWebXML.java.

TABLE 15. Troubleshooting Problems and Solutions (Continued)

Problem	Solution
<p>You see parser errors such as:</p> <pre>java.lang.NoSuchMethodError at org.apache.xpath.DOM2Helper.ge tNamespaceOfNode (DOM2Helper.java:348) at org.apache.xml.utils.Tree- Walker.startNode (Tree- Walker.java:281) at org.apache.xml.utils.Tree- Walker.traverse (Tree- Walker.java:119) at org.apache.xalan.trans- former.TransformerIdentity- Impl.transform (TransformerIdentity- Impl.java:320)</pre>	<p>Check that you have followed the instructions to copy the XML parser-related JAR files to the servlet container's lib/ directory, and that you have removed any default parser.jar files.</p>
<p>Running iPlanet, you see the following in your browser:</p> <pre>GX Error (GX2GX) socket result code missing!!!</pre>	<p>There is a mismatch between the web.xml and ias-web.xml files. All servlets mentioned in web.xml must have a corresponding entry in the ias-web.xml file. Use the kguidgen utility to generate a GUID for the servlet.</p>

Communication Between Enterprise Servers

In testing whether an enterprise server can send price and availability requests and product inquiry list transfer requests to another enterprise server installed at a partner, check for the following problems:

1. Determine if the Message URL defined in the partner profile is correct.
 - On the enterprise server side, you can view the Message URL in the partner profile detail page: check that both the host name and port of the partner's enterprise server are correct. If it is correct, then check that the NamingManager entries of the **Comergent.xml** file connect to the same database specified in the **DataSources.xml** file.
 - If you see a message in the enterprise server log of the form:


```
XML message does not conform to the PriceAvailability.dtd
```

 then check the **debs_home/Sterling/WEB-INF/bizobjjs/** directory to see that the correct DTDs are present.
 - If you see an error displayed in the enterprise server browser window of the form:

ProcessingFailure

then the partner's enterprise server received the price and availability request, but for some reason failed to process it correctly.

On the partner side, by looking at the log or console window, check that the partner's enterprise server receives price and availability requests from the enterprise server.

2. If the partner's enterprise server shows no sign of receiving a price and availability request that you initiate from the enterprise server, then:
 - *Either* the Message URL is incorrect or it is not retrieved correctly through the NamingManager.
 - *Or* a network problem is preventing the enterprise server from connecting to the partner's enterprise server. From your enterprise server, point a browser to the partner's Message URL: if you cannot obtain a response from the partner's enterprise server, then a network problem is preventing the two enterprise servers from communicating.
3. If the partner's enterprise server log or console window indicates that the price and availability request has been received, but an error is generated in processing the request, then you should check that the partner's enterprise server has correct DTDs in its *debs_home/Sterling/WEB-INF/bizobjjs/* directory.

Installing a Clustered Implementation

This chapter describes how to set up the Visual Modeler in a clustered environment. It covers:

- "General Steps" on page 197
- "Setting Up a WebLogic Cluster" on page 203

In addition to following the steps described in one of the servlet container sections, you should also set up the global cache using JavaSpaces. See:

- "Setting up a Database for Caching" on page 210
- "Setting up JavaSpaces for Caching" on page 210

General Steps

Terminology and Overview

A cluster provides an environment that supports higher performance and reliability than a single machine can. Typically, a cluster comprises two or more member machines that from the outside world appear to work as one machine: when users submit a request to the cluster URL, they are not aware of which machine in the cluster processes the request and returns the response.

The cluster URL is usually directed to a Web server that sits “in front” of the cluster: this Web server provides the entry point for users, and it is responsible for distributing the requests to cluster members as requests come in. The Web server acts as a load balancer and distributes requests using an algorithm to determine which cluster member machine should receive each inbound request.

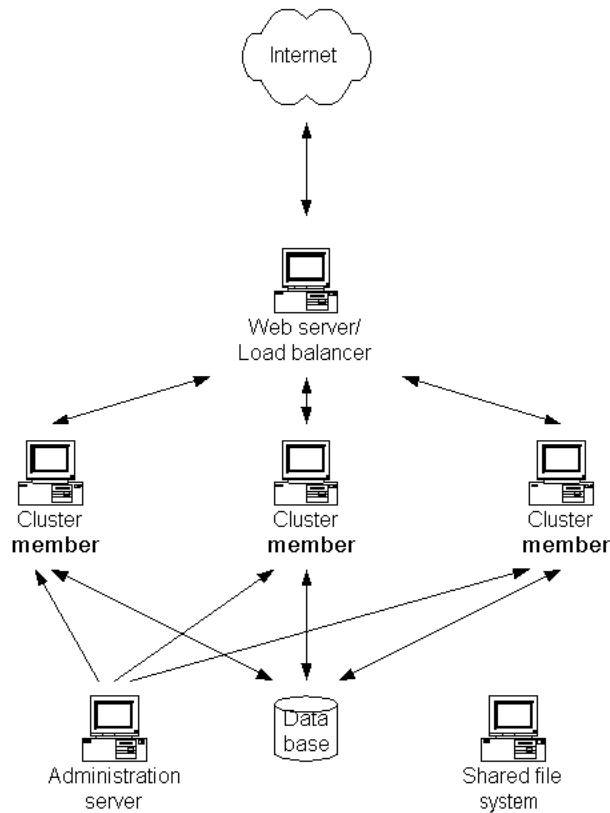


FIGURE 8. General Cluster Configuration

Administration Servers

In some cluster configurations, each cluster member is effectively independent of the others: you install the Visual Modeler into each cluster member and configure it independently of the other members of the cluster. Other cluster configurations make use of an administration server: this is a machine that manages the cluster.

Cluster members are typically registered with the administration server and the administration server maintains a single image of the Visual Modeler. When a machine joins the cluster, the administration server pushes a copy of the Web application to the new cluster member. In this case, each cluster member has the same configuration information because it has been pushed to them from the administration server.

The Visual Modeler uses Ecache to provide the notification mechanism required to synchronize cluster members.

Shared Files

To ensure that cluster members behave consistently with each other, they must access configuration files, templates, and image files that are common to all members of the cluster. You do this by establishing a shared file server and point to a common location on this file server.

- On UNIX systems, use an NFS file system to share common files. For example:

```
<context-param>
  <param-name>WritableDirectory.share.public.loadable
  </param-name>
  <param-value>/usr/Comergent/shared</param-value>
</context-param>
```

- On Windows systems, use one of two methods to set up a shared file server.
 - Using one method, on each cluster member you map the same drive letter to the shared file server, then use the drive letter to provide a common reference to the location of the shared files. For example:

```
<context-param>
  <param-name>WritableDirectory.share.public.loadable
  </param-name>
  <param-value>T:/Comergent/shared</param-value>
</context-param>
```

Here, the T: drive on each machine has been mapped to the C: drive on the file server machine.

- Or, using the other method, use the UNC convention to refer to the shared directory location. For example:

```
<context-param>
  <param-name>WritableDirectory.share.public.loadable
  </param-name>
  <param-value>\\fileserver\Comergent\shared</param-value>
```

</context-param>

Load Balancer

If you run your cluster using a load-balancing solution (either a hardware- or software-based solution), then make sure that the load-balancing is done in a session-sticky fashion. That is, all requests relating to a session should be handled by the same member machine in the cluster.

General Installation Instructions for Clustered Deployment

1. Depending on the cluster architecture, install the Visual Modeler on each instance or into the Administrator server that deploys the Web application to the managed servers.
2. If you are using SQL Server as the Knowledgebase database server, then make sure that you set the ServerId system property and element of the **DataServices.xml** file to a unique two-digit value on each machine that makes up the cluster. This ensures that generated keys are managed correctly. See "Support for SQL Server" on page 108 for more information.
3. If you are using 2-way encryption anywhere in the implementation, then follow these steps:
 - a. Make sure that you start one of the machines before the others.
 - b. Perform a persist operation that requires the use of 2-way encryption.
 - c. Identify the location of the **dcmsKey.ser** file on this machine and copy this file to the corresponding location on the other machines of the cluster.
4. Follow the steps described in "Sharing Directories" on page 115.
5. As a site administrator, set the value of the useSessionCaching system property to "true". This property is in the Profile Manager section of the system properties.
6. Enable your Visual Modeler implementation as a distributed implementation as follows:
 - a. As a site administrator, set the value of the GlobalCache: Implementation Class system property to com.comergent.globalcache.DistributedCache. This property is in the GlobalCache section of the system properties.

This tells the Visual Modeler to use the Ehcache configuration file **WEB-INF\properties\DistributedCache-Config.xml**.

- b. Enable the DistributedEventService by uncommenting the RefreshServiceHelper listener code in the **WEB-INF/web.xml** configuration file:

```
<!-- Start of Listeners -->
<listener>
  <listener-class>
    com.comergent.reference.appservices.cache.CacheManagersHelper
  </listener-class>
</listener>
<!-- comment this out to allow preferences refresh event to propagate
to other nodes -->
<!--
<listener>
  <listener-class>
    com.comergent.reference.appservices.cache.RefreshServiceHelper
  </listener-class>
</listener>
-->
<listener>
  <listener-class>
    com.comergent.dcm.core.SessionMonitor
  </listener-class>
</listener>
<!-- End of Listeners -->
```

- c. As a site administrator, set the value of the cronRefreshTime property. The cronRefreshTime property specifies the polling interval, in seconds, at which a node should check for modified or added cron jobs. Set the value of this property in the Job Scheduler refresh time in seconds field of the Job Scheduler section of the system properties. The default value, -1, prevents the node from periodically checking for changes to cron jobs.
7. By default, distributed nodes are discovered automatically using the Ehcache configuration for both the GlobalCache and EventService. However, you can also modify the cacheManagerPeerProviderFactory property settings for multicastGroupAddress and multicastGroupPort in the **WEB-INF\properties\DistributedGlobalCache-Config.xml** and **WEB-INF\classes\DistributedEventService-config.xml** files to specify the unique IP addresses and ports for a cluster to adjust the scoping of the discovery mechanism.

```
<cacheManagerPeerProviderFactory
  class="net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory"
  properties="peerDiscovery=automatic,
             multicastGroupAddress=230.0.0.1,
```

```
multicastGroupPort=4567, timeToLive=1" />
```

You can also modify the `timeToLive` property setting to restrict how far packets should go. The setting values are:

- 0 - the same host
- 1 - the same subnet
- 32 - the same site
- 64 - the same region
- 128 - the same continent
- 255 - unrestricted

The default `timeToLive` value is 1, the same subnet.

The `GlobalCache` and `EventService` configuration must be the same on each cluster node, and must be unique for each cluster. For example, if you have two separate clusters, each cluster's configuration must be consistent across that cluster's nodes. The clusters themselves must each have unique configurations so that they do not conflict.

8. Copy the **prefs.xml** configuration file to a shared location which is visible to all member machines of the cluster. The location of the file must be specified in the startup script for each cluster member as follows:

```
-Dcomergent.preferences.store=<Path to prefs.xml>
```

9. Configure the cluster to check for new and updated files as soon as possible. This ensures that all servers are in sync and will serve the same information to customers accessing your site. This is especially important in ensuring that the latest generated product index file is available at all times.

Place your configuration property XML files in a shared location accessible by all member machines of the cluster. Then, activate the `AutoReload` element of the **SearchConfigurationProperties.xml** configuration file as follows:

```
<AutoReload activated="true" reloadFilePeriod="30"/>
```

This activates the `AutoReload` function and instructs the cluster to check for updates every 30 seconds.

10. Follow any remaining steps required by your servlet container or load balancer to implement their specific solution. See "Setting Up a WebLogic Cluster" on page 203.

Contact your Sterling Commerce representative for information about setting up other clustering architectures.

Setting Up a WebLogic Cluster

You can use the clustering capabilities to set up a cluster of WebLogic Release 10.3 servlet containers to run your implementation of the Visual Modeler. In general, you should follow the instructions provided by BEA Systems to set up the cluster. This section provides some additional information used to install the Visual Modeler in the cluster.

Web Server

We suggest that you set up the cluster by placing a Web server or separate WebLogic Server as a front-end to the servlet container cluster. You should choose one of these options:

1. Set up a Web server with the appropriate WebLogic Web server plug-in. Supported Web servers include Apache and Microsoft Internet Information Server.

Note: If you use Apache, ensure that your Apache release matches the mod_wl_20.so version. At this time of writing (October 2003), Apache 2.0.42 works with the current mod_wl_20.so provided by WebLogic.

2. Set up a WebLogic Server with the HttpClusterServlet Web application. The HttpClusterServlet maintains the list of all servers in the cluster, as well as the load balancing logic to use when accessing the cluster.

When the user's browser makes a request, the Web server or HttpClusterServlet proxies the request to the WebLogic Server cluster. See the WebLogic documentation for further details.

Administration and Managed Servers

Typically, a WebLogic cluster comprises an Administration Server and one or more Managed Servers. The Web applications are deployed into the Administration Server and then as Managed Servers start or join the cluster, the Administration Server deploys the Web applications to each Managed Server. Consequently, you must deploy the Visual Modeler Web application **Sterling.war** file into the Administration Server first.

Note that when a Managed Server restarts, the Administration Server redeploys the Web applications to the Managed Server: this can take a considerable time, and so you should restart servers at times that ensure that they can be offline for the time they need to restart.

Preparation to Deploy the Visual Modeler Web Application

Because the same WAR file is used to deploy to all cluster members, you must make sure that this WAR file is correctly configured before you deploy the WAR file to the Administration Server. In particular:

1. Make sure that you have used the SDK to build the deployment WAR file.
2. While using the SDK, make sure that the following configuration properties are correctly set:
 - a. **web.xml**: make sure that the WritableDirectory parameters are correctly set to point to the shared directory location. See "Common Directories" on page 206 for more information. Make sure that you have declared the SharedPublicServlet class as described in "SharedPublicServlet Class" on page 207.
 - b. **weblogic.xml**: make sure that you have added a **weblogic.xml** file to the *sdk_home/projects/project/WEB-INF/* directory. See the example file in "WebLogic Releases" on page 87. To support session-sharing across the cluster members, consider adding the element described in "Session Sharing" on page 208.
 - c. Make sure that you have correctly specified the database connection information in the appropriate properties file so that they are correctly set in the **prefs.xml** configuration file.
3. Build the **Sterling.war** file using the SDK distWar target.
4. Copy the **prefs.xml** configuration file to a shared location which is visible to all member machines of the cluster. The location of the file must be specified in the startup script for each cluster member as follows:

```
-Dcomergent.preferences.store=<Path to prefs.xml>
```

Deploying the Visual Modeler Web Application

Follow these steps to deploy the Visual Modeler Web application into the cluster. These instructions assume that you have set up the cluster using the WebLogic administration console on the Administration Server: we refer to the name of the cluster as *cluster_name*. We also assume that the managed servers are up and

running. Make sure that you have used the SDK to create the **Sterling.war** file and that you have moved a copy of the file to a location on the Administration Server.

1. Log into the administration console of the WebLogic Administration Server.
2. Click **Servers** and verify that the managed servers are listed.
3. Click **Clusters** and verify that the name of the target cluster is *cluster_name*.
4. Click **Lock & Edit**.
5. Click **Deployments**.
6. Click **Install**.
7. In the next window, navigate to the location of the **Sterling.war** file and select the radio button next to the **Sterling.war** file name.
8. Click **Next**.
9. Select the **Install this deployment as an application** radio button.
10. Click **Next**.
11. Check the check box next to the cluster named *cluster_name*. By default, the **All servers in the cluster** radio button is selected. You should usually leave this setting unchanged.
12. Click **Next**.
13. In the **Name** field of the General panel on the Optional Settings page, enter the name of your Sterling deployment, for example, Sterling. Accept the defaults for the other values on the Optional Settings page.
14. Click **Next** to review your choices, then click **Finish** to complete the deployment.
15. Click **Activate Changes** to activate the deployment.

Deployment can take ten to twenty minutes. At the end of the deployment process, a page displays a Success message.

SQL Server

Because more than one deployment of the Visual Modeler is accessing the same Knowledgebase on SQL Server, you must set a two-digit server ID for each deployment. You must modify the servlet container command or script that starts the servlet container on each machine so that a Java system property is set: `Comergent.DataServices.General.ServerId`. This should be set on each machine so that each has a unique value: 01, 02, and so on.

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

```
set JAVA_OPTS=-DComergent.DataServices.General.ServerId=12
```

Cron Jobs

The Visual Modeler distinguishes between system cron jobs and application cron jobs. Typically, system cron jobs are run without an associated user and run on every system in a clustered environment whereas application cron jobs must be run associated to a user and usually should be run only by one machine in a cluster.

To set this up, you must do the following:

1. Make sure that in the deployment WAR file, the value of the cronApps system configuration property is set to “system”.
2. For the one application server that should run application cron jobs, make sure that a system property is set as follows:

```
-DComergent.Cron.cronApps=both
```

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

```
set JAVA_OPTS=-DComergent.Cron.cronApps=both
```

Note that how you do this will vary from one servlet container to another. Note that the valid values for this property are: “application”, “both”, “none”, and “system”.

3. Set the value of the Cron Job URL system property to the value of the URL used to access the cluster: for example:

```
http://loadbalancer/Sterling/msg/matrix
```

Common Directories

All the Managed Servers in the cluster must be able to access the same directory locations in the file system: this is where configuration files, shared data files, and other related files such as pagination data is stored for the cluster. You must ensure that all members of the cluster access this location using the same directory paths.

The location of the shared directories is specified in the Visual Modeler **web.xml** file using context parameter elements of this form:

```
<context-param>
  <param-name>WritableDirectory.share.public.loadable</param-name>
  <param-value>/tmp</param-value>
</context-param>
<context-param>
```

```

        <param-name>WritableDirectory.share.public.noloadable
        </param-name>
        <param-value>/tmp</param-value>
    </context-param>
</context-param>
        <param-name>WritableDirectory.share.private.loadable</param-name>
        <param-value>/tmp</param-value>
    </context-param>
</context-param>
        <param-name>WritableDirectory.share.private.noloadable
        </param-name>
        <param-value>/tmp</param-value>
    </context-param>
</context-param>

```

See "Shared Files" on page 199 for the form that the values of these parameters can take. Note that by default, these elements are commented out: in this case, each instance of the Visual Modeler Web application acts independently of the other instances in the cluster. All file accesses are performed locally on the machine running the Web application.

The following table summarizes which files should go where:

TABLE 16. Shared File Locations

Location	Purpose
share.public.loadable	Do not use.
share.public.noloadable	Image files and other files that should be accessible to Web servers to serve up static content. Examples include GIF files associated with promotions and storefront partners.
share.private.loadable	Class files to be shared across the cluster: this directory is used primarily for Sterling Configurator and Visual Modeler.
share.private.noloadable	Configuration files, pagination files, and other files that must be shared across the cluster, but which should not be accessible from users' browsers.

SharedPublicServlet Class

You must uncomment in the element that declares the SharedPublicServlet class: this class is used to serve up static content such as partner logos and promotion images that are uploaded to the Visual Modeler.

```

<servlet>
    <servlet-name>SharedPublicServlet</servlet-name>
    <servlet-class>

```

```
        com.comergent.dcm.core.SharedPublicServlet
    </servlet-class>
</servlet>
```

You must also uncomment in the following elements that map URLs to the SharedPublicServlet:

```
<servlet-mapping>
    <servlet-name>SharedPublicServlet</servlet-name>
    <url-pattern>/htdocs/partnerlogos/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
    <servlet-name>SharedPublicServlet</servlet-name>
    <url-pattern>/htdocs/promotions/images/*</url-pattern>
</servlet-mapping>
```

For each supported locale, uncomment in the corresponding element:

```
<servlet-mapping>
    <servlet-name>SharedPublicServlet</servlet-name>
    <url-pattern>/la/CO/htdocs/*</url-pattern>
</servlet-mapping>
```

For example, uncomment in the following element for the en_US locale:

```
<servlet-mapping>
    <servlet-name>SharedPublicServlet</servlet-name>
    <url-pattern>/en/US/htdocs/*</url-pattern>
</servlet-mapping>
```

Session Sharing

You must also provide information about how sessions are to be shared across the cluster using the **weblogic.xml** deployment file. You may have already created this file to pass in information about the WebLogic environment or you may have to create it only for this purpose. It should be located in your **Sterling.war** Web application file at the same level as the **web.xml** file.

You must add the following fragment to the **weblogic.xml** file:

```
<session-descriptor>
    <session-param>
        <param-name>PersistentStoreType</param-name>
        <param-value>file</param-value>
    </session-param>
    <session-param>
        <param-name>PersistentStoreDir</param-name>
        <param-value>
            <Directory location common to all members of cluster>
        </param-value>
    </session-param>
</session-descriptor>
```



```
</session-param>  
</session-descriptor>
```

Note that a more common setting is:

```
<session-param>  
  <param-name>PersistentStoreType</param-name>  
  <param-value>memory</param-value>  
</session-param>
```

This setting does not support session-failover.

Reloading Files

If shared configuration files can be updated, then each managed server may need to reload the shared copy to pick up changes made by other servers in the cluster. For example, the **SearchConfigurationProperties.xml** file has a setting:

```
<SearchSystemConfigurations>  
  <AutoReload activated="true" reloadFilePeriod="30"/>  
</SearchSystemConfigurations>
```

Set the `activated` attribute to “true” and set the `reloadFilePeriod` attribute to an interval (in seconds) to specify that if an interval of more than 30 seconds elapses between accesses, then the file should be reloaded.

Running a Clustered WebLogic Installation

In a clustered deployment of WebLogic, you must also perform these steps to ensure that the DTDs used by the Visual Modeler are correctly located. On each machine in the cluster:

1. Create or identify a designated directory that may be used to store the DTDs. For example, you can create a sub-directory called ***container_home/local/working/*** in each WebLogic installation.
2. Unjar the Visual Modeler WAR file, and copy the DTD files from their locations under **WEB-INF/** to the designated directory.
3. Modify **startManagedWebLogic.cmd** or **startManagedWebLogic.sh** to set a new runtime flag: `-DComergent.workingDir`. You can use the `$WL_HOME` variable if the designated directory lies under the ***container_home/*** location. For example:

```
-DComergent.workingDir=$WL_HOME/local/working
```

Setting up a Database for Caching

Introduction

This implementation of the distributed Global Cache uses the Knowledgebase database server to store session information. Note that in Release 9.0, only implementations that use the Oracle database server are supported.

1. Log in to the Visual Modeler system administration site as a site administrator.

Your system administration site URL is similar to:

```
http://server:port/Sterling/en/US/enterpriseMgr/admin
```

2. Click **System Services**.
3. Click **Commerce Manager**.
4. In the GlobalCache: Class Name property field, enter:
`com.comergent.dcm.cache.impl.db.DBCache`
5. Click **Save All and return to List**.

Setting up JavaSpaces for Caching

Introduction

This implementation of the distributed Global Cache uses the JavaSpaces technology from Sun Microsystems. This requires a dedicated machine to run the Jini Lookup server, the Javaspaces server, and optionally the transaction server. The steps needed to install and run the JavaSpaces server are described below.

Install the Required Servers

All the Jini servers have an implementation that can be activated using the **rmid** (RMI activation daemon). This means that the servers need to be registered with the **rmid** once, after that they will be automatically restarted if they crash or the machine has been rebooted. The system administrator needs to make sure that the **rmid** daemon is running at all times. In the following section we describe the steps needed to install the Jini Lookup server and JavaSpaces server, and to register the servers with the **rmid** daemon for the first time.

To Implement JavaSpaces

1. Download the Jini Starter Kit v1.2.1_001 from <http://www.sun.com/software/communitysource/jini/download.html>
2. Install the Jini Starter Kit by unzipping the file. This creates a directory called **jini1_2_1_001/** (this is version-dependent, so you may see slightly different numbers). In the following instructions we refer to this directory as **jini_home**.
3. Create a logs directory where you want the servers to store their logs: we will refer to it **logs_home**. This directory can be anywhere on the machine.

4. Start the **rmid** with the following arguments:

```
rmid -d log jini_home/logs -J-Djava.security.policy=none
```

5. You need to make the jar files with ‘-dl’ postfix accessible via a Web server, you can do this by one of the following steps:

- a. Either, run the supplied Web server using the following command (all on one line):

```
java -jar jini_home/lib/tools.jar -port <port>  
-dir jini_home/jini1_2_1/lib
```

You should select a value for <port> that is private to your network.

- b. Or, copy all the **jini_home/lib/*-dl.jar** files to one of your current Web servers.

6. At the command line, navigate to the **jini_home** directory.

7. Run the Jini Lookup server by entering:

```
java -jar ./lib/reggie.jar http://<host>:<port>/reggie-dl.jar ./  
policy/policy.reggie logs_home/reggie_log public
```

Replace <host> by the name of the machine and <port> by the value specified in Step 5a or the port at which the Web server you selected in Step 5b is listening.

8. (Optional) Run the Transaction server by entering:

```
java -jar  
-Djava.security.policy=./policy/policy.all  
-Dcom.sun.jini.mahalo.managerName=TransactionManager ./lib/mahalo.jar http://%HOST%:%PORT%/mahalo-dl.jar ./policy/policy.all  
logs_home/txn_log public
```

9. Run the JavaSpaces server by entering:

```
java -jar -Djava.security.policy=./policy/policy.all
```

```
-Dcom.sun.jini.outtrigger.spaceName=JavaSpace ./lib/outtrigger.jar
http://<host>:<port>/outtrigger-dl.jar ./policy/policy.all
logs_home/frontendspace_log public
```

If you delete the **logs_home**/directory, or if the directory is not available, then the **rmid** will not be able to restart the servers.

To Set Up the Visual Modeler

1. Change the `globalCacheImplClass` property in **Comergent.xml** from:

```
com.comergent.dcm.cache.impl.AppContextCache
```

to:

```
com.comergent.dcm.cache.impl.space.SpacesCache
```

2. Add the `globalCacheParameters` property to **Comergent.xml** in the General group:

```
<globalCacheParameters controlType="text"
runtimeDisplayed="true" ChangeOnlyAtBootTime="true"
visible="true" boxsize="60" displayQuestion="GlobalCache: Param-
eters to be passed to the Global Cache implementation" default-
Choice="" help="Enter a comma separated key value pairs to be
passed to the global cache implementation">
javaspaceName=JavaSpace,transactionServerName=TransactionManager
</globalCacheParameters>
```

The `javaspaceName` and `transactionServerName` are the properties used by the Visual Modeler to find both the JavaSpaces server and the Transaction server respectively. Those are the same names used when the servers are started above. If the `transactionServerName` is set to an empty String, or is not defined, then the Visual Modeler will not use the Transaction server when accessing the JavaSpaces server.

3. In **Comergent.xml**, set the `cronRefreshTime` property to the interval, in seconds, at which a node will poll the Knowledgebase to check for modified or added cron jobs. Set `cronRefreshTime` to a negative number to prevent the node from periodically polling the Knowledgebase. The default value is -1.
4. Modify the security policy settings for the servlet container as follows:
 - a. Copy the **policy.all** file to the directory in which the servlet container binaries are stored.
 - b. Modify the command that starts the servlet container JVM by adding:

```
-Djava.security.policy=./policy.all
```

- c. For WebLogic servers, add the following XML fragment to the **weblogic.xml** file:

```
<security-permission-spec>
  grant { permission java.security.AllPermission "", ""};
</security-permission-spec>
```

5. Optional: to test that the JavaSpaces server connection is working properly you can set the log level to VERBOSE, and the log flag to GlobalCache in the **Comergent.xml** properties file to get more information.

While the JavaSpaces services can be run on an application server with the Visual Modeler, it is better to run it on an independent, but highly available system. This prevents a single point of failure issue if the joint Visual Modeler/JavaSpaces server fails.

Part 2:

Implementation

The chapters in this section of the guide provide information required for you to implement the Visual Modeler at your enterprise.

Purpose

This guide provides an overview to extending current applications and developing new applications for the Visual Modeler. It presents a description of the system architecture, the main Java classes, and a description of the Visual Modeler Software Development Kit (SDK).

Audience

This guide presupposes 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 following conventions shown in Table 17, "Conventions", on page 216:

TABLE 17. 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>

Integrating the Visual Modeler with Selling and Fulfillment Foundation

Integration Overview

In some instances, complex products may have to be configured before they can be bought by customers. In some other instances, such products may have optional components that customers can configure based on their requirements. Visual Modeler enables you to create models that define the configurable options of a product, and to associate products to these models. The Sterling Configurator is a tool that is used to display the configurable products along with the available options to the end user.

The integration between the Visual Modeler and the Selling and Fulfillment Foundation is necessary to enable them to exchange information. The integration is required to ensure that the correct product information, as maintained in Selling and Fulfillment Foundation, is used for defining the models in the Visual Modeler. The prices applied on the products are based on the price list and the currency associated with the guest user. For more information about associating the price list, refer to the Business Center Pricing Administration Guide.

To integrate the Visual Modeler with the Selling and Fulfillment Foundation, you must perform certain configurations in the Visual Modeler application and the Applications Manager.

Configuring the Visual Modeler Properties

You must configure the values of certain properties in the Visual Modeler in order to enable it to obtain the correct product information from Selling and Fulfillment Foundation.

To configure the properties in the Visual Modeler:

1. Point your browser to the following URL:
`http://<hostname>:<port>/<context_root>/en/US/enterpriseMgr/admin`
Here, `hostname` is the IP address, `port` is the listening port of the machine in which the Visual Modeler is installed, and `context_root` is the context root of the hosted Visual Modeler application.
The Login page is displayed.
2. Log in as an administrator by entering your login ID and password, and clicking Log In.
3. Click the System Services hyperlink. The System properties page is displayed.
4. Click the Fulfillment hyperlink. The Properties for Fulfillment page is displayed.
5. Set the Sterling Order Fulfillment System URL property to `http://<hostname>:<port>/smcfs/interop/InteropHttpServlet`. This URL pertains to the Interop servlet of the Selling and Fulfillment Foundation.
Note: If the URL starts with `https`, ensure that a valid certificate is configured on the Selling and Fulfillment Foundation application server.
6. Set the Sterling Item Configurator URL property to:
`http://<hostname>:<port>/sbc/configurator/configure.action`
Here, `hostname` is the IP address of the machine in which the Selling and Fulfillment Foundation is installed, and `port` is the listening port of the machine in which Selling and Fulfillment Foundation is installed.
7. Set the following properties appropriately:
 - User name for the Sterling Fulfillment system
 - Password for the Sterling Fulfillment system

The values of these properties determine the user name and password that will be used to communicate with the Selling and Fulfillment Foundation server.

Configuring the Sterling Configurator Rules

To enable the Sterling Configurator to obtain the model information of the products from the Visual Modeler, you must specify the location of the models, properties, and rules pertaining to models in the Applications Manager.

To configure the Sterling Configurator rules:

1. In the Sign In page, log in as an administrator by entering your login ID and password, and clicking Sign In. The Application Console home page is displayed.
2. From the menu bar, navigate to Configurations > Launch Applications Manager. The Applications Manager is launched in a new browser window.
3. From the Applications Manager menu bar, navigate to Applications > Application Platform. The Application Rules side panel is displayed.
4. In the Application Rules side panel, select System Administration > Item Configurator.
5. Specify the paths to the location where the models, properties files, and rules are stored.

Notes:

- You must copy the .properties files from the **sic_properties.zip** file to the location specified for the properties files. The **sic_properties.zip** file is located in the <INSTALL-DIR>/repository/external folder.
- All the paths specified in the Applications Manager for the model repository are shared by the Selling and Fulfillment Foundation and the Visual Modeler. If the Selling and Fulfillment Foundation and the Visual Modeler reside on different machines, the paths should be mounted on a drive that is accessible to both. For more information about model repository, refer to the Selling and Fulfillment Foundation: Application Platform Configuration Guide.
- In Business Center, a model can be assigned to the item definition of a bundle item. The model-name is saved in the item definition. If you change the model-name at any point of time after it has been saved to the item definition, the item definition needs to be changed to point to the modified model-name. This situation could arise when a user edits the model definition in Visual Modeler.

Introduction to J2EE Web Applications

This chapter presents an overview of the Java 2 Platform, Enterprise Edition (J2EE) and how it is used to deploy Web applications. If you are already familiar with this architecture, then you can skip this chapter.

Architecture

The Visual Modeler is designed to conform to the Java 2 Platform, Enterprise Edition (J2EE) architecture as defined in *Java 2 Platform Enterprise Edition Specification, v 1.2* published by Sun Microsystems, Inc.

The Visual Modeler is deployed as a Web application that comprises a set of Java classes together with accompanying configuration files, HTML templates, and JSP (JavaServer Pages) pages. It must be installed into a servlet container that conforms to the J2EE standard.

Web Applications

A J2EE Web application is built to conform to a J2EE specification. You add Web components to a J2EE servlet container in a package called a *Web application archive* (WAR) file. A WAR file is a JAR (Java archive) file compressed file.

A WAR file usually contains other resources besides Web components, including:

- Server-side utility classes
- Static web resources (configuration files, HTML pages, image and sound files, and so on)
- Client-side classes (applets and utility classes)

The directory and file structure of a Web application deployed as a WAR file conforms to a precise structure. A WAR file has a specific hierarchical directory structure. The top-level directory of a WAR file is the *document root* of the application. The document root is the directory under which JSP pages, client-side classes and archives, and static Web resources are stored. The document root contains a subdirectory called **WEB-INF/**, which contains the following files and directories:

- **web.xml**: the Web application deployment descriptor. It describes the structure of the Web application.
- Tag library descriptor files.
- **classes/**: a directory that contains server-side classes: servlet, utility classes, and Java Beans components.
- **lib/**: a directory that contains JAR archives of libraries (tag libraries and any utility libraries called by server-side classes).

web.xml File

Every Web application deployed in a servlet container must have a **web.xml** file present in its **WEB-INF/** directory. The structure of every **web.xml** conforms to a DTD published as part of the J2EE specification.

The purpose of the **web.xml** is to specify the general configuration of the Web application as required by the J2EE standard. Specifically:

- initialization parameter values are provided for the Web application
- servlet classes used by the Web application may be declared and given names
- each servlet class is mapped to one or more URL patterns: when the servlet container receives a request whose URL matches a pattern defined in the **web.xml** file, then the corresponding servlet is used to process the request
- initialization parameter values are provided for each servlet if required

- session information (such as time out)
- the location of custom tag libraries used by the JSP pages

JSP Pages

Early Java-based Web applications used only servlets to generate the HTML that was sent back to users' Web browsers. Over time, template mechanisms were introduced that enabled Web developers to generate dynamic content by using templates to generate the HTML. Several such template systems are available, however the J2EE architecture has settled on the use of JSP (JavaServer pages) pages to display content.

When a J2EE application receives a request from a user's browser, it first processes the request to extract parameters from the request and to perform business logic initiated by the request. Once the processing is complete, the Web application must dispatch the request to a JSP page: it does this by using a *request dispatcher*. Typically, the servlet context invokes a request dispatcher by passing the target JSP page to the dispatcher and then the request and response objects are *forwarded* by the request dispatcher.

A JSP page comprises a combination of HTML, JSP tags, and scripting elements such as *scriptlets*.

- **HTML:** a JSP page can include any amount of normal HTML. This content is passed right through to the browser page without change.
- **JSP tags:** tags populate the dynamically-generated HTML with values calculated as the page is being generated. There are standard JSP tags such as `<jsp:getProperty>`, `<jsp:include>`, and `<jsp:forward>`. These are available to anyone creating a JSP page. In addition, you can specify that your Web application uses one or more custom tag libraries. Each custom tag library must be declared in the **web.xml** file for the Web application and the declaration must specify both the URI for the tag library and the location of the tag library descriptor (TLD) file.

<p>Attention: In the Visual Modeler, the use of the tag libraries is now deprecated. For performance reasons, we suggest that you use scriptlets. JSP tags can still be used in some existing applications or specialized integration tasks.</p>

- **Scripting elements:** You can intersperse the HTML and JSP tags in a JSP page with Java code that is contained between the scriptlet opening tag `<%`

(or `<jsp:scriptlet>`) and the closing tag `%>` (or `</jsp:scriptlet>`). Scriptlets are most commonly used to manage complex flow control in a JSP page.

Note that most JSP scripting elements can be invoked using a shorter form as described in the following table.

TABLE 18. Short Forms of Standard JSP Tags

Short form	XML form
<code><%</code>	<code><jsp:scriptlet></code>
<code><%=</code>	<code><jsp:expression></code>
<code><%!</code>	<code><jsp:declaration></code>
<code><%@</code>	<code><jsp:directive></code>

Data is passed to a JSP page using a variety of mechanisms, the most important of which are implicit objects and beans.

- **Implicit objects:** Every JSP page provides the Web developer with objects that can be used to display data on the generated HTML page. The most important of these are the page, request, session, config, and application objects.
- **Beans:** Most of the data generated by the business logic of the application is passed to the JSP page by adding Java beans to one of the implicit objects listed above.

Model 2 Architecture

The Visual Modeler is designed to conform to Sun's "Model 2" architecture. In this architecture, three functional components referred to as the Model, View, and Controller (MVC) partition the functionality of the Web application into logically distinct components.

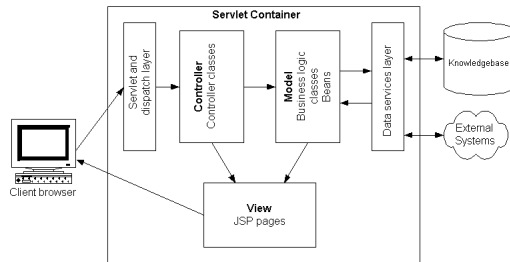


FIGURE 9. Model 2 Architecture

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

Controllers

In the Model 2 architecture, controllers are Java classes intended to manage the processing of an inbound request and then to forward the request to an appropriate JSP page. The basic structure of a Visual Modeler controller follows this form:

```
public class GenericController extends Controller
{
    public void execute() throws Exception
```

```
{
    //Dispatch some business logic
    BizObjs resultBizObjects = calculate();
    //Generate the beans
    Vector beans = generateBeans(resultBizObjs);
    //Attach the beans to the request
    attachBeans(beans);
    // Dispatch to JSP page
    String pageName = choosePageLogic();
    // Dispatch to JSP page
    Dispatcher rd = request.getDispatcher(pageName);
    rd.forward(request, response);
}

protected BizObjs calculate() throws Exception
{
    //do some processing
    return resultBizObjs;
}

protected Vector generateBeans(BizObjs bizObjs)
{
    //create beans from business objects
    return beans;
}

protected void attachBeans(Vector beans)
{
    Iterator it = beans.iterator();
    while (it.hasNext())
    {
        DataBean bean= (DataBean) it.next();
        request.setAttribute (beanName, bean);
    }
}

protected String choosePageLogic()
{
    //logic to determine where to forward the request
    return pageString;
}
}
```

Model

In the Model 2 architecture, the objects that represent data in the system are maintained by the model component. It is common to distinguish the business objects from the beans used in the JSP pages.

Once the business logic finishes creating and transforming the business objects, the controller class transforms the business objects into their corresponding beans. The beans are then passed to the JSP page for presentation.

View

The user interface of the Web application is served to the browser using JSP pages. Data is passed to each JSP page in the form of beans. These are classes with defined accessor methods that enable the logic on the JSP page to retrieve values using tags of the general form:

```
<%  
    DataBean dataBean = request.getAttribute("nameOfBean");  
    String stringProperty =  
        dataBean.getNamedProperty("nameOfProperty");  
%>
```

Note that it is possible to use a combination of scriptlets, simple JSP tags, and more sophisticated custom tags to manage page layout and the display of data.

Further Reading

The published literature on Web applications, J2EE, servlets, and JSP pages is vast. The following are recommended books for further reading:

- Hall, *Core Servlets and JavaServer Pages*, Second Edition, Prentice Hall, 2003
- Hunter, *Java Servlet Programming*, Second Edition, O'Reilly, 2001
- Fields and Kolb, *Web Development with JavaServer Pages*, Second Edition, Manning, 2001

This chapter describes the Visual Modeler architecture and introduces some of the important Java classes that the Visual Modeler and its applications use. It assumes a thorough understanding of the J2EE architecture.

This chapter is intended to help you to modify or extend existing applications or write new applications. Note that not all parts of the Visual Modeler conform to this architectural description.

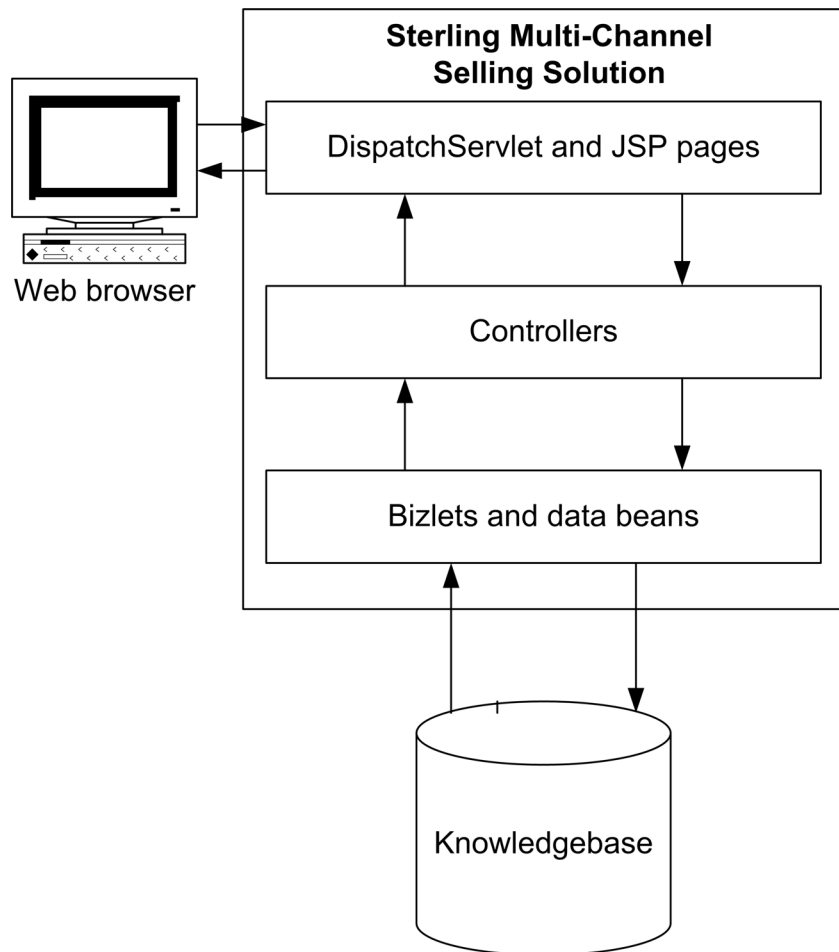


FIGURE 10. Visual Modeler Architecture

Visual Modeler Web Application

When you install the Visual Modeler into your servlet container, it installs as a WAR file, **Sterling.war**. When the WAR file deploys, it unjars into a directory

called **Sterling/**. The **WEB-INF/** sub-directory contains the **web.xml** file for the application.

The most important configuration settings in this file are:

- The definition of the `InitServlet` and `DispatchServlet`:
 - `InitServlet` loads when the servlet container starts. `InitServlet` reads in all of the configuration information for the Visual Modeler using the value of the `propertiesFile` element: by default this is **Comergent.xml**.
 - `DispatchServlet` is the main servlet used to process inbound requests. Most of the URLs defined in the servlet mapping section resolve to the `DispatchServlet`.
- The servlet mapping section maps most URL patterns to the `DispatchServlet`. Note that `"/msg/*"` is used to map requests to the `MessagingServlet`: this ensures that inbound XML messages are processed by this servlet class.
- The session configuration element sets a session timeout value of 30 (minutes). Each implementation of the Visual Modeler must carefully consider an appropriate value for this parameter. Bear in mind the following:
 - End users of the system may leave their browsers unattended while they step away from their desks. If an unscrupulous user can access the browser when a session is still valid, then they can access the system.
 - End users may punch out to other external systems in the course of using the Visual Modeler. The session timeout value must give enough time for users to punch out and return.
 - Each session uses system resources. The greater the session timeout value, then the greater the memory usage of the system.
- The location of the Comergent tag library descriptor (TLD) file is provided.

Processing Requests

When the Visual Modeler receives a request from a user's browser, it must determine how to process the request and how to display the result to the user. It does this using the **MessageTypes.xml** configuration files. These files determine the mapping between a request and the logic processing classes and JSP pages used.

1. When a request is received, the message type is identified and the appropriate controller invoked.
2. Additional business logic may be invoked using a business logic or bizAPI class.
3. The controller then forwards the request to the specified JSP page to render the output back to the user's browser.

The `messageTypeFilename` element of the `GeneralObjectFactory` element of the **Comergent.xml** file specifies the comma-delimited list of **MessageTypes.xml** file used to specify the message types. Each **MessageTypes.xml** file declares a list of message types organized by message group.

Each request specifies the message type as the `cmd` parameter. For example, if the URL is of the form:

```
../Sterling/catalog/matrix?cmd=search
```

then the name of the message type is "search".

Each message type is identified by the `Name` attribute of its `MessageType` element. The `Name` attribute identifies which message type is being requested when a user clicks a URL.

<p>Attention: You must make sure that each message group and message type have a unique name. You must check the collection of MessageTypes.xml files to ensure that you have not defined message groups and message types with the same name. See "Overriding MessageType Definitions" on page 233 for an exception to this rule.</p> <p>We suggest that you list message types alphabetically by name within message groups as a means of quickly identifying the duplication of message type names.</p>
--

`MessageType` elements have one or more of the following child elements:

- `BizletMapping`: used for message processing, it associates a Bizlet class and a method of this class to process the message.
- `ControllerMapping`: associates a controller to be used to process the request. For message processing, you can specify a `BizRouter` class to invoke a Bizlet class to process the message.
- `JSPMapping`: associates a JSP page to be used to display the result of processing the request.

A `MessageType` element may specify any combination of these three elements.

- If no `ControllerMapping` element is specified, then, by default, the `ForwardController` class is used. This class simply forwards the request to the JSP page specified by the `JSPMapping` element. If no `JSPMapping` element is found or if the specified JSP page is missing, then an error page is displayed.
- If a custom controller is specified, it may process the request itself (see "Controller Classes" on page 236), or it can invoke a business logic class using the `runAppJob()` method of the `AppExecutionEnv` class (see "AppExecutionEnv Class" on page 241).
- If no `JSPMapping` element is specified, then the business logic class or controller must specify which JSP page is to be used.

Each request or message is validated against the entitlements system to verify that the user can execute the message type. Not all users can execute all message types.

Overriding Message Type Definitions

The `MessageType` element has an optional attribute: `IsOverlay`. If this attribute is set to "true", then the `MessageType` definition overrides any previous definition of this message type given in any earlier **MessageTypes.xml** file listed in the `messageTypeFilename` element.

If two or more definitions are given for the same message type without one specifying the `isOverlay` attribute, then an initialization error is displayed and the first definition of the message type is used.

Note that the `IsOverlay` attribute does not change the location of the `MessageType`: this is still determined by the message group to which the first definition belongs or by the `MessageTypeRef` element that references the message type.

For example, to override the definition of the `adirectLogin` message type, you can define an element as follows:

```
<MessageType Name="adirectLogin" IsOverlay="true">
  <ControllerMapping>
    com.comergent.apps.common.controller.MyLoginController
  </ControllerMapping>
  <JSPMapping>../common/adirectPageLoader.jsp</JSPMapping>
</MessageType>
```

The `IsOverlay` attribute can also be used for `MessageGroup` declarations so that you can overwrite the definition of a message group, but its use is not recommended.

Default Elements

For each message group, you can specify default BizletMapping, ControllerMapping, and JSPMapping elements. These are used when no mapping is specified for a message type that belongs to the message group.

In general, if no default mapping is specified in a message group, then the system looks for a default mapping in the parent message group of the current message group. If no mapping is found anywhere in the message group tree, then values specified in the MessageGroupDefaults message group are used.

Key Java Classes

At a schematic level, the Visual Modeler applications all have the same structure: they are composed of controllers, business objects and bizlets, and JSP pages.

Wrapper Classes

Several of the standard classes used in J2EE Web applications have been wrapped in wrapper classes to manage any minor idiosyncrasies among the supported servlet containers:

ComergentContext

This class is used to wrap the servlet container context. You can use it to retrieve the Env object for environment information. Note that any context attribute that is set must be serializable. An exception is thrown if you attempt to set a non-serializable attribute.

It provides the *getResourceAsStream()* method: this method can be used to access a file as a stream for read-only access. You must use the *adjustFileName()* method of the LegacyFileUtils class for write access to a file.

ComergentDispatcher

This class is a lightweight wrapper of the standard RequestDispatcher class: it provides *forward()* and *include()* methods.

ComergentRequest

This class wraps the standard HttpRequest class and provides helper methods to parse the inbound requests and messages.

ComergentResponse

This class wraps the standard HttpResponse class. It provides a *localRedirect()* method to pass a request with a new message type. For example, you may want a

controller to process a request, and then to pass the result on to another controller: you do this by calling:

```
response.localRedirect(request, "messageType");
```

This has the effect of submitting the request to the DispatchServlet as if it had been received as an HTTP request.

ComerentSession

This class wraps the standard HttpSession class. When a user first logs in, a User data bean is created and added to the ComerentSession object. You can access user information through the ComerentSession *getUser()* method.

For example:

```
session.getUser().getUserKey()
```

will return the current user's key; and

```
session.getUser().getPartnerKey()
```

returns the key of the partner to whom the user belongs.

The ComerentSession object is used to store information that must be persistent for more than one request of a user's session. Use the *setAttribute(String s, Object o)* method to set an object in the session and *getSession(String s)* to retrieve it. Objects stored in the session must implement the Serializable interface: all generated data beans implement this interface and so these may be stored in the session.

The ComerentSession class also provides a *logout()* method: invoking this method immediately invalidates the servlet container session.

Servlets

The main servlets used are:

- **InitServlet:** this servlet loads when the servlet container starts. Its *init(ServletConfig config)* method initializes the ComerentAppEnv class.
- **DispatchServlet:** this servlet is used to service almost all requests processed by the Visual Modeler. Its principle method call is:

```
void dispatch(HttpServletRequest request, HttpServletResponse response)
```

This method creates a controller to handle the request with:

```
Controller controller createController(ComerentRequest comergen-
```

```
tRequest)
```

and then invokes:

```
controller.init(comergentContext, comergentSession,  
    comergentRequest, comergentResponse);  
controller.execute();
```

Note that the instance of the Controller class created by the *createController()* method is a function of the request. The request message type determines the Controller class because the controller is created by the GeneralObjectFactory class. The GeneralObjectFactory uses the **MessageTypes.xml** file to map from the request message type to a Controller class.

- DebsDispatchServlet: this servlet is used to process XML messages posted from another system to the Visual Modeler. If the content type of the request starts with “application/x-icc-xml” or “text/xml”, then it invokes the MessagingController to process the request.

Controller Classes

The Visual Modeler offers two different ways of using controllers to process requests:

Custom Controllers

You can write your own Controller class by extending the `com.comergent.dcm.caf.controller.Controller` class. When you do this, you must provide the application logic to determine the JSP page to which the request should be forwarded. For example:

```
boolean processingSuccess = false;  
/*  
 *  
 * Business logic processes request and sets processingSuccess to  
 * true if successful.  
 */  
  
if (processingSuccess)  
{  
    callJSP("SuccessMessageType");  
}  
else  
{  
    callJSP("FailureMessageType");  
}  
  
protected void callJSP(String messageType) throws  
    ControllerException, ICCEException, IOException
```

```
{
    String resource = getJSPName(messageType);
    ComergentDispatcher rd =
        request.getComergentDispatcher(resource);
    rd.forward(request, response);
}

protected String getJSPName(String messageType) throws ICCEException
{
    JSPObjectID id = new JSPObjectID(messageType);
    return GeneralObjectFactory.getGeneralObjectFactory().-
        getMapping(id);
}
```

SimpleController

You can extend the SimpleController class to process the request if there is only one exit point from the application logic. The SimpleController uses the message type of the request to determine the JSP page to which the request is forwarded once the application logic is finished. To extend the SimpleController class, overwrite the *calculate()* method.

MessagingController

This class is used to process XML requests (such as price and availability or shopping cart transfer requests from other systems).

DataBean Classes

Access to data in the Visual Modeler is managed through data objects: these are XML documents that describe the business entities such as partners, users, products, and so on. They describe the fields of the data object together with information about how they map to database tables in the Knowledgebase. Each data object XML file is used to generate a corresponding DataBean Java class.

The DataBean classes are the main classes used to represent each business entity in the Visual Modeler. Each business entity such as a user, partner, product, and so on, is represented in memory by an instance of the appropriate DataBean class. See CHAPTER 18, "Introducing Data Beans and Business Objects" for more information. Some legacy application may still use the BusinessObject class, but in general the use of the BusinessObject class is deprecated.

DataBean classes are also used to pass data to JSP pages. Any data object definition in the Visual Modeler XML schema may be used to generate a DataBean class by running the generateBean target (see the CHAPTER 23, "Software Development Kit" for more details).

The `DataBean` class is a general abstract class and all generated data bean classes extend this class. Each `DataBean` class provides `restore()` and `persist()` methods that retrieve and save data in the database respectively.

Some applications make use of application beans: see "Application, Entity, and Presentation Beans" on page 273 for a discussion of how these beans are used.

ObjectManager and OMWrapper Classes

You should not instantiate `DataBean` classes by using their constructors. Instead use the `ObjectManager` and `OMWrapper` classes to create new instances of objects as your applications require them. These classes follow the Factory pattern in that they provide a class designed to generate object instances as they are required. They enable you to switch from one object class to another without changing the application code that creates and uses the objects.

Creating Objects

In general, you should use the `OMWrapper` class rather than the `ObjectManager` class, but both can be used. You use these classes to create objects with the following methods:

```
ObjectClass temp_ObjectClass =  
    (ObjectClass) OMWrapper.getObject("ObjectName");
```

or

```
ObjectManager temp_ObjectManager = ObjectManager.getInstance();  
ObjectClass temp_ObjectClass =  
    (ObjectClass) temp_ObjectManager.getObject("ObjectName");
```

Mapping Object Names to Object Classes

The `ObjectManager` and `OMWrapper` classes use the `ObjectMap.xml` configuration file (located in `debs_home/Sterling/WEB-INF/properties/`) to determine which type of object is created from the object name provided in the `getObject()` method.

Attention: Do not add comments to the <code>ObjectMap.xml</code> file: these can cause errors on initialization.

Each Object element is of the form:

```
<Object ID="ObjectName">  
    <ClassName>ObjectClass</ClassName>  
</Object>
```

When the *getObject("ObjectName")* method is invoked, an instance of the *ObjectClass* class is returned. The *ObjectName* must be the name of a Java class or interface and the *ObjectClass* must be a subclass of the *ObjectName* class (possibly itself) or a class that implements the *ObjectName* interface.

If the **ObjectMap.xml** file does not have an *Object* element whose ID attribute matches the *ObjectName* parameter, then the *ObjectManager* or *OMWrapper* creates an instance of the *ObjectName* class. That is, it behaves as if there is an element of the form:

```
<Object ID="ObjectName">
  <ClassName>ObjectName</ClassName>
</Object>
```

For example, suppose that the **ObjectMap.xml** file contains the element:

```
<Object ID="com.comergent.bean.productMgr.ProductBean">
  <ClassName>
    com.comergent.bean.productMgr.MatrixProductBean
  </ClassName>
</Object>
```

Then the following method invocation will create an instance of the *MatrixProductBean* class:

```
ProductBean temp_ProductBean = (ProductBean)
  OMWrapper.getObject("com.comergent.bean.productMgr.ProductBean");
```

Note that the *MatrixProductBean* must extend the *ProductBean* class: otherwise a *ClassCastException* would be thrown at runtime. However, if there is no element whose ID attribute is *com.comergent.bean.productMgr.ProductBean*, then the same call would return an instance of the *com.comergent.bean.productMgr.ProductBean* class.

Restrictions

Note that you cannot create *Object* definitions so that the class specified in the *ClassName* element in one *Object* element is the ID attribute in another *Object* element. The only exception to this rule is when the class is used both as the ID and *ClassName* values for a single *Object* element. In particular, if you extend a data object (see "Extending Data Objects" on page 267), then:

1. Define an *Object* element that maps the extended class to the extending class:

```
<Object ID="<Extended class>">
  <ClassName><Extending class></ClassName>
</Object>
```

2. Make sure that you replace any reference to the extended data object in any `ClassName` elements to the extending data object.

Passing Parameters

If you need to pass parameters to the object constructors, then the following `OMWrapper` method is also available:

```
ObjectClass temp_ObjectClass = (ObjectClass)
    OMWrapper.getObjectArg("ObjectName", Object arg1, ... ,
        Object arg10);
```

In this form, you can pass up to ten parameters as Objects into the method invocation. The following `OMWrapper` and `ObjectManager` method calls enable you to pass in an unlimited number of parameters as an array of objects:

```
ObjectClass temp_ObjectClass = (ObjectClass)
    OMWrapper.getObject("ObjectName", Object[] args);
```

or

```
ObjectClass temp_ObjectClass = (ObjectClass)
    temp_ObjectManager.getObject("ObjectName", Object[] args);
```

For example, suppose that the **ObjectMap.xml** file contains the element:

```
<Object ID="com.comergent.bean.productMgr.OrderBean">
    <ClassName>com.comergent.bean.matrix.MatrixOrderBean</ClassName>
</Object>
```

Here, the `MatrixOrderBean` class is a subclass of the `OrderBean` class. Suppose that the `MatrixOrderBean` has a constructor of the form `MatrixOrderBean(CartBean cb)`.

Then the following method invocation will create an instance of the `OrderBean` class using an instance of the `CartBean` class as a parameter:

```
Cart temp_CartBean = (CartBean)
    OMWrapper.getObject("com.comergent.bean.partnerMkt.CartBean");
/*
    Code that processes the cart bean object
*/
OrderBean temp_OrderBean = (OrderBean)
    OMWrapper.getObjectArg("com.comergent.bean.productMgr.OrderBean",
        temp_CartBean);
```

Object Pooling

If you expect some classes of object to be created and used frequently, then you can use the `ObjectManager` and `OMWrapper` classes to create a pool of objects. The parent object (identified by the ID attribute) must implement the `poolable` interface.

This interface is a part of the `com.comergent.dcm.objmgr` package. It declares one method `reset()` that you must implement.

When you are finished with a poolable object, you can return it to the object pool by using the `return()` method as follows:

1. In the **ObjectMap.xml** entry for a pooled class, set the `MaxPoolSize` attribute to the number of objects you want created in the pool:

```
<Object ID="ObjectName" MaxPoolSize="n">
  <ClassName>ObjectClass</ClassName>
</Object>
```

2. Create instances of the object class using `OMWrapper` and `ObjectManager` as described above.
3. When you are finished with the object, then return the instance to the pool using:

```
OMWrapper.return(temp_ObjectClass);
```

4. or

```
temp_ObjectManager.return(temp_ObjectClass);
```

Note that if you create an object by passing in parameters as described in "Passing Parameters" on page 240, then a new object is created rather than re-using an object from the pool.

AppExecutionEnv Class

The `AppExecutionEnv` class can be used to run business logic classes. However, the use of business logic classes is deprecated, so use this class only to support legacy applications. You use the static methods `runAppObj()` to invoke the creation of a business logic class and to execute its prolog and service methods.

In its most common form, you can use:

```
AppExecutionEnv.runAppObj(String messageType, BizObjTable bizObjects)
```

The `AppExecutionEnv` class invokes the business logic class determined by the `messageType` string and which takes the `BizObjTable` vector of business objects as the input business objects.

AppsLookupHelper Class

There are many situations in the Visual Modeler where the status of a data object is managed using a lookup code. For example, the order status of an order can change several times through the placing of an order. There are also several examples of display fields such as the Title of a user which can take several well-defined values

and which need to be managed for different locales. This data is stored in the CMGT_LOOKUPS table of the Knowledgebase database schema.

For each lookup type, there can be one or more lookup codes and each code has an associated description string. For example:

TABLE 19. Lookup Example

Lookup Type	Lookup Code	Description
AddressType	10	Billing
AddressType	20	Shipping

You can use the AppsLookupHelper class to map a lookup code to a description string. By invoking the appropriate method of the AppsLookupHelper class, pass in the lookup code as a parameter and the corresponding String is returned. Depending on which lookup type you are interested in, you choose the appropriate method for that lookup type. The method used determines which lookup type is used to retrieve the lookup code from the CMGT_LOOKUPS table. For example, to retrieve an order status code string, you can write:

```
String orderStatusString =
    AppsLookupHelper.getOrderStatusForCode(orderStatusCode);
```

Conversely, you can retrieve the lookup code using:

```
int orderStatusCode =
    AppsLookupHelper.getCodeForOrderStatus(orderStatusString);
```

Most, though not all, lookup types have helper methods defined. Check the Java doc for the AppsLookupHelper class for details. For further information, see "Support for Lookup Codes" on page 248.

ComergentAppEnv Class

Use the ComergentAppEnv class to provide your code with environment information specific to the application. It provides the following useful methods:

- *adjustFileName()*: this method has been moved to the LegacyFileUtils class. See "LegacyFileUtils Class" on page 244.
- *constructExternalURL()*: use this method to construct a URL that enables a client to be re-directed back to the server. Primarily, you use this method to generate a redirect URL to enable the server to restore session information.
- *getEnv()*: this method returns the environment object.

- `getContext()`: this method returns the application context.

Global Class

The use of this class is deprecated. Its logging function has been replaced by the log4j API: see CHAPTER 20, "Logging" for more information. Its support for retrieving the values of properties has been replaced by the Preferences mechanism. If you need to continue to use code that uses the Global class, then replace each usage by the LegacyPreferences class.

GlobalCache Interface

Use this interface to define a cache that provides access to cached objects used by all Visual Modeler applications. It can be used to support a clustered environment in which the Visual Modeler is running on more than one machine.

To use a cache class that implements the GlobalCache interface, you must implement the methods of the interface. The cache class is loaded when the `InitServlet` `init()` method is invoked. You must provide the name of the class as the `General.globalCacheImplClass` element of the **Comergent.xml** file. A default implementation is provided with Visual Modeler:

`com.comergent.dcm.cache.impl.AppContextCache`.

You access the implementation of the GlobalCache interface by:

```
GlobalCache globalCache = GlobalCacheManager.getGlobalCache();
```

The interface supports the following methods:

- `public String store(Serializable entry)`: stores an object in the global cache, which remains until the application cleans it up.
- `public boolean store(String id, Serializable entry)`: stores an object in the global cache, which remains until the application cleans it up.
- `public String cache(Serializable entry)`: stores an object in the global cache. The object is available as long as the application is using it, but the cache system cleans it up automatically.
- `public String cache(Serializable entry, long lease)`
- `public boolean cache(String id, Serializable entry)`
- `public boolean cache(String id, Serializable entry, long lease)`
- `public boolean contains(String id)`: checks if the cache contains the specific object.
- `public Object get(String id)`: retrieves the cacheable object.

- *public Object remove(String id)*: removes a cacheable object.
- *public boolean gc()*: This method should be called by a Cron job so the cache can clean up unused entries.

LegacyFileUtils Class

The LegacyFileUtils class provides helper methods for working with files. Its use is deprecated, but it provides support for methods previously provided by the ComergentAppEnv class:

- *adjustFileName()*: It returns the real path name of a file. Use this method to access files for either reading or writing: do not use the *getRealPath()* method because this can return null. In a clustered environment, the *adjustFileName()* method ensures that all members of the cluster access the same file. You must use this method with four parameters:

```
adjustFileName(String fileName, boolean share, boolean xPublic,  
              boolean xLoadable);
```

Use of the one-parameter form of this method is deprecated. The boolean parameters are used to determine the location of the file using the configuration parameters specified in the WritableDirectory element of the **web.xml** file.

OutOfBandHelper Class

The OutOfBandHelper class provides a means to generate an output stream using a JSP page as a template. An example of its use is given here:

```
ComergentRequest request = ComergentAppEnv.getRequest();  
ComergentResponse response = ComergentAppEnv.getResponse();  
ByteArrayOutputStream stream = new ByteArrayOutputStream();  
OutOfBandHelper outOfBandHelper = new OutOfBandHelper(request,  
              response, stream);  
outOfBandHelper.getRequest().setAttribute(  
    ComergentRequest.COMERGENT_SESSION_ATTR,  
    request.getComergentSession());  
outOfBandHelper.callJSP(messageType);  
/*  
 * Initialize SendSMTP and use the stream to to set the body of the  
 * message  
 */  
String mimeType = "text/html";  
String smtpHost = Global.getString(  
    "C3_Commerce_Manager.SMTP.SMTPHost");  
SendSMTP smtp = new SendSMTP(smtpHost);  
StringBuffer sb = new StringBuffer(subject);  
String message = null;
```

```
String enc = ComergerentI18N.getComergerentEncoding();
message = stream.toString(enc);
//Send the mail
smtp.send( from, to, cc, subject, message, mimeType);
```

In this example, you can see how the `OutOfBandHelper` class is initialized using the existing request and response objects and an output stream. Its `callJSP()` method, generates the output stream by passing the request and response objects to the JSP page determined by the message type parameter, and the output stream can be used by the application to retrieve the content.

The `OutOfBandHelper` class makes use of session and context information when mapping a message type to a JSP page. Consequently, you can use different JSP pages for different locales in the same way as you do for processing browser requests and the `OutOfBandHelper` class will resolve which locale's JSP page to use and apply the same failover logic.

Preferences Class

The Preferences module provides the mechanism for accessing Visual Modeler properties. It is one of the modules provided in the platform modules: see "Preferences Service" on page 261 for more information. The basic usage of the Preferences API is as follows:

```
private static Preferences temp_Preferences =
    Preferences.getPreferences();
String temp_MyPropertyString =
    temp_Preferences.getString("MyProperty");
```

The main methods it supports to retrieve properties are:

- `public String getString(String key, String def)`
- `public boolean getBoolean(String key, boolean def)`
- `public double getDouble(String key, double def)`
- `public float getFloat(String key, float def)`
- `public int getInt(String key, int def)`
- `public long getLong(String key, long def)`

There are corresponding `putType()` methods for each `getType()` method: for example:

- `public void putString(String key, String value)`

If you invoke the `getPreferences()` method without a parameter, then you retrieve the singleton Preferences object that the Visual Modeler supports. If you pass in the

name of a class (for example `getPreferences(MyClass.class)`), then the object you retrieve is scoped: that is, the name of the properties whose values you retrieve using the Preferences object have the package path of the class prepended to the property name you provide.

For example, suppose that `MyClass` is in the `com.comergent.myApplication` package. Then the following fragments of code are equivalent:

```
private static Preferences temp_Preferences =
    Preferences.getPreferences();
String temp_MyPropertyString =
temp_Preferences.getString("com.comergent.myApplication.MyProperty");
```

and:

```
private static Preferences temp_Preferences =
    Preferences.getPreferences(com.comergent.myApplica-
tion.MyClass.class);

String temp_MyPropertyString =
temp_Preferences.getString("MyProperty");
```

PriceCheckAPI Class

The `PriceCheckAPI` class provides the main means for applications to retrieve pricing information for products. It provides a number of static methods: these take as arguments a `Vector` of pricing line items and partner keys for the current user and the partner serving up the prices: either the enterprise or one of the `Partner.com` partners.

The main method is `Check()`: this method has several forms, but in general they all specialize the following method:

```
public static Vector Check (Vector lineItems, Timestamp date,
    Long partnerKey, Long storefront, Long verticalKey,
    Long currencyKey)
```

All products must be passed in the `Vector` of pricing line items: these are objects of the `PricingLineItem` class. You can specify a quantity in each pricing line item. The date parameter enables you to retrieve prices as they would appear on a specified date: if the parameter is null, then the current date is used. The `partnerKey` parameter is the partner key of the user whereas the `storeFront` parameter is the partner key of the current storefront: that is, think of the `partnerKey` as representing the buyer and the `storefrontKey` as the seller. The `verticalKey` parameter is the key of the current customer type and `currencyKey` is the key of the current currency.

It also provides methods to retrieve a list of price lists:

- `getAssignedPriceListKey()` returns a List of all price list keys assigned to the partner of the current user regardless of the current selection of currency and customer type.
- `getInContextPricePriceListKey()` returns a List of all price list keys assigned to the partner of the current user based on the session settings for currency and customer type.

Transactions

The Visual Modeler for provides support for transactions: database actions that span one or more atomic operations. In general, you use the Transaction class to manage situations in which several data objects must be persisted together, and if one fails, then they should all fail.

Message Conversion Classes

Converter Classes

The Visual Modeler must be able to transform XML documents from one form to another. The system uses converters for this purpose: these are classes that implement the Converter interface.

<p>Note: The converter makes use of stylesheets: these can be compiled into Java classes. A system property setting, <code>compileStyleSheets</code>, controls whether the stylesheets are compiled or not.</p>
--

Message Categories

In order to convert from one document format to another, you must specify the source and target formats precisely. Each message must belong to a message family and a message version: together these define a *message category*. There can only be one form of a given message type within a message category.

For example, the message family dXML and the message version 5.0 uniquely determine a message category. Within this message category, there is only one form of the message type ShoppingCartTransfer.

Converter Interface

The Converter interface is defined by:

```
public interface Converter
{
    public void setConfig(MessageConversion mc);
}
```

```
public MessageConversion getConfig();
public Object getProperty(String name);
public void setNext(Converter next);
public Converter getNext();
public String getIncomingMessageType();
public String getConvertedMessageType();
public void setSource(Document doc);
public void setSource(InputStream is);
public void setSource(Reader reader);
public void setSource(DefaultHandler handler);
public void setTarget(Document doc);
public void setTarget(OutputStream os);
public void setTarget(Writer writer);
public void setTarget(DefaultHandler handler);
public void setParameter(String paramName, String paramValue);
public void convert() throws ConverterException;
}
```

To create a converter class, you must implement these methods. In your code, you create a converter using the `ConverterFactory`:

```
ConverterFactory cf = ConverterFactory.getConverterFactory();
Converter converter = cf.getConverter(String sourceMsgType,
    String sourceMsgCategory, String targetMsgCategory);
```

The static `getConverter()` method of the `ConverterFactory` class uses several parameters to identify which `Converter` class should be instantiated. It reads from the **MessageMap.xml** using the source and target message categories together with the message types to determine which `Converter` class must be used. Once created, the converter converts from a source document to a target document:

```
converter.setSource(srcDoc);
converter.setTarget(targetDoc);
converter.convert();
```

Note that the input and output to the conversion process can either be documents or streams.

Support for Lookup Codes

The Visual Modeler uses lookup codes to provide a mechanism for maintaining and displaying locale-specific strings to users. For each lookup type, you can define one or more lookup codes, and for each lookup code, you can define a string for each supported locale.

What lookup support does the Visual Modeler provide?

The Visual Modeler has the capability of automatically providing lookups between code values and their corresponding strings and from lookup code strings to code values.

If the “code” DsElement is set, then the “string” is automatically populated from the lookup cache. If the “string” value is set, then the “code” is looked up using the string value.

Are string values localized?

Yes. For a code-to-string lookup, the mechanism uses the user’s locale to determine which string value to use. For a string-to-code lookup, the mechanism uses the user’s locale when searching on a string value to find a corresponding code.

How do I define a code to string mapping?

Code-to-string relationships are defined in the **DsDataElement.xml** schema file. If both of the “code” and “string” DsDataElements are then used in a data object, then the code-to-string mapping is handled automatically.

The following is an example of a DataElement code-string pair.

```
<DataElement Name="OrderStatus" Description="Order Status"
  DataType="LONG" MaxLength="20" LookupType="OrderStatus"
  LookupString="OrderStatusString"/>
<DataElement Name="OrderStatusString" Description="Order Status"
  DataType="STRING" MaxLength="260" LookupType="OrderStatus"
  LookupCode="OrderStatus"/>
```

Are lookups performed for XML messages?

Yes. If a dataobject used for messaging contains a code-string pair, then the string value will automatically be used to look up the code.

How is the lookup cache loaded?

The lookup cache is loaded at system startup.

The Visual Modeler modular architecture is designed to make implementations easy to customize and upgrade. This chapter provides an overview of modular architecture, platform modules, and the module interfaces, and describes each module. It covers the following topics:

- "Overview" on page 252
- "Platform Modules" on page 253
- "Access Policy" on page 254
- "Authentication" on page 254
- "Base64" on page 254
- "Classpath Appender" on page 254
- "Cryptography Service" on page 254
- "Data Services" on page 255
- "Dispatch Authorization" on page 255
- "Dispatch Framework" on page 255
- "Email Service" on page 255
- "Event Service" on page 255

- "Exception Service" on page 255
- "Global Cache Service" on page 255
- "Help" on page 255
- "Initialization Service" on page 256
- "Internationalization" on page 258
- "Logging" on page 258
- "Memory Monitor" on page 261
- "Message Type Entitlement" on page 261
- "Object Manager" on page 261
- "Out Of Band Response" on page 261
- "Preferences Service" on page 261
- "Tag Libraries" on page 262
- "Thread Management" on page 262
- "XML Message Converter" on page 263
- "XML Message Service" on page 264
- "XML Services" on page 264

Overview

The Visual Modeler platform architecture enables building the platform in a more modular way, so that changes and upgrades to the platform can be made more quickly and simply, and so that the modules can be re-used to support different products built using them.

The benefits of providing a means of delivering platform functionality in platform modules and requiring that modules make calls to other modules only through their external interfaces areas follows:

- It is easier to compartmentalize the functionality of applications.
- It is easier to understand and manage the dependencies between parts of the Visual Modeler.
- It is easier to contain the customizations to single modules and understand what effect changes made in a module have on the whole system.

- Modules can be more easily upgraded independently of each other, minimizing the effect that an upgrade may have.
- Upgrades to modules that have not been customized will not affect customizations made in other modules.
- New functionality can be delivered in the form of a module that may be dropped into an existing deployment of the Visual Modeler.

Platform Modules

The Visual Modeler platform is developed as a set of interdependent modules that conform to a common organizational structure. In general, each platform module corresponds to a functional component of the Visual Modeler such as a service or a component of the Visual Modeler platform. The platform modules provide a Java API to other modules. Some modules provide a set of “helper” classes which are used by a number of other modules.

In general, each platform module has the following structure:

- Java classes: organized into the following trees. At build time, the directories for the module are assembled into a single JAR file.
 - `com.comergent.api.module`: external API interfaces: used by other modules to access functionality provided by the module. In general, when one module makes a call to another module’s class, it must do so through the other module’s external API. This is the `com.comergent.api` package for the module.
 - `com.comergent.module`: implementation classes: the implementation of the external API interfaces. When another module makes a call to the module’s external API, then the actual classes used are the implementing classes of the module’s interface. The implementation packages may include internal classes: used by the implementation classes, but not exposed to the outside world and not part of the supported Javadoc.
- Configuration files specific to the module such as properties files. These are intended to live in the class hierarchy so that they can be referenced through `getResource()` calls.

Module Interfaces

Each platform module must provide an external interface so that all calls to Java classes and interfaces within the module are invoked through the interface. This

external interface provides a comprehensive set of Javadoc pages so that writers of other modules can use the external interface reliably and easily.

The external interfaces are organized under the following main packages:

- `com.comergent.api`: this package has all the external APIs supported by the modules. These are organized by module:
`com.comergent.api.converter`, `com.comergent.api.logging`, and so on.

Invoking Interfaces

You can invoke an interface from a Java class by casting any object or child interface to the interface and then invoke any method that the interface declares. Each module uses one or other of these techniques, but not both. As you work on an existing module or create a new one, be consistent in how you invoke the interfaces. It will make it easier for your colleagues to work on the same module.

In general, you should always try to work with interfaces provided by the `com.comergent.api` packages: these are the interfaces that the platform modules will support from one release to the next, even though the underlying implementations of the interfaces may change.

Platform Module Descriptions

This section provides a brief description of the purpose of each platform module and examples of its use.

Access Policy

This module provides the service used to check access policies.

Authentication

This module provides the APIs used to authenticate credentials and users.

Base64

This module provides the classes used to convert data to and from Base 64 notation.

Classpath Appender

This module provides classes used to add paths to the classpath.

Cryptography Service

This module provides the services used to encrypt and decrypt data.

Data Services

This module provides a re-packaging and clean-up of the existing data services functionality. Its API has been moved out to a separate `com.comergent.api.dataservices` package. Data services now uses the same preferences mechanism as the rest of the Visual Modeler to manage its properties. Connection pooling has been unified into one pool, and is tunable. Pagination has been updated, and no longer relies on pagination files being written to the file system.

Dispatch Authorization

This module manages access checking that ensures that each user sees only those parts of the application to which they have been granted access.

Dispatch Framework

This module manages the dispatch framework of the Visual Modeler classes that wrap the servlet request, response, context, and session classes together with the base controller classes used by the dispatch mechanism.

Email Service

This module provides the basic APIs to initiate sending email from the Visual Modeler.

Event Service

This module provides the classes used by the EventBus and Events.

Exception Service

This module provides the basic exception framework and classes used by the Visual Modeler.

Global Cache Service

This module provides the APIs to be used to access the cache.

Help

This module provides the `ComergentHelpBroker` class: this is a simple wrapper class to the `ServletHelpBroker` class of the JavaHelp 2.0 implementation.

Initialization Service

The Initialization module provides the Initialization service. This is a package that helps you initialize the Visual Modeler using a consistent framework of classes and methods.

The Initialization Manager provides a focal point in which:

- Initialization tasks can be defined
- Policy on failed initialization can be enforced
- Configuration fragments can be aggregated

The Initialization Manager main responsibility is to act on a list of initialization tasks in a well-defined and predictable manner. That implies an ordered list which:

- either, can be defined programatically
- or, can be specified as an XML-format file

The following code extract provides a typical example of using the `InitManager` class.

```
InitManager initManager = InitManager.getInitManager();
try
{
    String resourceName = args[0];
    initManager.init(resourceName);
    // or programatically created
    //List modules = initModules();
    //ResourceLocator resourceLocator = createNewResourceLocator();
    //initManager.init(modules, resourceLocator);
}
catch (InitManagerException ime)
{
    log.error(ime, ime);
    System.exit(1);
}
// Initialization completed. OK to go on //
...
```

You can specify the initialization process using an configuration file. Here is a sample file:

```
<?xml version="1.0" encoding="UTF-8"?>
<initializationManager>
<resourceLocator>
    <path>/a/b/c</path>
    <path>.</path>
    <path>CLASSPATH</path>
```



```
    </resourceLocator>
<module name="ObjectManager"
  initClass="com.comergent.objectManager.InitHelper"
  <config name="Preferences">
    /com/comergent/objectManager/preferences.xml
  </config>
  <init-param name="param0">param0Value</init-param>
</module>
<module name="module1" initClass="com.comergent.module1.InitHelper"
  <config name="ObjectManager">
    /com/comergent/module1/objectMap.xml
  </config>
  <config name="MessageTypes">
    /com/comergent/module1/messageTypes.xml</config>
  <config name="Preferences">
    /com/comergent/modules1/preferences.xml
  </config>
  <init-param name="param1">param1Value</init-param>
</module>
<module name="module2" initClass="com.comergent.module2.InitHelper"
  <config name="ObjectManager">
    /com/comergent/module2/objectMap.xml
  </config>
  <config name="MessageTypes">
    /com/comergent/module2/messageTypes.xml
  </config>
  <config name="Preferences">
    /com/comergent/modules2/preferences.xml
  </config>
  <init-param name="param2">param2Value</init-param>
</module>
<!-- it is allowable to have no initClass -->
<module name="custom1" >
  <config name="ObjectManager">
    /com/comergent/module1/overlay/objectMap.xml
  </config>
</module>
</initializationManager>
```

In this example, when the following method is called by the Initialization Manager:

```
com.comergent.objmgr.ObjManagerInitHelper.init (initParams,
  configFragments, resourceLocator)
```

the following information is available:

- initParams has a list of key-value pairs: param0-param0Value
- configFragments has a list of:
 - /com/comergent/module1/objectMap.xml

- /com/comergent/module12/objectMap.xml
- resourceLocator can find the resource along the path of: /a/b/c, current, and the current classpath.

Internationalization

This module provides basic support for the internationalization capabilities provided by the Visual Modeler.

Logging

This module provides access to the logging service used to record activity in the Visual Modeler. Its property file, **log4j.properties**, is used to configure the behaviour of the logging service. The module is based on the log4j open source project and uses the same syntax for its configuration as follows:

Log4j has the following main components: *loggers*, *appenders*, and *layouts*. These three types of components work together to enable developers to log messages according to message type and level, and to control at runtime how these messages are formatted and where they are reported.

Configuration

You configure the logging platform module using the log4j.properties configuration file by specifying the properties of its loggers, appenders, and layout. For example, the following snippet is used to configure the root logger and the CMGT appender:

```
# Set root category priority
#log4j.rootCategory=info, CMGT
log4j.rootCategory=info, STDOUT
#log4j.rootCategory=info, CMGT, RTS

### START - CMGT
# CMGT appender
log4j.appender.CMGT=com.comergent.logging.ComergentRollingFileAppender
#log4j.appender.CMGT=com.comergent.logging.ComergentDailyRollingFileAppender

#log4j.appender.CMGT.layout=org.apache.log4j.PatternLayout
log4j.appender.CMGT.layout=com.comergent.logging.ConversionPattern

# The log format defaults to the "classic" format. This format is
# recommended for actual deployment to allow a log analyzer to
# work correctly.
log4j.appender.CMGT.layout.ConversionPattern=%d{yyyy.MM.dd
HH:mm:ss:SSS} Env/%t:%p:%c{1} %m%n
```

Loggers

Loggers are named entities. Logger names are case-sensitive and they follow the hierarchical naming rule: a logger is said to be an ancestor of another logger if its name followed by a dot is a prefix of the descendant logger name. A logger is said to be a parent of a child logger if there are no ancestors between itself and the descendant logger.

For example, the logger named “com.foo” is a parent of the logger named “com.foo.Bar”. Similarly, “java” is a parent of “java.util” and an ancestor of “java.util.Vector”. This naming scheme should be familiar to most developers.

The root logger resides at the top of the logger hierarchy. It is exceptional in two ways:

- It always exists;
- It cannot be retrieved by name.

Invoking the class static *Logger.getRootLogger()* method retrieves it. All other loggers are instantiated and retrieved with the class static *Logger.getLogger(String name)* method. This method takes the name of the desired logger as a parameter. For example:

```
private static final org.apache.log4j.Logger log =
    org.apache.log4j.Logger.getLogger(PriceCheckAPI.class);
log.debug("got current date: " + date);
```

Loggers may be assigned levels. The set of possible levels, that is DEBUG, INFO, WARN, ERROR and FATAL are defined in the *org.apache.log4j.Level* class. If a given logger is not assigned a level, then it inherits one from its closest ancestor with an assigned level. More formally:

Level Inheritance: the inherited level for a given logger, is equal to the first non-null level in the logger hierarchy, starting at the logger and proceeding upwards in the hierarchy towards the root logger.

To ensure that all loggers can eventually inherit a level, the root logger always has an assigned level.

Appenders

The ability to selectively enable or disable logging requests based on their logger is only part of the picture. More than one appender can be attached to a logger.

The *addAppender* method adds an appender to a given logger. Each enabled logging request for a given logger will be forwarded to all the appenders in that logger as well as the appenders higher in the hierarchy. In other words, appenders

are inherited additively from the logger hierarchy. For example, if a console appender is added to the root logger, then all enabled logging requests will at least print on the console. If in addition a file appender is added to a logger, then enabled logging requests for the logger and its children will print on a file and on the console. It is possible to override this default behavior so that appender accumulation is no longer additive by setting the additivity flag to false.

The rules governing appender additivity are summarized below:

- The output of a log statement of logger C will go to all the appenders in C and its ancestors. This is the meaning of the term "appender additivity".
- However, if an ancestor of logger has the additivity flag set to false, then logger's output will be directed to all its appenders and its ancestors up to and including the ancestor, but not the appenders in any of the ancestors the ancestor.
- Loggers have their additivity flag set to true by default.

Layouts

Sometimes, you may wish to customize not only the output destination but also the output format. This is accomplished by associating a layout with an appender. The layout is responsible for formatting the logging request according to your wishes, whereas an appender takes care of sending the formatted output to its destination. The `PatternLayout`, part of the standard `log4j` distribution, lets you specify the output format according to conversion patterns similar to the C language `printf` function.

For example, the `PatternLayout` with the conversion pattern:

```
%r [%t] %-5p %c - %m%
```

will output something like this:

```
176 [main] INFO PriceCheckAPI - got current date: 10/22/2005.
```

The first field is the number of milliseconds elapsed since the start of the program. The second field is the thread making the log request. The third field is the level of the log statement. The fourth field is the name of the logger associated with the log request. The text after the “-” is the message of the statement.

Memory Monitor

This module provides classes used to monitor and log memory consumption.

Message Type Entitlement

This module provides the service that checks the entitlement of users to invoke message types.

The interfaces are defined in the `com.comergent.api.dispatchAuthorization` package. This package contains factory classes, interfaces, and exceptions needed for the service. The implementation classes are in the `com.comergent.dispatchAuthorization` package.

The main entry point for this module is the class `EntitlementRepository`. An instance of this class is obtained from the `EntitlementFactory` class. Applications can create named instances of the `EntitlementRepository` class. Named instances will facilitate unit testing, and may be useful for alternative deployment environments.

An application needing to specify dispatch rules or other message type entitlement objects will execute logic similar to the following:

```
import com.comergent.api.dispatchAuthorization.EntitlementRepository;
import com.comergent.api.dispatchAuthorization.EntitlementFactory;
import javax.xml.dom.Document;
...
Document document = ...;
...
EntitlementRepository repository =
    EntitlementFactory.getEntitlementRepository();
repository.setRules(document);
```

Object Manager

This module provides the classes used to instantiate objects: see "ObjectManager and OMWrapper Classes" on page 238 for details.

Out Of Band Response

This module is used to send output to output streams other than the standard JSP pages.

Preferences Service

The Preferences module is used to retrieve and set configuration properties used by the Visual Modeler. You can retrieve properties along these lines:

```
private static final Preferences prefs =
    Preferences.getPreferences(MyClass.class);
// implicit scope of "com.comergent.apps.module.MyClass"
int max = prefs.getInt("PromotionManager.maxValue", 100);
int min = prefs.getInt("PromotionManager.minValue", 1);
```

The second parameter in the *getInt()* calls specify the value to return if no property with that name is found. The configuration file in which the property is defined is assumed to be on the classpath: for example in the file **com.comergent.apps.module.Preferences.xml**. If the XML properties file is read in using the Preferences service, then make sure that the XML file uses the Comergent root element. For example:

```
<Comergent>
  <PromotionManager>
    <maxValue>50</maxValue>
    <minValue>20</minValue>
  </PromotionManager>
</Comergent>
```

You can ensure that the Preferences service is used to initialize the properties by customizing the **WEB-INF/properties/init.xml** configuration file by adding an element along these lines:

```
<module name="PromotionMgr">
  <config name="Preferences">
    com/comergent/reference/apps/mktMgr/controller/Init.xml
  </config>
</module>
```

The Preferences class provides methods to get and put property values. For example:

```
prefs.putInt("PromotionManager.maxValue", 25);
prefs.putObject("currentShoppingCart", cartBean);
```

When using the *putObject()* method, the object must meet the requirements of the XMLEncoder API: essentially, that the object's fields must provide getter and setter methods.

Tag Libraries

The tag libraries provided by the Visual Modeler are produced as a platform module.

Thread Management

This module provides a centralized facility for handling threads: their creation, obtaining their status, and re-use. It is provided by the **backport-util-concurrent.jar** library. In general, an application developer will no longer have to invoke:

```
Thread t = new Thread(new MyRunnable());
```

Instead, having a centralized facility will allow you to:

- Pool and re-use thread when appropriate
- Track all running threads to help provide better accounting for CPU and resource usage.
- Provide simple status reporting (scoreboard strategy: central shared location where running thread can report its status).
- Provide simple aborting and interrupt signal via *Thread.interrupt()* invocations to allow long running (but looping) thread to quit early.

The module provides the following functionality:

1. Transparently provide pooling and re-use of thread.
2. For administrative functionality, provide means to query all running threads tracked by the thread manager.
3. For user of thread service, provide means to report current thread status to a common scoreboard.
4. Provide guidance to following simple loop or check interrupted status protocol to allow a long running or looping thread to quit early.
5. Provide a timer facility to allow running thread to be notified when a timer expired. This can be used to implement a simple time-out or timeshare policy.

API and Usage

The API will continue to follow the Runnable() pattern: the application obtains a Thread-like object and use it to execute.

```
Executor executor = ExecutorFactory.getPooledExecutor();
executor.execute(new MyComergentRunnable());
```

XML Message Converter

This module provides a facility for converting XML documents from one message category (family and version) to another. The package name for the API is `com.comergent.api.converter` and `com.comergent.converter` for the implementation classes.

The API package includes:

- ConverterFactory: this is the Factory class to create converters.
- Converter: this is the class that converts a document from one message category to another. It can take either documents or streams as the source and targets for conversion.

See "Converter Classes" on page 247 for more information.

XML Message Service

This module is used to create and post outbound messages as XML documents. The API includes `MsgContext` interface, `MsgService` interface, `MsgServiceFactory` class, and the `MsgServiceException` classes in the `com.comergent.api.msgService` package and the implementation classes are in the `com.comergent.msgService` package.

The `MsgService` interface contains a generic `service()` method to post a data bean and an XML document as specified in the message context.

The general usage pattern is as follows:

1. create a `MsgContext` instance using the `MsgContextFactory`;
2. set appropriate attributes on the context object;
3. create a `MsgService` instance for the target message family;
4. post a message by invoking the service method with a data bean and message context.

For example:

```
MsgContext ctx = new MsgContext();
ctx.setMessageType("ERPOrderCreateRequest");
ctx.setURL("http://www.server.com");
ctx.setMessageCategory("ERPOrderCreateRequest");
ctx.setContentType("text/xml");
ctx.setRemoteUser(username);
ctx.setRemotePassword(password);
MsgService msgService =
    MsgServiceFactory.getMsgService(ctx.getMessageCategory());
resultBean = msgService.service(requestBean, ctx);
```

XML Services

This module encapsulates functionality for XML parsing, XSL transformation, DOM wrappers, and utility classes.

Introducing Data Beans and Business Objects

This chapter presents a brief tutorial that demonstrates how you can use the Visual Modeler to work easily with data beans and business objects.

Attention: In Release 6.4 and later, the use of business objects is not supported. You should use data beans wherever possible.
--

What are Data Beans?

A data bean is a data source-independent representation of a real-world entity in the Visual Modeler. The Visual Modeler uses an external schema (defined as a set of XML files) to define the structure of each type of data bean. For example, data beans are used as data structures for users, product inquiry lists, partners, products, and workspaces.

- Use the `OMWrapper` and `ObjectManager` classes to create instances of the `DataBean` classes. See "ObjectManager and OMWrapper Classes" on page 238 for more information.
- You can create a `DataBean` using the `DataManager`. Invoke the `DataManager` method `getDataBean(String beanName)` to create a

DataBean of the named type. This method throws an InvalidBizobjException if no such DataBean class exists.

Note: The use of this method is deprecated because it does not support extensions of the data object.

Lifecycle of a Data Bean

In general, the basic flow of working with a data object is:

1. Instantiate a data bean object using the OMWrapper class.
2. Add data to the bean by using the set methods to directly insert values into the data fields.
3. Persist the data bean to save the new data object to its data source for the first time.
4. Subsequently, you can retrieve the same data object by setting the value for key fields, and then performing a *restore()* on the data bean to retrieve the current data field values from its data source.
5. Perform any business logic required on the data bean. This may change the in-memory values of fields, but not the values stored in the data bean's data source.
6. Save the changes to the data bean by persisting the data bean to its data source.
7. Later, you may want to delete the data object if it is no longer in use.
8. Eventually, you may want to remove the data from the data source entirely by erasing the data object.

In the case of data objects whose underlying data source is a database, the following table summarizes the Java method calls and the corresponding SQL methods called.

TABLE 20. Data Bean Lifecycle

Step	Java Method	SQL Method
Instantiate data object	<i>OMWrapper.getObject()</i>	
Populate data fields	<i>setDataField()</i>	
Persist for the first time	<i>persist()</i>	INSERT
Retrieve data object	<i>restore()</i>	SELECT

TABLE 20. Data Bean Lifecycle

Step	Java Method	SQL Method
Business logic that updates field values	<i>getDataField()</i> <i>setDataField()</i>	
Save changes	<i>persist()</i>	UPDATE
Delete data object	<i>delete()</i>	UPDATE ^a
Erase data object	<i>erase()</i>	DELETE

a. The Delete operation updates the ACTIVE_FLAG column of the underlying database table row: it does not remove the record from the table.

Defining a Data Bean

Data beans are defined using an XML schema. Data beans provide accessor methods to get and set values of particular data fields. In general, you should use data beans when customizing Visual Modeler applications.

Defining the Structure of a Data Object

Each data object must have a defined structure to enable the Visual Modeler to create an instance of the data object. The structure of a data object is defined in its schema XML file: it specifies what fields the data object has and whether it has child objects.

Each data object corresponds to a Java class that extends the `DataBean` class. We refer to these as data bean classes. The data bean classes are generated automatically as part of the SDK merge process. When you generate the corresponding data bean class, it provides methods that access the fields and child data beans that are declared in the data object XML file.

You can change the definition of the XML schema and hence of data objects and their corresponding data bean classes by editing the XML schema files.

The **DsRecipes.xml** configuration file is used to link each data object and its data source. It also specifies whether the ordinality of the data object is “1” or “n”. The data object file is used to specify the precise structure of the data object, and the **DsDataElements.xml** configuration file is used to specify the data type (LIST, LONG, STRING, and so on) of each element.

Extending Data Objects

When you define a data object with an XML schema file, you can declare that it extends another data object by using the `Extends` attribute. This capability is used in two ways:

- You can use one data object as the parent of several different extending data objects which all share a common set of data fields. For example, many data objects in the Visual Modeler extend the C3PrimaryRW data object: this data object provides the basic OwnedBy and AccessKey data fields used to manage access control.
- You can customize a data object by creating a data object that extends it. By adding data fields to the extending data object, you can add attributes that you need to use as part of your customization. By using the ObjectManager, you can ensure that the extending data object is created when the system is called upon to create a data object of the extended type. Provided that existing code uses the ObjectManager to instantiate instances of the extended data object, then when this code is invoked, instances of the extending data object are created, but these still support the extended data object's interfaces, and so the existing code will continue to work.

The DataManager uses a *recipe* and a *data object* to determine the element structure of the data bean or business object and the location of the data source that provides the element values. When you change the definition of data objects or create new definitions, you must re-run the generateDTD and generateBean SDK targets to create and compile the DataBean classes. See CHAPTER 23, "Software Development Kit" for more details. See "Extending Data Objects" on page 280 for alternate ways to extend data objects.

Data Bean and Business Object Creation

The Visual Modeler's ObjectManager and OMWrapper classes create data beans, and business logic classes and controllers process them. See "ObjectManager and OMWrapper Classes" on page 238 for more information.

Business logic classes are invoked by controllers: each controller is responsible for determining which business logic class (if any) must be created in response to a message and its message type.

The use of business objects and the BusinessObject class is deprecated. Where possible, you should use data bean classes, and use business objects only to maintain legacy code.

DataContext

The *restore()* method takes an instance of the DataContext class as a parameter. The DataContext class is used to specify information about the context in which the *restore()* operation is being performed. It can be used to specify the maximum number of results to be returned and for determining the number of results on each

page (pagination). It can also be used to specify whether an access check should be performed on the results of the *restore()* operation. By default, an access check is performed.

For example, the following code snippet creates a *DataContext*, sets some context values, and then uses the context and a query to restore a data bean:

```
DataContext temp_DataContext = new DataContext();
temp_DataContext.setMaxResults(DsConstants.NO_LIMIT);
temp_DataContext.setNumPerPage(-1);
skuMappingListBean.restore(temp_DataContext, query);
```

When a *DataContext* object is initialized, it retrieves from the configuration files values of the *DataService.General.MaxResults* and *DataService.General.NumPerCachePage* element to set these parameters for the restore operation. By default, no limit is set on either. There are accessor methods available if the behavior of the *DataContext* needs to be modified. See the *DataContext* Javadoc for further information.

The *DataContext* class provides a *setCacheId(String cacheId)* method to support pagination: it identifies the particular cache being used.

What is the DataContext class?

The *DataContext* class is used to control the behavior of restore and persist operations.

What behavior can be controlled?

A *DataContext* instance can control the following:

- How many query results appear on a page.
- The maximum number of query results that will be processed.
- The use of multiple page sets per Data Bean type and Session.

What are the Cache Id methods for?

The Cache Id methods allow an application to specify a unique identifier for pagination of result sets. This new capability allows an application to maintain multiple distinct result sets for a given Data Bean and Session.

If an application does not specify a Cache Id then a combination of Bean name and Session Id are used to identify the cache. In this case any subsequent attempt to restore the same Data Bean within the same session will overwrite any results.

The *DataContext* class provides the following methods to control Cache Id on Data Bean restore requests:

- void setCacheId(String cacheId): Sets a new cache id. This string is used in combination with the Bean name and session id to generate a unique identifier.
- String getCacheId(): Returns the current cache id (or null if it is not set).

How do Max Results and Num Per Page work?

The setting of Max Results determines the maximum number of records that can be retrieved during a restore. When that number is reached the request is freed.

The setting of Num Per Page determines how many records are saved in each result cache page. If the number found is less than Num Per Page, then no result cache is created.

Note that this combination of attributes allow the application to retrieve a set of paginated results while still specifying a maximum number of records to retrieve.

The DataContext class provides the following methods to Max Results and Num Per Page on Data Bean restore and persist requests:

- void setMaxResults(int maxResults) sets the maximum number of results returned for non-paginated results
- int getMaxResults() gets the maximum number of results to return for non-paginated results
- void setMaxPaginatedResults(int maxResults) sets the maximum number of results returned for paginated results
- int getMaxPaginatedResults() gets the maximum number of results to return for paginated results
- void setNumPerPage(int numPerPage)
- int getNumPerPage()

If an application wants to use the data services default limits, the appropriate property in DataContext must be set to **DsConstants.USE_DEFAULT**. The following are the default values:

- maxResults: 125
- maxPaginatedResults: 125
- numPerPage: 25

If the application does not specify a value for `numPerPage`, then the value specified in `prefs.xml` will be used. If a value is not set by the application nor the `prefs.xml` file, a value of -1 will be used, which means the request will not be paginated.

In addition, the following methods provide result set limits that are passed directly to the database as part of the SQL query. Since the Visual Modeler may discard results as part of its access policy checking (for example, does the user have the right to see this data?), these methods allow you to set a higher result set limit.

- `public void setDBResultLimit(int limit)`
- `public int getDBResultLimit()`

You can also set the `DataServices.General.LimitDBResults` preference. If `LimitDBResults` is set to true, results are automatically limited to the number allowed by `MaxResults` (or by `MaxPaginatedResults` for paginated results). Access policies must be expressed as SQL to use this mechanism. For Oracle databases, do not set the `LimitDBResults` preference to true.

Our access policies are handled in one of two ways. Many are converted to SQL WHERE clauses that are applied to the query. This allows the database to handle the access policy. If the policy is too complex (for example, it relies on a hierarchy of partners), then the access policy can be applied only when processing the results from the database. Such policies cannot be converted to SQL.

With Oracle, there are some cases in which the SQL generation will require that column aliases be defined in the XML schema. This is necessary only when the query joins multiple tables that use the same column name. This is not an issue for SQL Server.

How do I instantiate a DataContext instance?

A new `DataContext` instance is currently instantiated using the standard “new” mechanism:

```
DataContext dc = new DataContext();
```

What are the Default Settings for a new DataContext?

When “new DataContext()” is invoked, the attributes receive the following default values:

TABLE 21. DataContext Default Values

Attribute	Default Value
doAccessCheck	true
maxResults	DataServices.xml maxResults property
numPerPage	DataServices.xml numPerPage property
CacheId	null

List Data Beans

A special class of business objects are called *list data beans* and *list business objects*. You use these classes to manage a list of data objects of the same type. Whenever a data object element is declared with ordinality “n” in a Recipe element, then a list data bean is created. Access entitlements are still managed at the level of the singular business object.

Note: Earlier versions of data objects defined ordinality in the data object definition file. Now it is the recipe file that determines the ordinality of a data object. In Version 6.0 data objects, the ordinality attribute is still used to declare child, reference, and included data objects.

In general, you do not need to create DataBeans for list data objects: they are created automatically. See "DataBean Classes" on page 237 for more information. They support automatically generated methods that return a list of the data objects. For example, the following code fragment demonstrates how to restore a list of users. A DataContext object identified by “context” and a DsQuery object identified as “query” are used to restrict the users returned by the *restore()* call:

```
UserListBean userList = (UserListBean)
    OMWrapper.getObject("com.comergent.bean.simple.UserListBean");
// Restore the list.
userList.restore(context, query);
// Return immediately if no results found.
if (userList.getUserCount() == 0)
{
    return;
}
// At least one user in list, so walk through the list of users
ListIterator userIterator = userList.getUserIterator();
while (userIterator.hasNext())
```



```
{
    UserBean user = (UserBean) userIterator.next();
    // Perform any business logic on each user.
}
```

Note the use of the `DataContext` and `DsQuery` parameters in the `restore()` method: these are used to manage how the query is executed against the Knowledgebase.

Application, Entity, and Presentation Beans

There are several main sorts of data beans used in the Visual Modeler: data beans, application beans, entity beans, and presentation beans. This section describes the main differences between them.

- Data beans are the Java classes created automatically from the XML schema description of the business objects. Running the `generateBean` SDK target generates the source code for each data bean. These beans comprise the `com.comergent.bean.simple` package.

Where possible, you should use the `instanceof` command to determine the class of a data bean rather than querying for the business object type.

- Application beans are Java classes created to add functionality that simple beans do not support. For example, an application bean may provide extra methods that cannot be automatically generated, or it may combine two or more simple beans to pass data to a JSP page. The application beans are organized by application and each application has a package for its application beans whose name is `com.comergent.apps.<application name>.bean`

Application beans can be subclasses of simple beans, but more often they are Java classes that contain one or more simple beans as member variables.

For example, the

`com.comergent.appservices.productService.productMgr.BizProductBean` application bean class is a Java class that contains a member variable that implements the `com.comergent.bean.simple.IDataProduct` interface. The `BizProductBean` application bean class delegates methods such as `getProductID()` to the `com.comergent.bean.simple.IDataProduct` member variable, but in addition it provides methods to retrieve a product's features, its supersession chain, and prices. Note the use of the `IDataProduct`

interface rather than the `ProductDataBean` itself: this is an example of using a generated interface rather than the class. See "Generated Interfaces" on page 315 for more information on the generation and use of these interfaces.

By convention, if you create an application bean to wrap a data bean, then you must provide a method called `getDataBean()` that retrieves the data bean.

- Presentation beans are also used to pass data to JSP pages: typically, they differ from application beans in that they do not provide business logic. They may aggregate several data beans into a single class for ease of use, or provide formatting information. As with application beans, presentation beans must provide a method to provide access to the underlying data bean. For example, the `IPresProduct` interface provides the `getIRdProduct()` method: this returns the `IRdProduct` interface and you can downcast this to the underlying data bean or extended data bean if need be.
- Entity beans were used in prior releases of the Visual Modeler. They performed the same role as application beans. Their use is deprecated.

Using Stored Procedures

You can make use of stored procedures to restore data objects. The name of the stored procedure is declared in the `ExternalName` element of the data object.

When you define data objects, take care to specify the `SourceType` attribute. It can take the following values:

- "1": the underlying data source uses a table. This is the default value.
- "2": the underlying data source uses a stored procedure.

If no `SourceType` attribute is defined, then the default value means that a table is the underlying source type for the data object.

Data Bean Methods

In general, you should make use of the generated interfaces that each data bean provides: these organize the accessor and data methods to help you manage access to the data objects during their lifecycle. See "Generated Interfaces" on page 315 for more information.

Use the access policy security mechanism to provide access control.

IData Methods

The IData interface has these important methods:

- *copyBean()*: this method can be used to copy the values of data fields from one bean to another. It takes one argument: this must be a bean that is either an instance of the same class or a sub-class of the bean invoking this method.
- *delete()*: this method marks the corresponding data object as deleted: the ACTIVE_FLAG column of the database table corresponding to this data object is set to “N” when the object is persisted. Note that you must call *persist()* after calling *delete()*: if you do not, then the deletion does not take effect.
- *erase()*: this method removes the database record corresponding to the business object. Note that removing records from database tables can lead to data integrity problems if other tables refer to keys that have been deleted. In general, you should use this method only if you can account for all usages of the record and its keys and can delete the corresponding records from other tables.
- *generateKeys()*: this method populates the key fields of the data bean. You can call this method without invoking *persist()*. By invoking this method, you can use the generated keys to create other objects that require the keys.
- *setDataContext()*: this method sets the data context so that *restore()* and *persist()* calls use the right values for parameters such as the number of results per page in a paginated data set. See "DataContext" on page 268 for more information on the DataContext class.
- *persist()*: this method saves the data in the data bean to its data source.
- *prune()*: this method is used to mark the bean for deletion in memory. Calling *restore()* after *prune()* has no effect on the bean’s underlying data source.
- *restore()*: this method retrieves the data for the data bean from its data source. See "DataContext" on page 268 for information on the use of the DataContext class in the *restore()* method.
- *update()*: this method updates the database record corresponding to this business object.

Note that any method calls that change state must be followed by a *persist()* call to actually make the change to the database record.

The IData interface also provides the methods, *isRestorable()* and *isPersistable()*, that check whether a data object may be restored or persisted respectively.

IRd and IAcc Interface Methods

The IRd interface provides the read-only accessor methods to the data object fields. The IAcc interface extends the IRd interface by adding the set accessor methods for each data field. Distinguishing between these two interfaces provides you with the ability to pass a read-only object to a client application or JSP page.

For example, suppose that in the Condition data object file, **Condition.xml**, a DataField element is specified as follows:

```
<DataField Name="ControlType"
  Writable="y" Mandatory="y"
  ExternalFieldName="CONTROL_TYPE"/>
```

Then, in the automatically-generated IRdCondition interface, there is a method called:

```
public Long getControlType()
```

In the automatically-generated IAccCondition interface, there is a method called:

```
public void setControlType(Long value) throws ICCEException
```

The signatures of these accessor methods is determined by the corresponding DataElement definition in the **DsDataElements.xml** file:

```
<DataElement Name="ControlType" DataType="LONG"
  Description="Condition Control Type" MaxLength="20" />
```

<p>Note: If you set the Writable attribute of a data field to “n”, then the corresponding <i>setDataField()</i> method is not generated.</p>

Restoring and Persisting Data

These important operations may be performed on a data object: *delete()*, *persist()*, and *restore()*.

- By calling the *delete()* method on a data object, you mark this object as deleted, and no other application will retrieve this data object. The ACTIVE_FLAG column of the underlying database table has its value set to 'N'. Note that the data object data is not deleted from the data source. If the underlying database table for data object does not have an

ACTIVE_FLAG column, then do not use the *delete()* method. You can still use the *erase()* method to remove such data objects from the Knowledgebase.

- When you *persist* a data bean, the Visual Modeler saves the data held in the data object's DsElement tree to its external data source(s). Note that the Visual Modeler manages both the update of existing data objects and the creation of new data objects with the *persist()* method.
- When you *restore* a data bean or business object the Visual Modeler retrieves its data from its external data source(s). If no query object is specified in the *restore()* method, then all of the data objects whose values in the key fields match those in the data bean are restored.
 - Note that if you call *restore()* on a non-list data bean, then you should expect that its data is uniquely retrievable from the values set in its key fields. When the *restore()* call is issued, no check is performed to verify that only one record is retrieved, and so the first record retrieved will be used to populate the data bean. If no record is retrieved, then the *restore()* call throws an ICCEException.
 - When you call *restore()* on a list data bean, then you must usually specify a DsQuery. If no DsQuery is specified, then the restored list data bean will contain all the data beans of this type.

restore() Method

This section provides description of the main forms of the DataBean *restore()* method.

```
public void restore(DataContext dataContext, DsQuery dsQuery)
```

The principal form of the *restore()* method. Use the dsQuery parameter to specify query to be executed by the restore operation. The dataContext parameter determines the maximum number of objects returned, and for pagination the number of results per page. Use the dataContext parameter to specify whether to check that the current user has the correct entitlements to perform this operation. By default, an access check is performed, so you have to override the access check if you do not want this to be done, using the *disableAccessCheck()* method.

```
public void restore(DataContext dataContext)
```

This is equivalent to calling *restore(dataContext, null)*.

Here is an example of using the DataContext and DsQuery classes together to manage the *restore()* call:

```
try
{
    DataContext dataContext = new DataContext();
    if (doAccessCheck == true)
    {
        dataContext.enableAccessCheck();
    }
    else
    {
        dataContext.disableAccessCheck();
    }
    dataContext.setNumPerPage(pageSize);
    DsQuery dsQuery = QueryHelper.newWhereClause("PartnerKey",
        DsConstants.EQUALS, partnerKey);
    LightweightPartnerBean partnerBean =
    (com.comergent.bean.simple.LightWeightPartnerBean)
    com.comergent.dcm.util.OMWrapper.getObject(
        "com.comergent.bean.simple.LightWeightPartnerBean");
    partnerBean.restore(dataContext, dsQuery);
    QueryHelper.freeQuery(dsQuery);
    return partnerBean;
}
catch (ICCEException e)
{
    throw (new ProfileMgrException(e));
}
```

persist() Method

This section provides description of the main forms of the *DataBean persist()* method.

```
public void persist(DataContext dataContext)
```

If the *dataContext* specifies that an access check should be performed, then this form of the *persist()* method performs an access check before performing the operation. If the user does not have the appropriate entitlement, then the operation is not performed.

Miscellaneous Methods

getBizObj() Method

If you want to retrieve a business object representation of the data object and its data, then you can invoke the *getBizObj()* method. This is useful if you want to display the internal structure of the object. For example:

```
BusinessObject bo = bean.getBizobj();
ComergentDocument doc = bo.serializeToXml();
doc.prettyPrint();
```

Note that this is now a deprecated method.

writeExternal() Method

Use this method to write out an XML representation of the data bean and its data.

Child Data Objects

Many data objects declare child data objects using the `ChildDataObject` element. For example, the `ShoppingCart` data object declares `LineItem` as a child data object as follows:

```
<DataObject Name="ShoppingCart" Extends="C3PrimaryRW"
  ExternalName="CMGT_CARTS" ObjectType="JDBC" Version="6.0">
  ...
  <ChildDataObject Access="RWID" Name="LineItem">
    <Relationship CascadeDelete="y" CascadeErase="n"
      ChangeUpdatesParent="y">
      <JoinKeys>
        <JoinKey DstJoinField="ShoppingCartKey"
          SrcJoinField="ShoppingCartKey"/>
      </JoinKeys>
    </Relationship>
  </ChildDataObject>
  ...
</DataObject>
```

Its `Relationship` element has attributes that describe how child objects should be managed when the parent is updated and whether to update the parent when a child is changed. The `JoinKey` elements describes how to restore the child data objects: typically, by specifying how values in the parent data object are used to set values in the child data object.

When the parent data bean is generated, it generates a method called `getChildDataObjectIterator()` which returns an `ListIterator` object containing the child data beans. By iterating through the objects, you can examine each child data bean in turn and access its fields using the standard accessor methods.

For example, the `ShoppingCartBean` class supports the `getLineItemIterator()` method. The following lines of code demonstrate how to retrieve a field of a line item:

```
/*
shoppingCartBean is a ShoppingCartBean object that has already been
restored
*/
ListIterator lineItemIterator =
    shoppingCartBean.getLineItemIterator();
LineItemBean lineItemBean =
```

```
(LineItemBean) lineItemIterator.getLineItemBean(0);
Long quantity = lineItemBean.getQuantity();
```

When a parent data object is restored, the child data objects are not restored. They are restored only when the application accesses the children as described above.

Extending Data Objects

It is common for any implementation of the Visual Modeler to need to add data fields to data objects or to create data objects that extend existing data objects.

We recommend storing the additional data in a new database table. A new `DataObject` should then be defined that accesses the new table. Another new `DataObject` is then defined that extends the original `DataObject` by adding a new `IncludeDataObject`.

For example, suppose that you need to add a new data field to the Order data object to track “hosted” orders: orders that are placed at storefront partners. The extra data field is the partner key of the storefront partner. The recommended approach is as follows:

1. Create a new data object called `HostedPartner` that has exactly two fields: an `OrderKey` and a `PartnerKey`. Set it up to point to a two-column table: `CMGT_ORDER_X_PARTNER` with columns `ORDER_KEY` and `PARTNER_KEY`.

```
<?xml version="1.0"?>
<DataObject Name="HostedPartner"
  ExternalName="CMGT_ORDER_X_PARTNER" ObjectType="JDBC"
  Version="6.0">
  <KeyFields>
    <KeyField Name="OrderKey" ExternalName="ORDER_KEY"/>
    <KeyField Name="PartnerKey" ExternalName="PARTNER_KEY"/>
  </KeyFields>
  <DataFieldList>
    <DataField Name="OrderKey" ExternalFieldName="ORDER_KEY"
      Mandatory="n" Writable="y"/>
    <DataField Name="PartnerKey"
      ExternalFieldName="PARTNER_KEY"
      Mandatory="n" Writable="y"/>
  </DataFieldList>
</DataObject>
```

2. Create a new data object called `HostedOrder` that extends `Order`. The **HostedOrder.xml** file looks like this:

```
<?xml version="1.0"?>
<DataObject Name="HostedOrder" Extends="Order" ObjectType="JDBC"
  Version="6.0">
```



```
<IncludedDataObject Access="RWID" Name="HostedPartner"
  Ordinality="1">
  <Relationship CascadeDelete="y" CascadeErase="n"
    ChangeUpdatesParent="y">
    <JoinKeys>
      <JoinKey DstJoinField="OrderKey"
        SrcJoinField="OrderKey"/>
    </JoinKeys>
  </Relationship>
</IncludedDataObject>
</DataObject>
```

There are three basic approaches that can be used:

1. You can use extension to simply add any additional DataFields and override the table name. This allows you to include all of the data in a new table. This approach is most useful when you need the same data, but need a distinct copy of it. (Perhaps you maintain a snapshot of how an Order looked before it was turned into a HostedOrder)
2. You can extend Order to add an IncludedDataObject for HostedOrder, where HostedOrder only defines additional data for storage in another table. This means that changes to the original Order DataFields will still be persisted to the Order table, but the additional data for HostedOrder will be persisted to a different table. This is the recommended approach described above.
3. You can define HostedOrder specifying that Order is a IncludedDataObject. This accomplishes the same thing as the second alternative. The problem with this approach is that a HostedOrder does not extend Order, and can no longer be treated as an Order by application code.

Note: Using two tables has a slight disadvantage in performance, but query execution has not been a problem area. Using two tables may reduce data redundancy (depending on your requirements).

If you only occasionally reference the customer extension, then you may want to use a ChildDataObject to take advantage of the lazy link mechanism.

Data Bean Example

This section presents the process of defining and using a data object. Suppose that you want to use a data object to represent a simple enquiry from a customer. This will comprise:

- an email address for the customer

- the date the enquiry was made
- the date a response was returned (optional)
- the content of the enquiry
- the content of the response (optional)
- the product ID of the product about which the enquiry was made (optional)

To Create a Data Object Definition

1. Create the business object element Enquiry and add it to the **DsBusinessObjects.xml** file.

```
<BusinessObject Name="Enquiry" Version="6.0"  
  Description="Customer enquiry"/>
```

Use the Version attribute to manage different versions of business objects that may be in use simultaneously. Note that the Version attribute is also used to determine whether access checks are performed automatically (Version 5.0 or higher) or not.

2. Create the recipe for this business object and add it to the **DsRecipes.xml** file.

```
<Recipe Name="Enquiry" Version="6.0" Ordinality="n"  
  Description="Customer enquiry">  
  <DataObjectList>  
    <DataObject Name="Enquiry"  
      DataSourceName="ENTERPRISE" />  
  </DataObjectList>  
</Recipe>
```

The Name attribute of the recipe must match exactly (it is case-sensitive) to the Name of the business object. In Release 9.0, each recipe may have more than one data object defined in the data object list, but only one may be a *writable* data object. The data objects define the data source names as an attribute of each data object element. It is these entries that determine the sources from which the business object retrieves its data and the source to which the business object may be persisted.

3. Create a file called **Enquiry.xml** to define the data object. The Name of the data object element must match exactly (it is case-sensitive) the Name attribute defined in the DataObject entry of the recipe element.

In this example, the data for these data objects is held in a database table called CMGT_ENQUIRY, and the ExternalFieldName attribute of each DataField element specifies which column is to be used to retrieve the DataField value.

For example, the EMAIL_ADDRESS column of the CMGT_ENQUIRY table holds the email address value associated with an enquiry.

```
<?xml version="1.0"?>
<DataObject Name="Enquiry" Extends="C3PrimaryRW" Version="6.0"
  ExternalName="CMGT_ENQUIRY"
  Access="R" ObjectType="JDBC">
  <KeyFields>
    <KeyField Name="Key" ExternalName="ENQUIRY_KEY"/>
  </KeyFields>
  <DataFieldList>
    <DataField Name="EnquiryKey"
      Writable="n" Mandatory="y"
      ExternalFieldName="ENQUIRY_KEY"/>
    <DataField Name="EmailAddress"
      Writable="n" Mandatory="y"
      ExternalFieldName="EMAIL_ADDRESS"/>
    <DataField Name="EnquiryDate"
      Writable="n" Mandatory="y"
      ExternalFieldName="ENQUIRY_DATE"/>
    <DataField Name="ResponseDate"
      Writable="n" Mandatory="n"
      ExternalFieldName="RESPONSE_DATE"/>
    <DataField Name="TimeToRespond"
      Writable="n" Mandatory="n"/>
    <DataField Name="EnquiryContent"
      Writable="n" Mandatory="y"
      ExternalFieldName="ENQUIRY_CONTENT"/>
    <DataField Name="ResponseContent"
      Writable="y" Mandatory="n"
      ExternalFieldName="RESPONSE_CONTENT"/>
    <DataField Name="SKU"
      Writable="n" Mandatory="n"
      ExternalFieldName="SKU"/>
  </DataFieldList>
</DataObject>
```

Note the definition of the TimeToRespond data field: it has no ExternalFieldName attribute because it does not correspond to a database column. Values for this field are calculated at runtime and are set in the EnquiryBean so that its value can be displayed.

4. Define Enquiry and EnquiryList DataElements in **DsDataElements.xml**:

```
<DataElement Name="Enquiry" Description="Enquiry"
  DataType="HEADER"/>
<DataElement Name="EnquiryList" Description="Enquiry list"
  DataType="LIST"/>
```

5. Define a DataElement for each DataField in **DsDataElements.xml**. DataElements provide data type information used by the DataManager when it is retrieving or saving data for this business object type. For example:

```
<DataElement Name="EnquiryKey" LongName="Enquiry Key"
  DataType="LONG"MaxLength="20" />
<DataElement Name="EnquiryDate" LongName="Enquiry Date"
  DataType="DATE" />
<DataElement Name="ResponseDate" LongName="Response Date"
  DataType="DATE" />
<DataElement Name="EnquiryContent" LongName="Enquiry content"
  DataType="STRING" MaxLength="256" />
<DataElement Name="ResponseContent" LongName="Response content"
  DataType="STRING" MaxLength="256" />
```

Note that we have not included a DataElement for EmailAddress and SKU. The DataElements for these DataFields are already defined and you can re-use DataElements any number of times (as long as the data type is the same in each occurrence).

6. Create entries in the **ObjectMap.xml** file for this data bean. For example:

```
<Object ID="com.comergent.bean.simple.EnquiryBean">
  <ClassName>com.comergent.bean.simple.EnquiryBean</ClassName>
</Object>
<Object ID="com.comergent.bean.simple.IRdEnquiry">
  <ClassName>com.comergent.bean.simple.EnquiryBean</ClassName>
</Object>
<Object ID="com.comergent.bean.simple.IAccEnquiry">
  <ClassName>com.comergent.bean.simple.EnquiryBean</ClassName>
</Object>
<Object ID="com.comergent.bean.simple.IDataEnquiry">
  <ClassName>com.comergent.bean.simple.EnquiryBean</ClassName>
</Object>
```

7. Finally, define a data source element to correspond to the DataSourceName attribute defined in the DataObject element. This data source is defined in the **DsDataSources.xml** file as part of the schema. In most cases, this data source will already be defined: You only need define a new one if you are using a different database or other data source than the rest of the Knowledgebase. For example:

```
<DataSource Name="ENTERPRISE" Version="2.0">
  <Primary Type="SQL" DataService="JdbcService"
    SubType="ORACLE"
    ConnectionString="jdbc:<driver>:<server>:<port>:<sid>"
    UserId="userid" Password="password" />
  <Alternate Type="SQL" DataService="JdbcService"
    SubType="MSSQL"
```

```
        ConnectionString="jdbc:<driver>:<server>:<port>:<sid>"
        UserId="userid" Password="password" />
    </DataSource>
```

The `DataService` attribute of the `Primary` and `Alternate` elements determine which class is used to process the `EnquiryBean.restore()` and `persist()` methods. The remaining attributes determine exactly how the external source is accessed.

8. Run the `generateBean` SDK target to generate the source code for the new `EnquiryBean` and `EnquiryListBean` data beans and the corresponding interfaces. See "Generated Interfaces" on page 315 for more information on these interfaces.

You can now use `Enquiry` data beans and its interfaces in business logic classes. To create an instance of an `Enquiry` data bean, you invoke a method of the form:

```
OMWrapper().getObject("com.comergent.bean.simple.EnquiryBean")
```

This returns an `EnquiryBean` data bean and its structure is as specified in the `Enquiry DataObject`. Once you have an instance of the `QueryBean` class, then populate its key fields and restore the bean to retrieve its data:

```
int queryIndex = 0;
try
{
    String queryKey = request.getParameter("querykey");
    queryIndex = Integer.parseInt(queryKey);
}
catch (Exception e)
{
    //Throw exception if parameter not valid
}
QueryBean queryBean = (QueryBean)
    OMWrapper().getObject("com.comergent.bean.simple.EnquiryBean");
queryBean.setKey(queryIndex);
queryBean.restore();
```

To retrieve a list of enquiries:

```
// Use default settings for DataContext parameters
DataContext context = new DataContext();
// Retrieve enquiries relating to a particular product ID, MXWS-7000
DsQuery query =
    QueryHelper.newWhereClause("SKU", DsQueryOperators.EQUALS,
        "MXWS-7000");
EnquiryListBean enquiryList = (EnquiryListBean)
    OMWrapper().getObject("com.comergent.bean.simple.EnquiryListBean");
// Restore the list.
enquiryList.restore(context, query);
// Walk through the list...
```

```
ListIterator enquiryIterator = enquiryList.getEnquiryIterator();
while (enquiryIterator.hasNext())
{
    boolean isModified = false;
    EnquiryBean enquiry = (EnquiryBean) enquiryIterator.next();
    // Process each enquiry
}
```

In general, you should try to ensure that applications that use the EnquiryBean use one of the generated interfaces rather than the data bean itself. This will enable the application to separate out the implementation of the data object from its interface and let you manage what access the application has to the object's data. To retrieve an instance of a class that implements the IAccEnquiry interface, you can use:

```
IAccEnquiry temp_IAccEnquiry = (IAccEnquiry)
    OMWrapper.getObject("com.comergent.bean.simple.IAccEnquiry");
```

DsElement Tree

This section describes methods to retrieve metadata about databeans. It also describes the DsElement tree used to store data in the data object and business object classes. It is covered here only to support legacy applications: all new applications that use the data bean classes should not need to be concerned with it.

Data objects are created as objects of data bean classes. Each data object holds its content as a tree of components called DsElements (see "DsElements" on

page 287). Their content is retrieved from external systems using the XML schema, and the recipes and data sources defined in the XML schema.

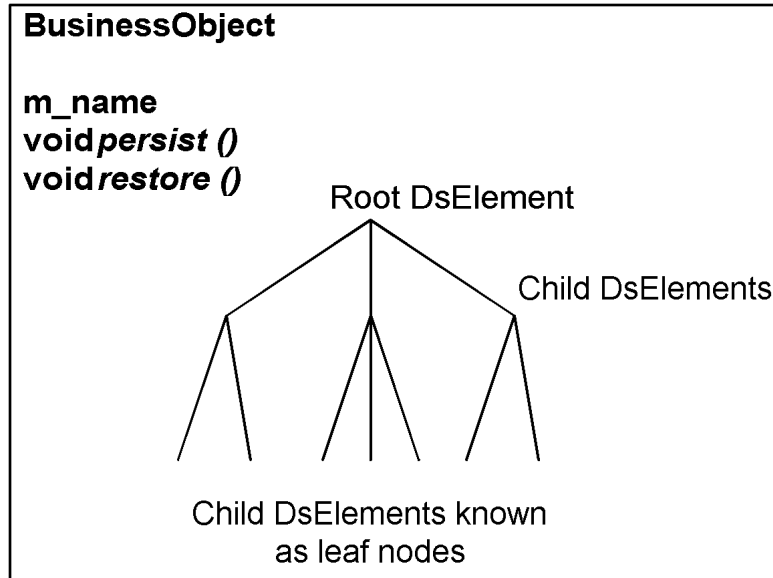


FIGURE 11. Business Object

When the DataManager creates a data bean or business object, it uses the XML schema to determine the structure of its DsElement tree. The DsElement tree is the Java representation of the structure of the business object. The schema also determines the data types that may be inserted at leaf nodes and whether constraints are placed on the values of the node. You access the DsElement tree by invoking the business object method *getRootElement()*.

DsElements

Each DsElement contains data and a DataMap that defines how its data corresponds to its data source. A DsElement may be the child of another DsElement (its *parent*). A DsElement tree is a collection of DsElements, all but one of which have another element in the tree as its parent. By definition, the DsElement with a null parent is the *root* DsElement.

DsElement

m_children
m_parent
m_dataMap
m_value

DsElementcloneDsElement ()
DsElementaddChild (DataMap dataMap)
void delete ()
String getName ()
int getType ()
DsElementgetParent ()
DsElementgetElementByName (String s)
void deleteChild (DsElement child)

FIGURE 12. DsElement Methods

The DsElement class provides various additional methods to support navigating through a DsElement tree, notably *children()* that returns an Iterator of the child DsElements of a given DsElement. As well as *getRootElement()*, the business object class also provides the *getElementByName()* method to access directly a named DsElement in its tree.

All DsElements that have the same name, for example *child_name*, and which are children of a DsElement must have a parent whose name is *<child_name>List*. The XML schema identifies such elements by defining their ordinality to be “n” as opposed to “1”. A DsElement maintains its children in a Vector called *m_children*.

The DsElement has these important methods:

- *addChild()*: adds a new DsElement defined by the DataMap of this DsElement.
- *cloneDsElement()*: returns a copy of this DsElement.
- *delete()*: sets the DsElemState to DsElemState.DELETED.

- *deleteChild()*: removes a child from the vector `m_children` by specifying it as a `DsElement`.
- *getName()*: returns the name of the element as defined by its `MetaData`.
- *getParent()*: returns the parent of this `DsElement`.
- *getType()*: returns the type of the element as defined by its `DataMap`.

DsElement MetaData

It is sometimes useful to retrieve information about a data field and its underlying `DsElement`. You can use the `IData` interface method *getMetaData(String elementName)* to this. It returns an object that implements the `IMetaData` interface. This interface supports the following methods:

- `public int getDataType()`: returns values as defined in `DsDataTypes`
- `public long getMaxLength()`: returns maximum length in bytes
- `public long getMaxCharLength(Locale locale)`: returns maximum length in characters
- `public Object getMinValue()`: returns the minimum allowed value (or null if there is no minimum)
- `public Object getMaxValue()`: returns the maximum allowed value (or null if there is no maximum)
- `public int getCountAllowedValues()`
- `public ListIterator getAllowedValueIterator()`
- `public Object getDefaultValue()`

Note that each generated `DataBean` class implements the `IData` interface, and so these methods are available to all the generated data beans.

BusinessObject Methods

Use of business objects is deprecated. This section provides information about some business object methods for reference only.

restore() Method

This section provides description of the main forms of the `BusinessObject restore()` method.

```
public void restore(BusinessObject queryObj, int maxResults,  
    boolean accessCheck)
```

The principal form of the *restore()* method. Use the *queryObj* parameter to specify query to be executed by the restore operation. The *maxResults* parameter determines the maximum number of objects returned. Use the *accessCheck* parameter to specify whether to check that the current user has the correct entitlements to perform this operation. Once the access check has been performed, then the *restore(BusinessObject queryObj, int maxResults)* is called.

```
public void restore(BusinessObject queryObj, int maxResults)
```

This method calls the *restore()* method *restore(this, queryObj, maxResults, false)* of the underlying data object.

```
public void restore(BusinessObject queryObj)
```

This is equivalent to calling *restore(queryObj, 0)*.

```
public void restore()
```

This form of the method calls the *restore(null, 0)* method.

***persist()* Method**

This section provides descriptions of the main forms of the *BusinessObject persist()* method.

```
public void persist(boolean synch, boolean commit,  
    boolean accessCheck)
```

The boolean parameters determine respectively whether the *persist* operation is synchronized, should be committed to the underlying data source, and whether an access check should be performed prior to persisting.

```
public void persist(boolean synch, boolean commit)
```

This form of the method is equivalent to *persist(synch, commit, false)* for business objects whose *Version* attribute is 4.0 or less. It is equivalent to *persist(synch, commit, true)* for business objects whose *Version* attribute is 5.0 or more.

```
public void persist()
```

This form of the method calls *persist(false, true)*.

The *BusinessObject* class also has these methods:

- *delete()*: empties the business object by deleting its *DsElement* tree.
- *getRootElement()*: returns the root *DsElement* of the *DsElement* tree.

- *getType()*: returns the name of the root element of the DsElement tree. This is the type of the business object.
- *setRootElement()*: sets the root element of this business object.

The Visual Modeler offers developers several mechanisms to manage security in their applications. This chapter describes how you can use the entitlements, access control lists, and access policies to manage what users can do: what functions they may perform and what access they have to data objects. It covers:

- "Managing Message Types" on page 293
- "Managing Access to Data Objects Using Access Policies" on page 296
- "Password Policies" on page 302
- "Passing Login Data Through a URL" on page 304

Managing Message Types

As you customize the Visual Modeler, you must take into account which types of users can execute which message types and which Web pages should be accessible to which users.

Each message type corresponds to a request that the user's browser makes to the server. Message types are organized into message groups. A role is defined as a collection of message groups that are either granted or denied to the role.

```
<RoleDefinition Name="Partner.SalesRep">  
  <Description>
```

```
        This is the role associated with the Lead Users.  
        Lead Users can work leads that are assigned to them.  
    </Description>  
    <Grant>LeadMgmtDetailGroup</Grant>  
    <Grant>ProposalGroup</Grant>  
</RoleDefinition>
```

In Release 9.0, roles are aggregated into functions: a function is intended to be the collection of roles that correspond to a business function such as finance or sales. Users are assigned functions, and the set of functions available to be assigned to a user depends on their user type. A function is declared in the **Entitlements.xml** configuration file using an element of this form:

```
<UserFunctionMapping Name="IndirectSalesExecutive">  
    <Description>Sales</Description>  
    <Role>Partner.IndirectBuyer</Role>  
    <Role>Partner.SalesRep</Role>  
    <Role>Partner.SalesManager</Role>  
    <Role>Partner.CustomerServiceRepresentative</Role>  
</UserFunctionMapping>
```

The same role can be included in more than one function. Consequently, you can define functions that overlap in some roles, or define a function that is only a subset of another function.

Checking for Entitlement

The system will test whether a user can execute a message type when a request is received. However, to prevent users from seeing error pages, in general, you should perform an entitlement check for each link on a JSP page to test that the user can execute the message type associated to the link.

You can use the *canRequest(String messageType)* method of the User class. You can retrieve the current User object from the session as follows:

```
User sessionUser = comergentSession.getUser();
```

For example, the following lines in a JSP page are used to determine whether the current user can access a promotion detail page.

```
User sessionUser = comergentSession.getUser();  
...  
<% if (sessionUser.canRequest("PromotionDetailDisplay"))  
    {  
%>  
    <A HREF="<%= link("partnerMkt", "PromotionDetailDisplay",  
        "PromotionKey=<%= promotion.getKey() %>") %>" %>> %>">  
    <%= ph(promoName) %></A>  
<%
```

```
    } else {  
%>  
        <%= ph (promoName) %>  
<%  
    }  
%>
```

Managing User Types

There may be situations in which you need to modify an existing user type or you may need to create a new user type.

Adding a Role to a User Type

The definitions of user types are declared in the `UserTypeDefinition` elements in **Entitlements.xml**. For example, in Release 9.0, this is the definition of the `RegisteredUser` user type:

```
<UserTypeDefinition Name="RegisteredUser">  
  <Description>  
    Known direct commerce users with no partner affiliation.  
  </Description>  
  <Label>User</Label>  
  <MandatoryRoleSet>  
    <Role>Registered.User</Role>  
    <Role>Review.User</Role>  
  </MandatoryRoleSet>  
</UserTypeDefinition>
```

You can add a function or role to a user type simply by editing the **Entitlements.xml** file accordingly and by granting appropriate message groups to the new role. Note that you must restart the Visual Modeler for the new function to be available for assignment.

The `MandatoryRoleSet` element specifies the set of roles that cannot be removed from a user's entitlements. All users of this user type have these roles.

Creating a User Type

You can create the definition of a new user type simply by adding it to the **Entitlements.xml** file. Each user type is associated with partner types. The `PartnerTypeDefinition` elements of the **Entitlements.xml** configuration file determine which user types are available to which partners, so that only users of those types can be created for each partner. For example, consider the following `PartnerTypeDefinition` element:

```
<PartnerTypeDefinition Name="IndirectPartner">
  <Description>
    An IndirectCommercePartnerType partner has a
    relationship with the enterprise for
    the purpose of indirect commerce.
  </Description>
  <UserType>IndirectUser</UserType>
</PartnerTypeDefinition>
```

This says that when a user is created for a partner whose type is IndirectPartner, then only the IndirectUser user type may be selected.

Managing Access to Data Objects Using Access Policies

Access policies are designed to conform with the Java Authentication and Authorization Services (JAAS) model.

Access policies are particularly important for data objects that can be modified using the DsUpdate functionality. If your implementation of the Visual Modeler uses DsUpdate, then you must use access policies to manage the data objects updated using DsUpdate.

Access policies are applied to a data object by use of a ResourceClass element. For example:

```
<ResourceClass>com.comergent.bean.simple.PartnerBean</ResourceClass>
```

This element is declared within an AccessPolicy element. You can apply the same access policy to several different data objects by listing each of them as a ResourceClass element. Access policies are inherited by data objects that extend other data objects. For example, if an access policy declares the ShoppingCart as a ResourceClass, then the same access policy is also applied to the ChannelShoppingCart data object because it extends the ShoppingCart data object.

Release 7.1 and higher support *predictive access control*: when a data object controlled by an access policy is restored, the data services layer will attempt to amend the restoring query to reflect the access privileges defined in the access policy. If the data services layer does this, then it will not perform the access policy check on the returned result set.

Overview

An access policy controls access to a data object by specifying the conditions under which a user can perform an action on the data object, referred to as the *resource*. The following actions can be performed on data objects:

- Create
- Delete
- Restore
- Update

The conditions are specified as evaluating principals and expressions, and comparing them to the permitted values of the access policy. In general, principals are attributes of the user attempting an action on the data object, but they may be defined more generally. Expressions may be likened to SQL queries: they act as filters on the lists of data object being tested for access.

For example, suppose that you want to use an access policy to specify that only users that belong to a partner can update their partner profile. In this case, the action is Update and the condition that you want to define is that if you evaluate the partner key of the user object, then it must equal the partner key of the partner data object. In this example, the principal being evaluated is the partner key of the user.

Inheritance

When you define an access policy on a data object, it is inherited by all the data objects that extend it. Note that this means that if one data object extends another and you want to define different access policies for each, then you must declare distinct access policies for each of them.

AccessPolicy.xml Configuration File

You define access policies using an **AccessPolicy.xml** configuration file. Each AccessPolicy element declared in this file can be applied to one data object type: the data object is specified as the DataObject attribute of the AccessPolicy element.

Principal Qualifiers

Principal qualifiers are defined using the PrincipalQualifierDefinition element of the **AccessPolicy.xml** configuration file. Principal qualifiers are essentially Java classes that implement the PrincipalQualifier interface.

```
<PrincipalQualifierDefinition PrincipalType="UserType"  
    Class="com.comergent.dcm.entitlement.UserTypeQualifier"/>
```

Access Policies

Each `AccessPolicy` element specifies which `PrincipalQualifier` is to be used to evaluate the principal conditions by specifying the name of the `PrincipalQualifier` as the `PrincipalQualifier` attribute of the `AccessPolicy` element.

```
<AccessPolicy Name="UserPolicy" DataObject="UserContact"
  PrincipalQualifier="UserType">
```

Access Checkers

`AccessChecker` elements are used to define the individual checks that can be made to determine whether a user can access a data object. Each `AccessPolicy` element declares one or more `AccessChecker` elements. Each `AccessChecker` element specifies the permitted values of the principal to be compared, the action type to be checked, and any expressions to be evaluated to filter the data objects that can be acted on.

```
<AccessChecker>
  <Principal Select="Partner.DirectCommerceUser"/>
  <Principal Select="Partner.User"/>
  <ActionType Type="Restore"/>
  <BooleanExpression>
    <ComparativeExpression Operator="Equals">
      <Term>user.PartnerKey</Term>
      <Term>resource.PartnerKey</Term>
    </ComparativeExpression>
  </BooleanExpression>
</AccessChecker>
```

In this example of an `AccessChecker` element, the action type being checked for is “Restore”. The access policy is checked by comparing the user role of the user to see if one matches either “Partner.DirectCommerceUser” or “Partner.User”. The `Expression` element is evaluated to see if the `PartnerKey` field of the data object is equal to the partner key of the user, and this filter is applied to the data objects in question.

If there is more than one `BooleanExpression` element in an `AccessChecker` element, then use the `Operator` attribute to specify whether the boolean expressions should be combined using AND or OR. If no `Operator` attribute is specified, then OR is used by default.

Access Services

You can make use of access services to help retrieve information used to check access policies. Each `AccessServiceDefinition` element provides a name and a class. For example:

```
<AccessServiceDefinition Name="ownersPartnerKey" Type="resource" >
```

```
com.comergent.reference.dcm.entitlement.OwnersPartnerKeyService
<Description>
    Returns the partner key as a Long value for the owner of the
    resource if the resource extends C3PrimaryRWBean. Otherwise
    returns null.
</Description>
</AccessServiceDefinition>
```

This access service retrieves the owner key of the resource on which access is being checked.

Boolean Expressions

BooleanExpression elements are used to express the exact conditions under which access is granted to objects. They may be nested and they take an Operator attribute to specify how child elements should be combined.

As well as child BooleanExpression elements, you can also use ComparativeExpression elements, SetExpression elements, and Not elements to build up complex conditions:

- ComparativeExpression: use this element to compare the values of two fields.
- SetExpression: use this element to test membership of lists.
- Not: use this to wrap another expression so that the opposite boolean value is used.

Example

This fragment of the **AccessPolicy.xml** configuration file provides an example of how access policies are used. It determines access to order inquiry lists as described below.

```
<AccessPolicy Name="OrderInquiryListPolicy"
  PrincipalQualifier="UserRole">
  <Description>
    Controls access to Order Inquiry Lists.
  </Description>
  <ResourceClass>
    com.comergent.bean.simple.OrderInquiryListBean
  </ResourceClass>
  <ResourceClass>
    com.comergent.bean.simple.LightWeightOILBean
  </ResourceClass>
  <AccessChecker>
    <Description>
      Direct partner users with the listed roles can read an
```

```
    inquiry list if they own it or routed it to another user.
</Description>
<Principal>Anonymous.User</Principal>
<Principal>Registered.User</Principal>
<Principal>Partner.DirectBuyer</Principal>
<Principal>Partner.ProcurementUser</Principal>
<Principal>StorefrontCustomer*.TransferUser</Principal>
<Principal>StorefrontCustomer*.User</Principal>
<Principal>StorefrontCustomer*.AnonymousUser</Principal>
<Principal>StorefrontCustomer*.RegisteredUser</Principal>
<ActionType>Restore</ActionType>
<BooleanExpression Operator="Or" >
  <ComparativeExpression Operator="Equals">
    <Term>user.UserKey</Term>
    <Term>resource.OwnedBy</Term>
  </ComparativeExpression>
  <ComparativeExpression Operator="Equals">
    <Term>user.UserKey</Term>
    <Term>resource.RouteFromUserKey</Term>
  </ComparativeExpression>
  <BooleanExpression Operator="And">
    <SetExpression Operator="Intersection" >
      <Set>
        <Term>"Partner.BasicAdministrator"</Term>
      </Set>
      <Set>user.roleNameSet</Set>
    </SetExpression>
    <ComparativeExpression Operator="Equals">
      <Term>user.PartnerKey</Term>
      <Term>service.ownersPartnerKey</Term>
    </ComparativeExpression>
  </BooleanExpression>
</BooleanExpression>
</AccessChecker>
<AccessChecker>
  <Principal>Anonymous.User</Principal>
  <Principal>Registered.User</Principal>
  <Principal>Partner.DirectBuyer</Principal>
  <Principal>Partner.ProcurementUser</Principal>
  <Principal>StorefrontCustomer*.TransferUser</Principal>
  <Principal>StorefrontCustomer*.User</Principal>
  <Principal>StorefrontCustomer*.AnonymousUser</Principal>
  <Principal>StorefrontCustomer*.RegisteredUser</Principal>
  <ActionType>Update</ActionType>
  <ActionType>Create</ActionType>
  <ActionType>Delete</ActionType>
  <BooleanExpression Operator="Or">
    <ComparativeExpression Operator="Equals">
      <Term>user.UserKey</Term>
```

```

        <Term>resource.OwnedBy</Term>
    </ComparativeExpression>
    <BooleanExpression Operator="And">
        <SetExpression Operator="Intersection" >
            <Set>
                <Term>"Partner.BasicAdministrator"</Term>
            </Set>
            <Set>user.roleNameSet</Set>
        </SetExpression>
        <ComparativeExpression Operator="Equals">
            <Term>user.PartnerKey</Term>
            <Term>service.ownersPartnerKey</Term>
        </ComparativeExpression>
    </BooleanExpression>
</BooleanExpression>
</AccessChecker>
<AccessChecker>
    <Description>CustomerServiceRepresentatives can create,
    modify or delete any enterprise cart, but not storefront
    carts.
    </Description>
    <Principal>
        Enterprise.CustomerServiceRepresentative
    </Principal>
    <ActionType>Restore</ActionType>
    <ActionType>Update</ActionType>
    <ActionType>Create</ActionType>
    <ActionType>Delete</ActionType>
    <BooleanExpression Operator="And">
        <Not>
            <SetExpression Operator="Intersection">
                <Set><Term>service.ownersUserType</Term></Set>
                <Set>
                    <Term>"StorefrontCustomerUser"</Term>
                    <Term>"StorefrontCustomerAnonymousUser"</Term>
                    <Term>"StorefrontCustomerRegisteredUser"</Term>
                </Set>
            </SetExpression>
        </Not>
        <SetExpression Operator="Intersection">
            <Set>service.csrAssignedPartners</Set>
            <Set><Term>service.ownersRootPartnerKey</Term></Set>
        </SetExpression>
    </BooleanExpression>
</AccessChecker>
<AccessChecker>
    <Principal>*</Principal>
    <ActionType>Restore</ActionType>
    <ActionType>Create</ActionType>

```

```
<ActionType>Update</ActionType>
<ActionType>Delete</ActionType>
<BooleanExpression>
  <Never/>
</BooleanExpression>
</AccessChecker>
</AccessPolicy>
```

The way to read this fragment is as follows:

- This access policy determines access to `OrderInquiryListBeans` and `LightWeightOILBeans`
- Users who have one of the listed roles (`Anonymous.User`, `Registered.User`, and so on) may have `Restore` access (that is, have read access to the resource) if they satisfy one of the declared `BooleanExpressions`:
 - Either:
 - The user's key is equal to the owner key of the resource.
 - Or:
 - The user's key is equal to the routed from key of the resource.
 - Or:

Password Policies

Users authenticate themselves when they log in to the Visual Modeler using a username and a password. The Visual Modeler supports the ability to specify explicit password policies: these control the length and format of passwords as well as how passwords are created. Password policies are also used to determine what to do if a number of unsuccessful attempts are made to log in to the Visual Modeler. You can customize the password policies for your implementation. This section describes the password policies configuration file and how you can customize your password policies.

Configuration

The configuration of your password policies is managed in the **PasswordPolicies.xml** configuration file. This file declares each policy, the class used to implement the policy, and the parameters associated with the policy. For example, the following `PasswordPolicy` element specifies the policy governing the permitted lengths of passwords:

```
<PasswordPolicy Name="PasswordLengthPolicy" Type="Password"
  Enabled="true">
  <Description>
    This is the Policy to enforce password length
  </Description>
  <PolicyClass>
    com.comergent.reference.authentication.password.PasswordLength
  </PolicyClass>
  <ParamList>
    <Param Name="MinLength" Value = "5"/>
    <Param Name="MaxLength" Value = "15"/>
  </ParamList>
</PasswordPolicy>
```

The `PolicyClass` element declares the class used to test the policy. Each policy class must implement the `IPolicyClass` interface. The `ParamList` element provides the specific parameters used to test policies. In this example, the minimum length for passwords is set to be five characters and the maximum length is set to twelve characters.

Each policy has a type: this determines when the policy is exercised. The current types are:

- Initialization: policies of this type are used to determine how passwords are created.
- Password: policies of this type are exercised whenever a password is created or modified.
- Creation: policies of this type are used when passwords are created.
- Login: policies of this type are used to manage what to do when users log in.

You can customize the **PasswordPolicies.xml** configuration file using the SDK. By changing the parameter values you can change the behavior of the current out-of-the-box policies. You can also add your own policies: see "Creating a Custom Password Policy" on page 304.

The current password policies include:

- `UserCreatePolicy`: This policy specifies whether passwords can be created by users or whether they must be system generated.
- `PasswordLengthPolicy`: This policy specifies the permitted lengths of passwords. The value of `PasswordLengthPolicy`'s "MaxLength" parameter must be less than the value set in the `UserAuthenticator` field in the **DsDataElements.xml** file.

- DictionaryCheckPolicy: This policy checks for strings that cannot be used for passwords.
- PasswordReusePolicy: This policy controls the re-use of passwords by users.
- PasswordExpirationPolicy: This policy determines how frequently passwords must be changed by users.
- IncorrectLoginPolicy: This policy determines how many unsuccessful logins can be attempted before a user is locked out of the Visual Modeler.

Creating a Custom Password Policy

You can create your own password policy. Your policy class must implement the IPolicyClass interface. This interface declares the following method:

```
public PolicyCheckResult checkPolicy(IPasswordPolicy pp, HashMap hm);
```

The IPasswordPolicy interface is documented in the Javadoc provided with the Visual Modeler. The Hashmap object is used to pass in the parameters required for the policy.

The PolicyCheckResult class has a *hasError()* method. If this returns true, then you should handle the error condition appropriately.

Passing Login Data Through a URL

Most users of the Visual Modeler enter the system by pointing their browsers to the appropriate login page. However, sometimes, you may want to enable users to access a specific page such as the detail page of an order directly.

You can do this simply by constructing the URL as you would like it to be, for example:

```
http://server/Sterling/en/US/direct/matrix?cmd=OrderDisplay&-  
ShoppingCartKey=600501
```

When a user clicks on this link, their request is routed to the appropriate login page. In the login form, the request data from the original URL is automatically encoded as hidden form parameters, for example:

```
<input type="hidden" name="cmd" value="directLogin" >  
<input type="hidden" name="validate" value="true" >  
<input type="hidden" name="LoginData-messageType"  
    value="OrderDisplay"/>  
<input type="hidden" name="LoginData-ShoppingCartKey"
```



```
value="600501"/>
<input type="hidden" name="LoginData-entryPoint" value="direct"/>
```

When the login form is submitted, if the login is successful, then the parameters that begin with “LoginData-” are processed by the LoginController and added back to the request object with “LoginData-” removed from the parameter names. When the request is forwarded to the message type specified by the LoginData-messageType parameter, the original parameters are now available to the controller and JSP pages used to process the request.

Note that in general the message type in the original URL will be changed to the fallback redirect message type for the message type or message group to which the message type belongs. Consequently, take care that your intended message type is the fallback redirect message type for its message group.

You specify the fallback redirect message type along these lines:

```
<MessageGroup Name="AdvisorGroup">
  <FallbackRedirect>
    <Redirect EntryPoint="partnerMkt">PartnerHomePageDataDisplay"
  </Redirect>
  <Redirect EntryPoint="catalog">MatrixHomePageDisplay"</Redirect>
  <Redirect EntryPoint="advisor">MatrixHomePageDisplay"</Redirect>
  <Redirect EntryPoint="configurator">MatrixHomePageDisplay"
  </Redirect>
</FallbackRedirect>
  ...
  Message type definitions
  ...
</MessageGroup>
```

The way to read this XML extract is as follows: if an unauthenticated request is tries to execute any message type in the AdvisorGroup of message types, then redirect them to the PartnerHomePageDisplay message type if the endpoint of the request is “partnerMkt” or redirect them to the MatrixHomePageDisplay message type if the endpoint is one of the other three declared.

The LoginData-entrypoint is used to specify which entry point is used to access the system. It is retrieved from the original URL to ensure that the user is directed to the correct login page. A message group may have more than one default message type: they are differentiated by their Key attribute that specifies different entry points. For example:

```
<GroupDefault Key="partnerMkt" Value="IndirectWorkspaceDisplay" />
<GroupDefault Key="marketPlace" Value="DirectWorkspaceDisplay" />
```

This chapter describes the logging mechanism provided by the Visual Modeler. It enables application writers to log activity in the Visual Modeler. It uses the log4j API and **log4j.properties** configuration files to configure the logging behavior.

The logging capability also provides support for auditing changes to data objects. See "Auditing Changes to Data Objects" on page 309 for more information.

Overview

The log4j API provides a flexible and extensible logging framework to manage the logging behavior of the Visual Modeler. This section describes the use of the framework as you customize and extend the Visual Modeler.

Note that this framework replaces the previous framework used by the Visual Modeler: this used the Global class and its *logLevel()* methods. These are now deprecated.

To use the log4j API, you should create a Logger class in each class file along these lines:

```
private static final org.apache.log4j.Logger log =  
    org.apache.log4j.Logger.getLogger(NameOfClass.class);
```

When you want to make a log entry call:

```
log.info("This is a log entry");
```

The method you call depends on the logging level at which you want to record the message. You can use the following methods:

- *debug()*
- *error()*
- *fatal()*
- *info()*
- *warning()*

You can also use *log(priority, message)*, but in general the listed methods should be sufficient.

log4j.debug System Property

By setting the `log4j.debug` system property to true, you can echo out the current log settings. For example, include the following in the servlet container startup script:

```
-Dlog4j.debug=true
```

On startup, you should see logging output like this:

```
log4j: Trying to find [log4j.xml] using context classloader
sun.misc.Launcher$AppClassLoader@1362228.
log4j: Trying to find [log4j.xml] using sun.misc.Launcher$AppClass-
Loader@1362228 class loader.
log4j: Trying to find [log4j.xml] using ClassLoader.getSystemRe-
source().
log4j: Trying to find [log4j.properties] using context classloader
sun.misc.Launcher$AppClassLoader@1362228.
log4j: Using URL [jar:file:/home/hle/ws/32-cmgt-modules/modules.cryp-
tography-tool/target/cmgt-cryptography-tool-2.0.0-SNAPSHOT-app.jar!/
log4j.properties] for automatic log4j configuration.
log4j: Reading configuration from URL jar:file:/home/hle/ws/32-cmgt-
modules/modules.cryptography-tool/target/cmgt-cryptography-tool-
2.0.0-SNAPSHOT-app.jar!/log4j.properties
log4j: Parsing for [root] with value=[WARN, A1].
log4j: Level token is [WARN].
log4j: Category root set to WARN
log4j: Parsing appender named "A1".
log4j: Parsing layout options for "A1".
log4j: Setting property [conversionPattern] to [%-4r [%t] %-5p %c %x -
%m%n].
log4j: End of parsing for "A1".
log4j: Parsed "A1" options.
log4j: Finished configuring.
```

Auditing Changes to Data Objects

In many implementations, you may want to provide an audit trail that tracks changes made to data in the Visual Modeler. In Release 7.0.1 and higher, you can do this by logging any changes made to data objects. If you set the logging level to INFO or higher in any DataBean class, then whenever *persist()* is invoked on an instance of this class, a log message is written out to the Logger for the class. For example: the following is a sample line that is written out when a change is made to a partner:

```
2006.01.18 13:41:05:546 Env/http-8080-Processor23:INFO:PartnerBean
Updating: com.comergent.bean.simple.PartnerBean KeyFields - Partner-
Key: 301 Changes -PartnerKey -> old: 301 new: 301PartnerName -> old:
Scalar2 new: Scalar2 LegalName -> old: null new: null ParentCompany -
> old: null new: nullStatus -> old: A new: A DunBradID -> old: null
new: nullBusinessID -> old: Scalar2-001 new: Scalar2-
001PartnerTypeCode -> old: 10 new: 10PartnerLevelCode -> old: 20 new:
20XMLMessageVersion -> old: dXML 4.0 new: dXML 4.0BusinessTransaction
-> old: SELL new: SELL NetWorth -> old: null new: null NumEmployees -
> old: null new: null PotRevCurrFy -> old: null new: null PotRevNextFy
-> old: null new: null ReferenceUseFlag -> old: null new: null Coterm-
DayMonth -> old: null new: nullURL -> old: http://www.scalar.com new:
http://www.scalar2.com LogoURL -> old: null new: null DistiAccess ->
old: null new: null YearEstd -> old: null new: null AnalysisFy -> old:
null new: null FyEndMonthCode -> old: null new: null AccountManagerKey
-> old: null new: null MessageURL -> old: null new: null EmailAddress
-> old: null new: nullCommerceCategory -> old: 2 new: 2 PartnerRefNum
-> old: null new: null ParentKey -> old: null new: null RootPartnerKey
-> old: null new: null ParentCode -> old: null new: null CustomField1
-> old: null new: null CustomField2 -> old: null new: null
CustomField3 -> old: null new: null CustomField4 -> old: null new:
null CustomField5 -> old: null new: null PartnerCom -> old: null new:
null Storefront -> old: null new: null URLName -> old: null new: null
ContentType -> old: null new: nullPartnerStatusCode -> old: 10 new:
10OrganizationType -> old: DirectPartner new: DirectPartner Inherited-
PartnerStatusCode -> old: null new: nullCreditLimit -> old: 0.0000
new: 0.00AvailableCredit -> old: 0.0000 new: 0.0000CreditCurrencyCode
-> old: 23 new: 23 MaxAssignableReps -> old: null new: null Remote-
Prices -> old: null new: null RemotePriceExpiryInterval -> old: null
new: nullCoopPercentage -> old: 0.000000 new: 0.000CoopAccountMax ->
old: 0.000000 new: 0.00 PartnerID -> old: null new: nullOwnedBy ->
old: 0 new: 0AccessKey -> old: 5601 new: 5601UpdateDate -> old: 2006-
01-18 13:39:33.0 new: 2006-01-18 13:41:05.484UpdatedBy -> old: 0 new:
0CreateDate -> old: 2006-01-04 13:19:38.0 new: 2006-01-04
13:19:38.0CreatedBy -> old: 0 new: 0
```

You can dynamically change the logging level for any class in the Visual Modeler through the administration UI. However, if you do this, then the change to the logging level is not persistent, and will be lost if the servlet container is restarted. In addition, the logging is written out to the standard Appender which may not be secure.

You should specify any audit logging by customizing the **log4j.properties** configuration file: this ensures that the auditing will continue to be done even if the servlet container is restarted, and you can specify a custom Appender to process the audit information. For example, you can specify that the Appender posts the logging message to a remote Web server which can be secured independently of the Visual Modeler.

As an example, the following entries in the **log4j.properties** configuration file ensure that all changes to the UserContact data object are audited:

```
log4j.logger.com.comergent.bean.simple.UserContactBean=info
log4j.appender.com.comergent.bean.simple.UserContactBean=com.comer-
gent.logging.ComergentRollingFileAppender
log4j.appender.com.comergent.bean.simple.UserContactBean.layout =
org.apache.log4j.PatternLayout
```

If you want to specify that a remote log server can connect as a client in order to save audit information from the Visual Modeler, then you could specify:

```
log4j.appender.com.comergent.bean.simple.UserContact-
Bean=org.apache.log4j.net.SocketHubAppender
log4j.appender.com.comergent.bean.simple.UserContactBean.port=4321
```

Release 9.0 of the Visual Modeler has undergone the following architectural changes designed to make implementations easier to customize and upgrade:

- "Modules" on page 312
- "Generated Interfaces" on page 315

These changes are related in that the interfaces are organized by modules and that changes to interfaces may be contained to changes within individual modules.

Overview

The motivation to make these architectural changes are to ensure that customizations are more contained and can be better preserved during upgrade from one release of the Visual Modeler to another.

By providing a means of delivering functionality in modules and by requiring that modules make calls to other modules only through their external interfaces, the following benefits are achieved:

- It is easier to compartmentalize the functionality of applications.
- It is easier to understand and manage the dependencies between parts of the Visual Modeler.

- It is easier to contain the customizations to single modules and understand what effect changes made in a module have on the whole system.
- Modules can be more easily upgraded independently of each other, minimizing the effect that an upgrade may have.
- Upgrades to modules that have not been customized will not effect customizations made in other modules.
- New functionality can be delivered in the form of a module that may be dropped into an existing deployment of the Visual Modeler.

Modules

The Visual Modeler is developed as a set of interdependent modules that conform to a common organizational structure. In general, each module corresponds to a functional component of the Visual Modeler such as an application or a component of the Visual Modeler platform. Some modules may support both a Java API and a user interface whereas other may just support a Java API provided to other modules. Some modules provide a set of “helper” classes, JSP pages, and other files such as Javascript files and images which are used by a number of other modules.

In general, each module has the following structure:

- Java classes: organized into three trees. At build time, the directories for all of the modules are assembled in to a single tree under the `com.comergent` package.
 - external API interfaces: used by other modules to access functionality provided by the module. In general, when one module makes a call to another module’s class, it must do so through the other module’s external API. This is the `com.comergent.api` package for the module.
 - implementation classes: the implementation of the external API interfaces. When another module makes a call to the module’s external API, then the actual classes used are the implementing classes of the module’s interface. The implementation packages may include internal classes: used by the implementation classes, but not exposed to outside world and not part of the supported Javadoc. This is the `com.comergent.apps` or `com.comergent.appservices` package for the module.
 - reference components: Controller classes and JSP pages always comprise part of the reference implementation and their source is provided with the

Visual Modeler. Resource bundles are also provided as part of the reference. This is the `com.comergent.reference` package for the module.

- JSP pages: possibly organized into directories depending on the organization of the module. They should always access other modules' classes through the external APIs exposed by the other modules. This ensures that JSP pages can be re-used from release to release provided that the external APIs are supported.
- Resource bundles, Javascript, and static files (such as images and HTML fragments).
- Configuration files specific to the module such as **MessageTypes.xml** files and business rules.

Module Interfaces

Each module must provide an external interface so that all calls to Java classes and interfaces within the module are invoked through the interface. This external interface provides a comprehensive set of Javadoc pages so that writers of other modules can use the external interface reliably and easily.

The external interface for each module will typically be a combination of handcrafted interfaces and automatically-generated interfaces. Most modules provide handcrafted interfaces for presentation beans that enable presentation beans to manipulate data beyond the simple accessor methods of the generated data bean interfaces. The presentation beans usually wrap a data bean and implement the same interfaces, but in addition they implement helper methods and some business logic.

The external interfaces are organized under the following main packages:

- `com.comergent.api`: this package has all the module external APIs. These are organized into:
 - `apps`: these are the application hand-crafted APIs. Typically, these are presentation bean interfaces, utility interfaces, and factory classes.
 - `appservices`: these are the packages provided by the service modules used by other applications.
 - `dcm`: these are the external APIs offered by the Visual Modeler platform.
- `com.comergent.bean.simple`: this package has all the automatically-generated bean interfaces and the data bean classes themselves.

The generated interfaces are provided for each of the data objects declared in the XML schema files. These are organized to provide appropriate levels of access to the data fields of the underlying data beans. This helps to ensure that there is a clearer separation between presentation and business logic in the Visual Modeler. See "Generated Interfaces" on page 315 for more information about the generated interfaces.

Invoking Interfaces

You can invoke an interface from a Java class by casting any object or child interface to the interface and then invoke any method that the interface declares. In the Visual Modeler, use one of the following techniques to do this:

- "Using the Object Manager" on page 314
- "Using Factory Classes" on page 315

Each module uses one or other of these techniques, but not both. As you work on an existing module or create a new one, be consistent in how you invoke the interfaces. It will make it easier for your colleagues to work on the same module.

In general, you should always try to work with interfaces provided by the `com.comergent.api` packages: these are the interfaces that the modules will support from one release to the next, even though the underlying implementations of the interfaces may change.

Using the Object Manager

You can use the `ObjectManager` class to return an appropriate interface as follows. Suppose that you want to retrieve the `IAccProduct` interface to set the data fields of a product. Then make a call along these lines:

```
IAccProduct temp_IAccProduct =
    (com.comergent.bean.simple.IAccProduct)
    com.comergent.dcm.util.OMWrapper.getObject(
        "com.comergent.bean.simple.IAccProduct");
```

Provided that there is an entry in the **ObjectMap.xml** file that specifies the object to be returned and provided that the object implements the `IAccProduct` interface, then this call will succeed and methods on the interface can be invoked. For example, if the **ObjectMap.xml** file contains:

```
<Object ID="com.comergent.bean.simple.IAccProduct">
    <ClassName>com.comergent.bean.simple.ProductBean</ClassName>
```

Then, the `ObjectManager` returns a `com.comergent.bean.simple.ProductBean` object and this can be cast to the `IAccProduct` interface because the

com.comergent.bean.simple.ProductBean class implements the com.comergent.bean.simple.IAccProduct interface.

Using Factory Classes

Calls to an interface can be provided by Factory classes that return an instance of the interface. For example, the package com.comergent.api.apps.commerce provides a public interface IIquiryListFactory. If another module needs an instance of this Factory interface, then it calls the CommerceAPI class's *getFactory(int i)* method. The int parameter determines what sort of Factory class is returned. In turn, the calling module can now invoke methods on the IIquiryListFactory to return inquiry list interfaces of the appropriate type. For example, *getInquiryList(Long listKey, boolean bFillPrices)* returns an object that implements the IIquiryList interface.

Generated Interfaces

When you need to access data on a particular data object, you must use the generated interfaces that each data object provides. These generated interfaces are created and compiled when the SDK generateBean target is run as part of the deployment of your Visual Modeler.

For each data object declared as a DataObject within the **DsRecipes.xml** file, and for any parent, reference, or child data objects, the following classes and interfaces are generated and compiled in the com.comergent.bean.simple package:

- *<Name>.java*: this is the data bean class. It implements the interfaces listed here. In addition, if the data object extends another data object, then the bean extends the *<Parent>.java* bean.
- *IAcc<Name>.java*: this interface extends the *IRd<Name>.java* by providing the write (set) accessor methods on all of the data fields of the data object. In addition, if the data object extends another data object, then the IAcc interface extends the *IAcc<Parent>.java* interface.
- *IData<Name>.java*: this interface extends the *IAcc<Name>.java* by providing *restore()* and *persist()* methods on the data object. In addition, if the data object extends another data object, then the IData interface extends the *IData<Parent>.java* interface.
- *IRd<Name>.java*: this interface provides the read-only (get) accessor methods to the data fields of the data object. In addition, if the data object

extends another data object, then the IRd interface extends the IRd<Parent>.java interface.

- In addition, list beans also implement the IData<Name>List.java interface. Each list interface extends the IDataList.java interface as well as the list interface of any parent object.

In general, you should use the IRd interface for any objects to be passed to JSP pages so that the objects are effectively read-only. Only use objects that implement the IData interface when you know that you need to either restore or persist the data object.

Example of a Generated Interface

Consider the ACL data object: the ACL.xml file reads:

```
<?xml version="1.0"?>
<DataObject Name="ACL" Extends="C3PrimaryRW"
  ExternalName="CMGT_ACLS"
  Access="RWID" Ordinality="1"
  ObjectType="JDBC" Version="5.0">
  <KeyFields>
  <KeyField Name="AccessKey" ExternalName="ACL_KEY"
    KeyGenerator="ACLKey"/>
  </KeyFields>
  <DataFieldList>
    <DataField Name="AccessKey"
      Writable="n" Mandatory="n"
      ExternalFieldName="ACL_KEY"/>
    <DataField Name="ACLName"
      Writable="y" Mandatory="n"
      ExternalFieldName="NAME"/>
  </DataFieldList>
  <ChildDataObject Name="Access" />
</DataObject>
```

Consequently, the IRdACL.java class declares:

```
public interface IRdACL extends IRdC3PrimaryRW
```

and exposes the methods:

- public Long getAccessKey();
- public String getACLName();

The IAccACL.java class declares:

```
public interface IAccACL extends IAccC3PrimaryRW, IRdACL
```

and exposes the methods:

- `public void setACLName(String value) throws ICCEException;`
- `public void addAccess(AccessBean bean) throws ICCEException;`

The `IDataACL.java` class declares:

```
public interface IDataACL extends IAccACL, IDataC3PrimaryRW, IData
```

In general, this interface may declare no additional methods beyond those declared in the `IData` interface because all the standard methods to read and write data from external data sources are declared in this interface.

This chapter and the next two chapters present a description of how to implement business logic classes (BLCs) at an implementation of the Visual Modeler. Before reading this chapter, you must have a working understanding of the basic architecture of the Visual Modeler and of Java.

<p>Note: The use of BLCs is deprecated. In general, new applications should use bizlets, controllers, and BizAPIs to implement their business logic.</p>

Key Concepts

To understand fully how the Visual Modeler works as an application, you must understand its architecture.

An installation of Visual Modeler processes requests as they are received from users' browsers, and messages from other Visual Modelers and from external systems. You must configure the Visual Modeler to process each type of request and message.

The core of the Visual Modeler is the Sterling Commerce Manager. This powerful and flexible server is designed to seamlessly integrate a network of channel partners and the external systems that make up the e-commerce environment of each partner.

Each Visual Modeler server in the network of sales partners works both as a server in relation to inbound requests from browsers and as a client as it retrieves information from other Visual Modeler servers and external systems.

To customize the Visual Modeler in your environment, you need to consider how the system retrieves data from your external systems. In general, you can use the schema and Service classes to retrieve data from a local database source or from another Visual Modeler server by exchanging messages. However, you have to produce custom BLCs to retrieve information from an external system other than these.

Application Logic Classes

Application logic classes are implemented as bizAPI, business logic , or controller classes.

- bizAPI classes are used to manage the business logic of business objects. Conceptually, each bizAPI class corresponds to a business object and its methods correspond to the actions that can be performed on the business object. For example, the OrderInquiryList bizAPI class provides the following methods: *duplicate()*, *copyLineItem()*, and *changeOwner()* which correspond to actions that can be performed on a product inquiry list. It implements the `com.comergent.api.apps.orderMgmt.oil.IOrderInquiryList` interface.

The bizAPI classes are defined in the `com.comergent.apps.<application>.bizAPI` packages. Typically, they implement an interface declared in the corresponding `com.comergent.api.apps.<application>` package.

For example, the Order bizAPI class is in the `com.comergent.apps.orderMgmt.orders.bizAPI` package. It extends the OrderInquiryList class and implements the `com.comergent.api.apps.orderMgmt.orders.IOrder` interface.

- Each BLC is a subclass of the BLC abstract class. This class implements the ApplicationObject interface. BLCs perform the business logic of your implementation of the Visual Modeler. Each BLC contains a table of business objects such as session, user, and shopping cart for example. In executing the *service()* method of a BLC, it invokes the *persist()* and *restore()* methods of these business objects.

Note: In general, the use of BLC classes is deprecated. You should use either controllers or bizAPI classes to manage your business logic.

- Some Visual Modeler use controller classes to perform business logic. These classes are to be found in the `com.comergent.reference.apps.<application>.controller` packages for each application.

The Visual Modeler comes with a number of standard bizAPI classes, BLCs, controllers, and JSP pages. However, you may need to create new logic classes or modify the existing classes.

Business Objects

See CHAPTER 18, "Introducing Data Beans and Business Objects" for more information.

XML Schema

You should manage data access using the the schema and Service classes.

Naming Service

To retrieve parameters at runtime, the Visual Modeler provides a naming service to access either a flat file or a database to recover parameters.

Application logic classes can invoke a naming service by calling the static class NamingManager methods `getInstance()` and `getInstance(int i)`. Both these methods return an object that implements the NamingService interface.

- If no integer argument is provided, then an object of default type is created, either a NamingServiceProperties object or a NamingServiceDatabase object.
- If the integer argument is the constant NamingManager.DATABASE, then a NamingServiceDatabase object is created.
- If the integer argument is the constant NamingManager.PROPERTIES, then a NamingServiceProperties object is created.
- If the integer argument is not one of these two, then an object of default type is created.

In all cases, the Visual Modeler accesses the **Comergent.xml** file to determine exactly how the NamingService object should be created:

- If a NamingServiceDatabase object is to be created, then the NamingManager.database entries are used to specify the connection to the database.

- If a NamingServiceProperties object is to be created, then the NamingManager.properties entry is used to determine which properties file holds the parameter values.

Once the NamingService object is created, you use the methods listed below to retrieve the parameters as a NamingResult class:

- public NamingResult get(int key)
- public NamingResult get(Long key)
- public NamingResult get(String key)

The key parameter provides a means of retrieving only those parameters whose name begins with the key string.

The NamingResult class provides the *get(String parameter)* method to return the value of the parameter.

NamingService Example

For example the following code fragment recovers the value of the message URL parameter for a distributor referred to by its partner key.

```
NamingService namingService = NamingManager.getInstance();
NamingResult namingResult = namingService.get(partnerKey);
String url = namingResult.get(NamingResult.MESSAGE_URL);
```

Note that by default, the type of NamingService created is a NamingServiceDatabase object because in **Comergent.xml** the NamingManager defaultType element is set to "database".

You can use the Visual Modeler Software Development Kit (SDK) to install and customize your implementation of the Visual Modeler. The HTML documentation provided with each version of the SDK provides an overview of how the SDK works and how to use it to manage projects. This chapter describes the basic structure of a customization project. Follow the guidelines here to organize your project so that it follows the customizations guidelines.

Project Organization

Each project built using the SDK is created on top of a release of the Visual Modeler. When you create the project using the `newproject` target, the SDK creates a set of project files that are suitable for that release. All of the customizations that you make in the project are made by adding files to the project. Files can be added to the project in these ways:

- Use the `customize` target to copy a file from the release into the project. When you use the `customize` target, the file is automatically copied into the appropriate sub-directory of the project.
- Create the file manually in the appropriate sub-directory of the project.

See "Project File and Directory Locations" on page 324 for information about where files must be located.

Project File and Directory Locations

In this section, we assume that you created a project called *project*, and that you have a project directory called *sdk_home/projects/project/*. Ensure that each of the project files is in the appropriate location under the project directory as follows:

- Java source files: these must be placed under the *project/src/* directory, and follow the package organization for the Visual Modeler.
- JSP pages: these are organized by module and locale under the *project/WEB-INF/web/* directory.
- Schema files: these comprise the data object files and the supporting data services files. All your customizations should be maintained under the *project/WEB-INF/schema/custom/* directory. Make sure that the `schemaRepositoryExtn` element is set to “WEB-INF/schema/custom”.

Java Source Files

In the *project/src/* directory, follow these guidelines to organize your customizations to the Visual Modeler:

- Use the `com/comergent/api/` packages to add your extensions to the Visual Modeler API. In general, you should create new classes that extend the existing API: do not overwrite the release API because that can affect any upgrade.
- Use the `com/comergent/apps/` and `com/comergent/appservices/` packages to add implementation classes: these may be entirely new classes or new classes that extend existing implementation classes.
- Use the `com/comergent/reference/` packages for controller classes. You can customize existing controller classes or create new controller classes.

JSP Pages

In the *project/WEB-INF/web/* directory, follow these guidelines to organize your customizations to the Visual Modeler:

- Where appropriate, use the existing organization of JSP pages to add new JSP pages or to customize existing ones.
- If you are adding a new functionality module, then create a new directory under the appropriate locale(s) for the module, and follow the same naming convention as you do for Java classes created for the module.

Schema Files

In the *project/WEB-INF/schema/custom/* directory, follow these guidelines to organize your customizations to the Visual Modeler:

- To add new data objects:
 - Put the XML definition of the data object in *project/WEB-INF/schema/custom/*. For example, create the file *project/WEB-INF/schema/custom/CustComment.xml*
 - Modify *project/WEB-INF/schema/custom/DsBusinessObjects.xml* by adding the new business object. For example:

```
<?xml version="1.0"?>
<Schema Name="project" Description="project Custom Schema"
  Version="6.0">
  <BusinessObject Name="CustComment" Version="6.0"
    Description="CustComment BusinessObject"/>
</Schema>
```

- Modify *project/WEB-INF/schema/custom/DsDataElements.xml* by adding the new data elements for the header and list data objects, together with any new fields declared by the data object. For example:

```
<?xml version="1.0"?>
<Schema Name="project" Description="project Custom Schema"
  Version="6.0">
  <DataElement Name="CustComment" Description="Customer Comment
data object"
  DataType="HEADER"/>
  <DataElement Name="CustCommentList" Description="Customer Com-
ment list data
  object" DataType="HEADER"/>
  <DataElement Name="CustCommentKey" Description="Customer Com-
ment Key"
  DataType="LONG" MaxLength="20"/>
</Schema>
```

- Modify *project/WEB-INF/schema/custom/DsRecipes.xml* by adding a recipe element. For example:

```
<Schema Name="project" Description="project Custom Schema"
  Version="6.0">
  <Recipe Name="CustComment" BusinessObject="CustComment"
  Description="Default Approvals List Recipe" Version="6.0">
  <DataObjectList>
  <DataObject Name="CustComment" Access="RWID"
  DataSourceName="ENTERPRISE" Ordinality="n"
  Version="6.0"/>
  </DataObjectList>
  </Recipe>
</Schema>
```

```
        </DataObjectList>
    </Recipe>
</Schema>
```

- Modify the appropriate key generator file, for example ***project/WEB-INF/schema/custom/OracleKeyGenerators.xml***, by adding any new keys required:

```
<?xml version="1.0"?>
<Schema Description="project Custom Schema" Name="project"
    Version="6.0">
    <KeyGenerator Name="CustCommentKey" KeyProcedureName="CUSTCOM-
MENTKEY"
        GeneratorType="PROCEDURE" />
</Schema>
```

This chapter describes localization issues to consider while you work on Visual Modeler applications.

Overview

The Visual Modeler has built-in support for:

- multiple currencies
- multiple languages
- number and date formats
- character sets

You can also manage other aspects of localization for specific markets such as:

- local laws and regulations
- currency processing
- shipping and export information
- taxes

Support for internationalization is managed using locales. Each locale identifies a language and country. By identifying which locale is to be used when displaying

information to a user, you ensure that the user sees information that is both specific to their locale and presented as they would expect to see it.

When users log in to the Visual Modeler, a locale is assigned to the session: this is the preferred locale specified in the user's profile. Users can change their preferred locale in their user profile and the change takes effect the next time they log in. User administrators can change a user's preferred locale just as they can change other aspects of a user's profile.

The system default locale is specified in the **Internationalization.xml** configuration file using the `defaultSystemLocale` element. You can specify a default locale for each language: see "Failover Behavior" on page 332 for more information.

The Visual Modeler offers full Unicode support for data entry and display.

A significant amount of localization can be performed using Java ResourceBundles: see "Resource Bundles and Formats" on page 339 for more details.

Supporting Locales

If you plan to implement the Visual Modeler to provide support for more than the `en_US` locale, then you must produce pages to reflect local language and other locale-specific information (such as office locations).

Presentation and Session Locales

When a user logs in to the Visual Modeler, the authentication process retrieves their preferred locale: this is defined in their user profile. The system makes use of two logically distinct locales:

- session locale: this determines what data is retrieved for data objects from the Knowledgebase.
- presentation locale: this determines what JSP pages and resource bundles are used to render HTML pages to the user.

In general, the set of locales that you support as presentation locales must be a subset of the possible session locales. For example, you choose to maintain `fr_CA`, `fr_CH`, and `fr_FR` as session locales, but only support `fr_FR` and `fr_CA` as presentation locales.

When a user first logs in, the system calculates a presentation locale for the user session as follows:

1. If the user's preferred locale is declared in the Visual Modeler **web.xml** file, then set this to be the presentation locale.
2. If not, then consult the **Internationalization.xml** file: if the `useCountryDefaulting` element is set to "true", then identify the default country locale for the language of the user's preferred locale. Check to see if the default country locale is declared in the **web.xml** file. If it is, then set the presentation locale to this.
3. If either the `useCountryDefaulting` element is set to "false" or the default country locale is not present in the **web.xml** file, and if the `useGeneralDefaulting` element is set to "true", then set the user's presentation locale to the default system locale specified by the `defaultSystemLocale` element.
4. If the Defaulting elements are set to false or if no locale is identified that is declared in the **web.xml** file, then the presentation locale is set to the session locale.

This presentation locale is used to determine the user's experience as they navigate through the Visual Modeler by controlling which JSP pages and properties files are used to render the Web pages that they see. At the same time, the preferred locale is also set as their *session locale*: this session locale is used to determine what data is retrieved from the database when localized data objects are displayed to the user.

<p>Attention: You must make sure that every locale you create in the database either has a corresponding set of entries in the web.xml file or that its default country locale has entries in the web.xml file and you enable country defaulting. If you do not do this, then some users may not be able to access the system.</p>

JSP Pages and Properties Files

1. For each JSP page, there must be at least one JSP page located in the appropriate module sub-directory under the system default locale directory. When you first install the Visual Modeler, the default system locale is set to `en_US`. Consequently a full set of JSP pages is provided under ***debs_home/SterlingWEB-INF/web/en/US/***. If you change the default system locale, then take care to fully populate the corresponding directories for the new locale.
2. All visible text on each page is declared using the Comergent tag library text tag or the corresponding `cmgtText()` method. For example:

```
<cmgt:text id='cmgt_channelMgmt/channelCartDisplay/  
ChannelCartDisplayData_7' bundle='channelMgmt.channelCartDis-
```

```
play.ChannelCartDisplayDataResources'>Build Product List
</cmgt:text>
```

or

```
String title =
    cmgtText("cmgt_commerce/search/AdvancedSearchBody_2",
        "Inquiry Lists Search");
```

The bundle attribute must correspond to a file in the `com.comergent.reference.jsp` package of the class tree. For the example above, there must be a file called

ChannelCartDisplayDataResource.properties in the *debs_home*/**Sterling/WEB-INF/classes/com/comergent/reference/jsp/channelMgmt/channelCartDisplay/** directory. The id attribute must be unique within the properties file. For the example above, there should be a line of the form:

```
cmgt_channelMgmt/channelCartDisplay/
ChannelCartDisplayData_7=Build Product List
```

3. For each additional supported locale (say, *la_CO*), you must copy the following directories from **debs_home/Sterling/WEB-INF/web/en/US/** to **debs_home/Sterling/WEB-INF/web/la/CO/**:
 - **cic/**
 - **common/**
 - **home/**
4. For each additional supported locale (say, *la_CO*) and for each JSP page, you must:
 - a. Either create a new JSP page for the locale and put it in the corresponding directory location in the Web application: a directory under **debs_home/Sterling/WEB-INF/web/la/CO/**. If the same page can be used for more than one locale in the same language (for example, *fr_FR* and *fr_CA*), then make sure that you put it in the default locale for the language. See "Failover Behavior" on page 332 for more information about default locales for languages.

- b. Or prepare a properties file that contains the appropriate text for each id. These properties files are organized so that there is one for each JSP page and JSP fragment.

Note:	HTML and Javascript characters such as "<", ">", "'", and so on must not be included in the property values. These characters must be escaped using the HTML or Javascript mechanisms to escape characters. For example: use "<" for "<" in HTML and "\" for "'" in Javascript.
--------------	--

The properties files must conform to the Java standard for properties files used by resource bundles. Specifically, they should follow this naming convention: **<Name of JSP page>Resources_la_CO.properties**. They must be text files in which each line should take this form:

cmgt_module/package/JSPname_n=Display text for this locale

For example:

```
cmgt_channelMgmt/channelCartDisplay/  
ChannelCartDisplayData_7=Build Product List
```

The properties files are all located in the **debs_home/Sterling/WEB-INF/classes/com/comergent/reference/jsp/** directory and are organized by module within this directory in the same way that the module JSP pages are organized within a module. Note that if you want to change the location of these resource bundles, then you must customize the text tag to retrieve the resource bundles from their new location.

If you add text to a JSP page, then take care to update the corresponding locale JSP pages or properties files, either with amended text for an existing tag id or by adding a new id.

Notes

Note the following:

- The length of the translated text can be significantly different: this can affect the layout of a Web page.
- Drop-down lists and Javascript functions can have text that if translated will affect the logic of the Visual Modeler. See "Javascript" on page 337 and "JSP Pages" on page 337.
- Local regulations can effect the display of information (such as the display of prices in both Euros and a local currency).
- Take particular care if the logical flow of pages must change to reflect local practice (such as the display of an export notice or tax information).

Debugging

You can use the `debugJSPResourceBundle` element of the **Internationalization.xml** configuration file to help you identify missing strings. Set this element to "true" and if a string is missing from the referenced resource bundle, then an error message is displayed on the browser page. You should set this value to "false" in your production systems.

Failover Behavior

This section describes what happens when resources (JSP pages or properties) are not defined for the user's current presentation locale. Note that the failover behaviors are slightly different for JSP pages and resource bundles:

- JSP pages can fail over from a specific locale to the default country for the language locale and then to the system default locale. For example: `fr_CA` to `fr_FR` to `en_US`.
- Resource bundles fail over according to the Java specification:
`*_fr_CA.properties` to `*_fr.properties` to `*.properties`.

Two properties in the **Internationalization.xml** configuration file are used to manage failover behavior for JSP pages:

- `useCountryDefaulting`: if this is set to true, then default to the country specified in the appropriate language element if no resource is present for the presentation locale.
- `useGeneralDefaulting`: if this is set to true, then default to the system locale if no resource is available for the presentation locale.

Resource Bundles

You do not need to translate all text strings into each locale. If a text string is not present for a given id in a resource bundle properties file, then the standard Java failover process is followed. For example, if the

ChannelCartDisplayDataResource_fr_CA.properties does not define the `cmgt_channelMgmt/channelCartDisplay/ChannelCartDisplayData_7` string, then, if it exists the **ChannelCartDisplayDataResource_fr.properties** file is consulted. If this file does not exist or does not have an entry for this id, then the **ChannelCartDisplayDataResource.properties** file is consulted.

JSP Pages

Not all the JSP pages need be available for all supported locales. For example, you may choose to use `en_US` pages for all but a small number of pages viewed by

en_CA users. This section describes what happens when a message type is processed:

The request is forwarded to the JSP page specified by the JSPMapping element of the message type in the appropriate **MessageTypes.xml**.

1. If the JSP page does exist for the current locale, then this page is used to generate the Web page.
2. If the JSP page does not exist for the current locale, then the failover mechanism identifies the default locale for the language of the current locale. This is declared as the defaultCountry element for the language in the **Internationalization.xml** configuration file.
3. If a JSP page exists in the language-default locale, then this page is used to generate the Web page. For example, the following element in **Internationalization.xml** specifies that US is the default country for the en language locales, and so if a JSP page is not present for the en_CA locale, then the corresponding en_US JSP page is used.

```
<en visible="false">
  <defaultCountry ...>US</defaultCountry>
</en>
```

4. If there does not exist a JSP page for the default country, then the failover mechanism identifies the default system locale. This is declared as the value of the defaultSystemLocale element of the **Internationalization.xml** file. If a JSP page exists in the system default locale, then this page is used to generate the Web page.
5. Finally, if no JSP page exists in the default system locale, then an exception is thrown and an error page is displayed.

Methods to Retrieve Locales

Most of the time you should be able to make use of the Visual Modeler's built-in support to display appropriate content to users for their locales. If you do need to manually access locales, then the `ComergentI18N` class can be used. It provides the following methods:

- `getDefaultLocale()`: returns the system default locale.
- `getComergentLocale(boolean b)`: if `b` is true, then returns the user's presentation locale; otherwise returns the user's session locale.
- `findPresentationLocale(Locale sessionLocale)`: used to calculate what presentation locale should be used for a given session locale.

Using Properties Files in Code

You can make use of properties files in your Java code too. For example, to retrieve the locale-specific String that corresponds to the String keyString defined in the **com.comergent.reference.jsp.AdvisorBodyResources.properties** file, use:

```
String temp_NamedPopertiesFile =
    "com.comergent.reference.jsp.AdvisorBodyResources.properties";
ResourceBundle temp_ResourceBundle =
    com.comergent.dcm.util.ComergentI18N.-
        getBundle(temp_NamedPopertiesFile);
String temp_LocalisedString =
    temp_ResourceBundle.getString("keyString");
```

This uses the current locale of the user as stored in the user's session. If you want to force the use of a different locale, then use:

```
Locale specific_Locale = new Locale("fr", "CA");
String temp_NamedPopertiesFile =
    "com.comergent.reference.jsp.AdvisorBodyResources.properties";
ResourceBundle temp_ResourceBundle =
    com.comergent.dcm.util.ComergentI18N.-
        getBundle(temp_NamedPopertiesFile, specific_Locale);
String temp_LocalisedString =
    temp_ResourceBundle.getString("keyString");
```

Data for Internationalization

If you expect enterprise users and end-users to be entering data in multi-byte characters, then you need to consider the length of data fields and their corresponding database table columns. In our experience, data entered into the Visual Modeler that uses multi-byte characters can be up to three times as long in the database as the strings used for the en_US locale. Consequently, you should review the length of fields in which you expect data to be entered that will take multi-byte characters: notably name and description fields.

If you want to change the length of fields, then bear in mind that you have to both change them in the **DsDataElements.xml** configuration file and make the corresponding change to the SQL script that is used to generate the Knowledgebase schema.

For example, to make the Description field of the Product data object suitably long for multi-byte characters, you must do the following:

1. Identify the data field that is used to hold product descriptions. Because the Product data object is a localizable data object (Localized="y"), this is the

Description field of the ProductLocale data object. Its corresponding database table and column is CMGT_PRODUCT_LOCALE.DESCRPTION.

```
<DataField Name="Description" ExternalFieldName="DESCRIPTION"
  Mandatory="n" Writable="y"/>
```

2. Suppose that you want to allow for descriptions that are up to 240 characters long:

```
<DataElement Name="Description" DataType="STRING"
  Description="Description" MaxLength="240" />
```

3. Change the corresponding SQL statement that creates the CMGT_PRODUCT_LOCALE table so that the DESCRIPTION column is set to VARCHAR2(720):

```
DESCRIPTION VARCHAR2(720) DEFAULT 'Not available',
```

4. Run the appropriate SDK targets (merge and createDB) to make the changes to your implementation of the Visual Modeler.

Note that in this example, the Description data field is widely used by many different data objects and so changing its definition in the **DsDataElements.xml** configuration file can have unanticipated side-effects elsewhere. An alternative approach is to create a new data field called ProductDescription and to use this in the ProductLocale data object. Thus, you could put in the **ProductLocale.xml** file:

```
<DataField Name="ProductDescription"
  ExternalFieldName="DESCRIPTION" Mandatory="n" Writable="y"/>
```

Then put in the **DsDataElements.xml** configuration file:

```
<DataElement Name="ProductDescription" DataType="STRING"
  Description="This is the product description field"
  MaxLength="240" />
```

Note also that if you provide a Javascript methods to validate that users have entered valid data in fields, then when you check for length of fields, check for the length specified in the corresponding DataElement.

Email Templates

If your system supports languages other than English and your installation of the Visual Modeler uses email templates to generate messages that are sent to users, then bear in mind that these need to translated.

Release 6.4 has introduced the ability to use JSP pages to generate email messages: This provides support for internationalizing email messages by using the existing framework for internationalizing JSP pages.

For legacy applications, you can use the default templates provided by the Visual Modeler: these are located in *debs_home*/Sterling/WEB-INF/templates/.

HTML Pages

Static HTML pages must be translated where appropriate. If you want to provide support for multiple languages simultaneously, then you should take care to produce pages for each language. Provided that you maintain the location of these pages consistently across your locale directory structure, then the relative references to these pages will always resolve correctly to the correct HTML page.

For example, the following JSP fragment will dynamically generated URLs to point to a locale-specific **Example.html** page:

```
<A HREF="<cmgt:link app="catalog">
/static/Example.html
</cmgt:link">
resourceBundle.getString("ExamplePage")
</A>
```

In this example, a resource bundle is used to determine the displayed text for the link.

Images

In general, use images that do not have embedded text. Doing so, ensures that you can use the same images in more than one locale: thereby reducing the cost of localization and maintenance.

However, where necessary you should provide localized versions of images. Just as for static HTML pages, you can use relative URLs to ensure that locale-specific images are retrieved from the correct location relative to the JSP page.

In particular, remember that all of the buttons in externally facing pages are image buttons with text. Where necessary, you should create localized versions of each button. The image source URLs can then be generated as follows:

```
<IMG ALT="Locale-specific alternate text goes here"
SRC=" ../images/button.gif"></A>
```

Javascript

Take care to localize displayed text used in your Javascript. For example, alert dialog boxes should reflect the user's locale in the displayed text.

- Some Javascript files are included in the Web pages along these lines:

```
<script language='JavaScript' src='../js/genericUtil.js'>
</script>
```

You must maintain these Javascript files for each locale so that the browser can correctly include these in the generated Web pages.

- When Javascript is defined within a JSP page or an included JSP fragment, then display text must be wrapped in the text tag. For example:

```
alert("<cmgt:text id="">Product ID is missing.</cmgt:text");
```

When these tags are processed as part of the SDK tool, then the id attribute is changed into a unique ID, and the ID and body of the tag are added to the resource bundle for the JSP page or fragment.

JSP Pages

In general, all localization for labels, explanatory text, populated lists, and locale-specific formatting for dates and currencies should be reflected in the JSP pages created for a locale.

A useful organizing principle is to create a HashMap of all localized strings on page, and then to refer to this throughout the rest of the page. For example:

```
HashMap localized = new HashMap();
localized.put("TaskListHeader",
    cmgtText("cmgt_taskMgr/TaskWorkspaceData_3","Task List:"));
localized.put("QuickSearchTitle",
    cmgtText("cmgt_taskMgr/TaskWorkspaceData_4","Search for Tasks"));
localized.put("TaskID",
    cmgtText("cmgt_taskMgr/TaskWorkspaceData_5","ID"));
localized.put("TaskName",
    cmgtText("cmgt_taskMgr/TaskWorkspaceData_6","Name"));
localized.put("Status",
    cmgtText("cmgt_taskMgr/TaskWorkspaceData_7","Status"));
localized.put("Priority",
    cmgtText("cmgt_taskMgr/TaskWorkspaceData_8","Priority"));
localized.put("CreateDate",
    cmgtText("cmgt_taskMgr/TaskWorkspaceData_9","Create Date"));
request.setAttribute("localized", localized);
```

You can reference these strings using the scripting capabilities along these lines:

```
<cic:span css="banner" value="${localized['TaskListHeader']}"/>
```

This technique has the advantages that JSP pages are more readable, that you can re-use localized strings easily, and it is closer to the JSF model.

See "Calendar Widget" on page 338 for information about localizing this UI component. For example, populate a drop-down list of days of the week for a French-language locale as follows:

```
<SELECT Name="DayOfWeek">
<OPTION VALUE=0>dimanche</OPTION>
<OPTION VALUE=1>lundi</OPTION>
<OPTION VALUE=2>mardi</OPTION>
<OPTION VALUE=3>mercredi</OPTION>
<OPTION VALUE=4>jeudi</OPTION>
<OPTION VALUE=5>juin</OPTION>
<OPTION VALUE=6>vendredi</OPTION>
<OPTION VALUE=7>samedi</OPTION>
</SELECT>
```

You can also use resource bundles to manage locale-specific display information. For example, this would be an alternate method for populating a drop-down list of days of the week in the Gregorian calendar:

```
<SELECT Name="DayOfWeek">
<OPTION VALUE=0><%= resourceBundle.getString("Sunday") %></OPTION>
<OPTION VALUE=1><%= resourceBundle.getString("Monday") %></OPTION>
<OPTION VALUE=2><%= resourceBundle.getString("Tuesday") %></OPTION>
<OPTION VALUE=3><%= resourceBundle.getString("Wednesday") %></OPTION>
<OPTION VALUE=4><%= resourceBundle.getString("Thursday") %></OPTION>
<OPTION VALUE=5><%= resourceBundle.getString("Friday") %></OPTION>
<OPTION VALUE=6><%= resourceBundle.getString("Saturday") %></OPTION>
</SELECT>
```

Calendar Widget

When you use the calendar widget in a JSP page, then it must be localized. You do this by customizing the **I18N.js** Javascript file to be found in the locale directory *debs_home/Sterling//la/CO/js/*. For example, to support the de_DE locale, create a file called *debs_home/Sterling/de/DE/js/I18N.js* that reads:

```
// DEFAULT LOCALE (English)
var MONTH_NAMES = new Array('Januar', 'Februar', 'Maerz', 'April',
'Mai', 'Juni', 'Juli', 'August', 'September', 'Oktober', 'November',
'Dezember', 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
'Sep', 'Okt', 'Nov', 'Dez');
var DAYOFWEEK_HEADER_NAMES = new
Array("So", "Mo", "Di", "Mi", "Do", "Fr", "Sa");
```

```
var WEEK_START_DAY = 0;
// Create CalendarPopup object
var popupCal = new CalendarPopup();
```

Style Sheets

The Visual Modeler uses cascading style sheets to set the formatting of HTML elements. If you use fonts for a specific locale, then make sure that you create a style sheet that specifies these fonts. For each locale save this locale-specific style sheet in the same relative location.

In JSP pages, you can include a locale-specific cascading style sheet, say **customer.css**, with the following:

```
<LINK rel="stylesheet" href="../../css/customer.css" type="text/css">
```

System Properties

In general, the configuration files only present data to administrators. To localize these files, you should not need to change the names or values of elements, but you should consider changing the Help text for elements. Note that there is only one set of configuration files for each Visual Modeler, and so you should use the language of the default system locale for these files.

Resource Bundles and Formats

PropertyResourceBundles and Properties Files

The Visual Modeler makes extensive use of properties files to manage locale-specific data. These have replaced the use of ResourceBundle Java classes. See "Supporting Locales" on page 328 for more details.

ResourceBundles

A useful mechanism to manage localization is the use of Java ResourceBundles.

Note: The use of resource bundles classes in the Visual Modeler is deprecated. You should use properties files as described in "Supporting Locales" on page 328.

These are classes that manage locale-specific information. ResourceBundle classes used in the Visual Modeler all extend the ListResourceBundle. These define the mapping between name Strings and the value Strings returned when the *getString* (*String nameString*) method is invoked.

By following the naming convention for ResourceBundles, you can create locale-specific ResourceBundles for all of the locales you need to support. For example, you can create the following ResourceBundles to be used in a new application called Inventory:

- InventoryResourceBundle
- InventoryResourceBundle_fr
- InventoryResourceBundle_fr_FR
- InventoryResourceBundle_fr_CA

The following scriptlet can retrieve the appropriate resource bundle for use in a JSP page:

```
<%  
    String baseName = "AdvisorResourceBundle";  
    ResourceBundle resourceBundle =  
        AdvisorResourceBundle.getBundle (baseName,  
            session.getLocale ());  
%>
```

NumberFormats and DateFormats

You can use the NumberFormat class to help you display numbers in locale-specific ways. You create an instance of a NumberFormat by passing in the locale to the constructor.

For example, the following scriptlet displays the total number of shopping carts in a format appropriate to the locale:

```
<%  
    NumberFormat numberFormat =  
        NumberFormat.getInstance (session.getLocale ());  
    int number = request.getParameter ("ShoppingCartsTotal");  
%>  
<P>The number of active shopping carts in use is:  
<%= numberFormat.format (number) %>  
</P>
```

Similarly, use the DateFormat class to help you display date in locale-specific ways. You create an instance of a DateFormat by passing in the locale to the constructor.

For example, the following scriptlet displays the current date in a format appropriate to the locale:

```
<%  
    DateFormat dateFormat =
```

```
        DateFormat.getInstance(session.getLocale());
        Date todaysDate = new Date();
    %>
    <P>It is now:
    <%= dateFormat.format(todaysDate) %>
    </P>
```

This chapter describes the framework for exception handling in the Visual Modeler. You should follow this to ensure consistency across your implementation of the system, and to help other people working on the implementation.

ComergentException Hierarchy

Exception Root

ComergentException

All compile time exception classes declared in the production software should inherit ultimately from `com.comergent.dcm.util.ComergentException` class. This class extends `java.lang.Exception` to provide chaining and an independent user message.

ICCEException

`ICCEException` provides a convenience subclass of `ComergentException`. Rather than create a set of exception classes for a subsystem, you can use the `ICCEException` class uniformly across a subsystem.

ComergentRuntimeException

All runtime exception classes should inherit from `com.comergent.dcm.util.ComergentRuntimeException`, which extends `java.lang.RuntimeException` to provide identical functionality.

Subsystem Grouping

A subsystem of the Visual Modeler is defined to be either a distinct and separable application, or an application level or a system level service. A subsystem is a logical organization. It may span multiple packages in the Java package hierarchy or comprise part of a package.

Each logical subsystem is expected to declare its own exception root class. This root inherits from `ComergentException` and is the parent class of all compile time exceptions within the subsystem. The subsystem is defined to be either a distinct and separable application, or an application level or a system level service. A subsystem is a logical organization. It may span multiple packages in the Java package hierarchy or comprise part of a package, although you should organize your package structure in conformance with the logical subsystem organization.

For example, suppose there is a subsystem named `Foo`. There should be a class `FooException`:

```
public class FooException extends ComergentException
{
    public FooException(String msg)
    {
        super(msg);
    }

    public FooException(String msg, Exception ex)
    {
        super(msg, ex);
    }
}
```

Suppose `Foo` responds to a bad initialization state by throwing `BadInitializationException` for all subsequent requests. This exception would inherit from `FooException`:

```
public class BadInitializationException extends FooException
{
    ...
}
```


Subsystem by Subsystem Exception Policy

Each subsystem should implement a consistent policy for differentiating exceptions. Either it should subclass the subsystem exception class for each distinct exception type (this is the standard Java style policy) or the subsystem's root exception should inherit from `ICCEException`, and should set the status parameter to differentiate exceptions (this is the `ICCEException` policy).

For example, if subsystem Foo chooses a Java style exception policy, then `FooException` should extend `ComergentException`. If subsystem Bar chooses an `ICCEException` policy, then `FooException` should extend `ICCEException` (which in turn extends `ComergentException`).

```
public class BarException extends ICCEException
{
    ...
}
```

Exception Chaining

Each subsystem is expected to throw only exceptions from its own subsystem to its caller. If an underlying service throws an exception that a given subsystem cannot handle, then it is expected to catch that exception and rethrow an exception that is meaningful in its own context. The new exception should use a chaining constructor to include the original exception, so that when the exception is finally handled and logged, the original exception is not lost.

For example, suppose subsystem Foo attempts to open a property file and could incur an IO exception. If it implements a Java style exception policy, then it may declare a new exception class, `FooPropertyFileException`, which extends `FooException`. The IO Exception catch statement would throw a new `FooPropertyFileException` with a constructor that passes a message and the original I/O exception.

```
try
{
    ...
    Properties props = new Properties();
    props.load(input);
    ...
}
catch (IOException ex)
{
    // chain the io exception
    throw new FooPropertyFileException("Loading file" + filename, ex);
}
```

Throwing, Catching, and Logging Exceptions

When to Throw Exceptions

Exceptions should be thrown when the contract between a method and its caller cannot be fulfilled. This is the usage identified in the Java Language Specification. Unfortunately, this provides only a little guidance since the contract can be defined so broadly that exceptions are unnecessary, or defined so narrowly that exceptions occur frequently. As a general rule of thumb, exception usage should balance the following two opposing goals:

Exceptions should not be the norm.

- They involve the creation of an additional object, so, if only from a performance standpoint, it is problematic if exceptions can occur frequently.
- Mixing data and control should be avoided. The alternative to throwing an exception is often returning a null value from a method. This means that the return value encapsulates two meanings (success or failure and whatever the data means when present). It is good programming practice to avoid this usage where possible.

If null is a reasonable value for the stated purpose of a method, or if a method is expected to fail often in the normal course of operation, then it is reasonable to return null to indicate failure; otherwise it is better to throw an exception.

Throwing Runtime or Compile Time Exceptions

According to the Java Language Specification, runtime exceptions should be thrown when the caller has provided erroneous input (in essence, breached the method contract) and it would be burdensome to declare a compile time exception. For example, if a caller invokes a method passing a negative value for a parameter that is an array index, it is reasonable to throw a runtime exception. Otherwise throw compile time exceptions.

Catch Clauses and Throws Declarations

Catch clauses and throws declarations should avoid being overly general. If the called method throws, for example, `FileNotFoundException`, then the caller should catch `FileNotFoundException`, not `Exception` or `Throwable`. The reason for this is that if the underlying code changes to throw a new exception, or ceases throwing

this exception, then it is desirable that the change produces a compilation error to signal to the programmer to consider the new situation.

There are exceptions to this rule where practicality should prevail. If the variety of exceptions that can be thrown is large and our response is the same in all cases, then there is no reason to catch each individually.

Logging Exceptions

If a method catches an exception and handles it (that is, does not rethrow it) then it should log it. Presumably this method knows the significance of the exception, and knows whether to log it with an error severity or some other lower level severity. Empty catch statements should be regarded with great suspicion.

Never do this:

```
catch (SomeException ex)
{
}
}
```

Do this:

```
catch (SomeException ex)
{
    Global.logVerbose(ex);
}
}
```

Or this:

```
catch (SomeException ex)
{
    ex.printStackTrace(Global.debugStream);
}
}
```

When exceptions from underlying subsystems or third party packages are caught and chained to a new exception, there is no need to log the exception. Some process further up the hierarchy will eventually catch and handle it, and the process will know how to log it.

Displaying Exceptions

In general, users of the Visual Modeler should not see exceptions: the appropriate subsystem must handle the exception gracefully by responding appropriately to the error condition.

The Visual Modeler error pages place the exception stack trace between HTML comments. By viewing the source of the displayed Web page, you can read the stack trace.

If an exception stack trace is passed to the JSP page, then bear in mind that the buffer limits of the JSP page may prevent a full exception message from being passed to the Web page. If a long exception stack trace is passed to a JSP page, then you can display it by modifying the buffer of the JSP page. Use the buffer tag as follows:

```
<%@ page buffer=1024kb %>
```

Once the error condition has been diagnosed and fixed, then you should remove this tag because it impacts performance.

This chapter describes the creation of cron jobs that run as part of the Visual Modeler.

Overview

Certain tasks within an implementation of the Visual Modeler are not initiated in response to user input. For example, the hourly synchronization of order data with an external system or the weekly import of catalog data from a third party is best done without user intervention. These jobs can be scheduled to run at suitable intervals using the Job Scheduler functionality provided by the Visual Modeler.

Cron jobs can be defined either as system cron jobs or as application cron jobs.

- A system cron job is run by the Visual Modeler and is not associated with any user. A system cron job calls Visual Modeler classes directly. A system cron job must be run by a class that extends the `SystemCron` abstract class. Typically, system cron jobs perform tasks such as cleaning the cache.
- Each application cron job is run as a user: the username and password of the user are provided when the cron job is created using the Job Scheduler user interface. Application cron jobs work by posting XML messages to the Visual Modeler which are then processed by the system. An application cron job must be run by a class that extends the

ApplicationCron abstract class. Typically, you use application cron jobs to perform necessary administrative tasks that touch user or product data such as order synchronization.

Attention: Note that a system cron job should not attempt *restore()* and *persist()* operations itself. There is no user associated with the cron job class and so the access checking built in to the data access methods will throw an exception.

CronManager and CronScheduler

The definition and creation of cron jobs is managed by the CronManager class. Cron job configuration information is represented in memory by the CronConfigBean data bean. The definition of cron jobs are maintained in the Knowledgebase.

The scheduling and running of cron jobs is managed by the CronScheduler class. This singleton class is instantiated at server startup time.

CronJob Interface

Each cron job is a Java class that implements the CronJob interface:

```
public interface CronJob extends java.lang.Runnable
{
    /**
     * Specify the Cron Configuration bean object.
     *
     * @param config Cron configuration bean object.
     */
    public void setCronConfiguration(CronConfigBean config);

    /**
     * Return the Cron Configuration bean object.
     *
     * @return CronConfigBean object.
     */
    public CronConfigBean getCronConfiguration();

    /**
     * Initialization function. This function is called
     * immediately after the object is created.
     *
     * @return true if initialization success, false otherwise.
     */
    public boolean init();

    /**
```

```
* Return the current scheduled time.
*
* @return Current schedule time in Calendar object.
*/
public Calendar getSchedule();

/**
 * Reschedule the cron to reflect the changes made to the
 * configuration parameter. This function is called by the
 * Cron Manager whenever cron configuration changes.
 */
public void reschedule();

/**
 * Whether the job needs to be run again. This function is
 * useful if there is some problem in the current run and you
 * want to retry at specified time.
 *
 * @return true if the job is allowed to retry if the job
 * did not run successfully
 * on the last time of execution
 */
public boolean retry();

/**
 * Determines whether to stop this cron job from running.
 *
 * @return true if the job has been slated to not run again
 */
public boolean stopRun();

/**
 * Compute next cron run time: this is usually based on the cron
 * run interval.
 */
public void computeNextSchedule();

/**
 * Check to determine if the cron job is
 * in a good state to run before triggering the thread to run.
 *
 * @return true or false. True means ready to run.
 */
public boolean isOKtoRun();

/**
 * Is called when the thread starts.
 *
 * @return false if the job needs to be stopped. Return true to
```

```
    * continue running.
    */
    public boolean service();

    /**
     * Checks whether the next run time is later than the end run date.
     *
     * @return true if next run time greater than end run time
     */
    public boolean isExpired();
}

```

To create a new cron job, follow these steps:

1. Write a CronJob class: you must extend either the SystemCron or ApplicationCron classes. Both these classes are abstract and they both extend the abstract class AbstractCronJob.

The only method that you need to implement is *service()*. This is the method that processes the inbound post initiated by the CronScheduler.

- If the job is passed parameters that are defined using the Job Scheduler user interface, then you can retrieve the parameters using the *getParameter(String s)* and *getParameters()* methods of the AbstractCronJob class. These methods behave identically to the corresponding methods of the HttpServletRequest class.
 - If you want the result of the job to be saved to the database, then the *service()* method must call the *setExecutionOutcome(String s)* method.
 - You can specify that the cron job should be re-executed at a later time by calling the *setRetry(Calendar c)* method of the AbstractCronJob class. Use the Calendar parameter to specify when the job should be re-executed.
2. Using the Job Scheduler user interface provided as part of the system administration application, define the cron job by specifying the cron job class, the schedule to determine when it is run, and any parameters to be passed to the cron job at runtime. If the cron job is to run as an application cron job, then you must also provide the username and password of the user.

Parameters are passed in to the cron job using the same syntax as for HTTP request parameters. For example: Name1=Value1&Name2=Value2.

This chapter describes how you can use filters. It covers:

- "Filters Overview" on page 353
- "Available Filters" on page 354

Filters Overview

A filter is an object that performs filtering tasks on either the request to a resource (a servlet or static content), or on the response from a resource, or both. They are defined as part of the J2EE 2.3 specification.

Filters perform filtering in the *doFilter()* method. Every Filter has access to a FilterConfig object from which it can obtain its initialization parameters, a reference to the ServletContext which it can use, for example, to load resources needed for filtering tasks.

Filters are configured in the deployment descriptor of a Web application. Examples of typical filters include:

- Authentication Filters
- Logging and Auditing Filters
- Image conversion Filters

- Data compression Filters
- Encryption Filters
- Tokenizing Filters
- Filters that trigger resource access events
- XSLT filters
- Mime-type Chain Filters

Available Filters

This section describes some of the filters provided in the Visual Modeler. All the filters are part of the `com.comergent.dcm.core.filters` package. It covers:

- "DosFilter" on page 354
- "WSDLFilter" on page 355

DosFilter

This filter can be used as the basis for filters to protect the Web application from denial-of-service attacks.

To use this filter, write a class that extends the `com.comergent.dcm.core.filters.DosFilter` class, and in it, override the `isRequestDenied()` method to implement the logic you want to use to identify and block denial-of-service attacks.

Then, modify the `web.xml` configuration file, to declare your implementing class as a filter like this:

```
<filter>
  <filter-name>DosFilter</filter-name>
  <filter-class>
    com.comergent.dcm.messaging.CustomDosFilter
  </filter-class>
</filter>
```

and

```
<filter-mapping>
  <filter-name>DosFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

WSDLFilter

The WSDLFilter class is used to transform the Web service WSDLs if they are accessed using the standard URLs: `http://server:port/s/dXML/5.0/OrderInterface.wsdl`, and so on.

Managing and Displaying Constrained Fields

This chapter covers the topic of managing constrained data fields which can take only one of a number of values: we called these data fields *constrained*. Examples include partner levels (such as “Gold”, “Silver”, and so on), partner territories (such as “North-west”, “Benelux”, and so on), and skill levels (such as “Expert”, “Certified”, and so on). You can manage these data fields in different ways in the Visual Modeler. Your choice depends on how they are to be maintained and used.

Options

You have the following options to specify a constrained data field and the permitted data fields:

- Maintain the data field as a set of values in a database table. Assign values to business objects either by a cross-reference table or by references to a key for each value in the business object table.
- Maintain the values as a constraint element in the XML schema (declared in the **DsConstraints.xml** file). Specify the constraint as an attribute of the DataElement associated with the data field.
- Embed the permitted values as values of a <SELECT> form element in an HTML template.

We recommend that you maintain the permitted values for a field as a database table unless:

- the values are not going to be modified at run-time
- the data field may take only one value in each business object
- the values can be displayed in a natural order that is determined by the values themselves such as their alphabetical order.

We recommend against using the third option for the following reasons:

- It becomes a maintenance problem to update templates or application code if you want to modify the list of permitted data values.
- It represents a security problem because users may modify the HTML to pass back forbidden values. You have to either add Javascript (that a user can remove) to validate the selection or validate the returned value as part of the business logic.

Criteria

Your selection depends on the functionality of the data field. Ask yourself these questions to determine how the data field is being used:

1. Can you assign a business object only one or multiple values of a constrained data field?

If your answer is that multiple values may be assigned to the same business object (example: a partner that may operate in multiple territories), then you *must* use a database table for the field values and a cross-reference table to assign values to the business object.

2. Can you enter new values of the data field when creating a new business object or do you need to verify that a value entered for the data field is a valid member of the constraint set?

If only single values are permitted, and your answer to Question 2 is that new values are permitted, then you *must* use a database table to hold the field values. However, you do not have to use a cross-reference table to assign data field values to business objects. You cannot dynamically add values to the list of permitted values of a constraint element through the current Visual Modeler interface.

3. Are the possible values that the constrained data field may take maintained dynamically or are they read once at start-up?

If your answer to Question 1 was single value, and your answer to Question 2 is that new values are not permitted, but you do require dynamic updating, then you *must* use a database table. If the constrained values are unchanged once the Visual Modeler has started, then you can use a constraint element.

4. Do you need to sort the constrained data values for display? If yes, then is it sorted by value (say, alphabetically) or by some defined order that cannot be inferred from the values themselves?

Finally, if the data field values need to be sorted by an order not inherent in the values themselves, then this ordering information must be maintained in a database table. However, if you only order the values using some self-evident ordering (such as alphabetical), then you can use the constraint element choice.

Introduction

This chapter outlines a Best Practices security model behind the Visual Modeler. It covers the following topics:

- “Role Definition and Security Policies” on page 363
- “Information Assets” on page 366
- “Protection Mechanism for Information Assets” on page 369
- “Protection of Critical Functions” on page 370
- “Threat Scenarios” on page 372
- “HTTP Sessions” on page 373

A Web-based application operating as an e-commerce site is likely to be regulated by industry consortiums or must comply with widely-accepted good development practices. In general, most of these situations will involve some guidelines that attempt to not only help develop robust applications, but also provide a relatively secure environment that can adequately safeguard the use of personal or financial information at the site. The range of regulatory or best practice guidelines is fairly extensive. Some provide regulations for which compliance is mandatory for doing business in a particular market; some provide corporate-wide process and

procedures that aid in specific goals, such as protecting personal information; some are more focused on good software engineering practices. Some examples include:

- The Open Web Application Security Project (OWASP)
OWASP is a development community that recommends software best practices.
- ISO 17799 - ISO/IEC Security Standard
ISO 17799 is a standard establishing guidelines and general principles for initiating, implementing, maintaining, and improving information security management in an organization.
- Payment Card Industry (PCI)
PCI is an industry consortium that regulates sites that accept and process credit card transactions.
- Sarbanes-Oxley (SOX)
SOX is a US legal requirement for public corporations.
- American Institute of Certified Public Accountants SAS 70 (AICPA)
AICPA's Statement on Auditing Standards no. 70 (SAS 70) is an auditing standard by which organizations can assure that they have adequate safeguards and controls in place for hosting or processing data belonging to their customers.

Guidelines and practices included through these and other bodies range from good software engineering practices to building secure networks to corporate accounting practices. In keeping with this environment, this chapter presents the basic building blocks that are likely to be needed to operate a robust and secure site. Not all aspects of all organizations' guidelines will apply to any one site, but it is likely that some of these guidelines could be mandatory. Use this chapter as a starting point in providing a compliant solution, but more in-depth practices may be needed.

This chapter covers:

- "Role Definition and Security Policies" on page 363
- "Information Assets" on page 366
- "Protection Mechanism for Information Assets" on page 369
- "Protection of Critical Functions" on page 370
- "Threat Scenarios" on page 372

Role Definition and Security Policies

Administration Model

This section describes the entities assumed to be present in the administrative domain in which the Visual Modeler resides, including networks, servers, and administrative roles. This is likely not an exhaustive list. It is likely that various network devices will exist within this environment, and perhaps other servers.

Networks

The following network zones are assumed to exist. These networks are connected to themselves as outlined below through gateways.

- **External Network:** Directly visible from the Internet. Hosts the Web server with static content. The External network is accessible to the Internet through a firewall. It is assumed that the firewall and appropriate standard security practices are sufficient to prevent shell level access from the Internet. The external network has a gateway to the De-Militarized Zone (DMZ) that permits highly controlled access from the Web server to the application server.
- **DMZ:** This network is not directly visible from the Internet. A constrained gateway permits the Web server residing on the External Network to access the Application server residing on this network and another, similar, gateway permits access from the DMZ network to the Internal Network. Web server routes messages to the application server through dedicated ports.
- **Internal Network:** Not visible from the Internet, nor from the External Network. Database resources reside here. Application servers in the DMZ connect to Database Servers in this network through a constrained gateway.

Servers in the DMZ and Internal Network spaces have private addresses, complying with RFC 1918. In general, only the minimal access needed to operate the systems should be allowed.

Servers

The following servers are assumed to exist. The term “server” here represents a software application that is more or less continuously listening on one or more network ports, responding to requests received on the ports. Software servers reside on computer hardware. Generally, though not necessarily, there will be a one-to-one relationship between a server software system, and a server hardware entity.

- Web server resides in the External Network. It responds to HTTP and HTTPS requests from the Internet or internal corporate intranets.
- Application server resides in DMZ. Some HTTP and HTTPS requests are delegated to the Application server for dynamically generated response. The Application server maintains connections with the Database server.
- Database server resides in the Internal Network or DMZ.

Roles

This section describes roles within the administrative context of the Visual Modeler. These are roles assigned to data center personnel acting as employees or agents of the Enterprise. They are distinguished from the roles of individuals who interact directly with the Visual Modeler Web application (online users). Online users have capabilities managed directly by two Visual Modeler Entitlement services. Dispatch (or “MessageType”) Entitlement service manages page flow privileges. The Access Policy service manages fine-grained data-level access.

- Database Administrator
 - Responsible for Database Servers.
 - Can log into database server.
 - Can read, create, update, or delete databases, database tables, indexes, and other database resources.
 - Can create backups and restore from backups.
 - Can create Database users and manage them.
 - Sets the application user access and authentication.
 - Does not have root level authority in server OS.
 - Does not have direct access to Application Server machine (or network).
 - Does not have Visual Modeler application access
- System Administrator
 - Responsible for Server Hardware, and Server Software.
 - Has root access to server machines within their zone of responsibility.
 - Has the authority to start and stop server processes.
 - As root, can read, write, update, or delete files in file systems.

- Can back up and restore files. Can create OS-level users and manage them.
- Does not have access to log in to database server.
- Deploys the Visual Modeler application in the production infrastructure.
- Loads the minimal data set in the XML files to the database.
- Does not have Visual Modeler access.
- Developer
 - Responsible for preparation of deployment Web archives (“WAR” files).
 - Has the authority to create Web archives representing the Visual Modeler executable.
 - Can set properties and business rules governing Visual Modeler operation, including properties that configure access to the database, properties that configure the JCE Key store, and so on.
 - Has the authority to create or modify the initial Visual Modeler dataset. This dataset is a part of the deployment archive.
 - Does not have any kind of access to the Production Database or Application servers.
 - Does not have username/password to access production Visual Modeler.
 - Does not move code from development/QA environments to production.
- Network Administrator
 - Configures and manages network.
 - Has authority to create and assign network resources, including domain names, IP addresses, firewall level, and so on.
 - Does not have database access.
 - Does not have Visual Modeler application access.

Data Center Roles

The following are assumed about Data Center administrative roles:

- Roles are segregated. System Administrators cannot be Developers, Network Administrators, nor Database Administrators. Database Administrators cannot be Developers, System Administrators, nor Network Administrators, and so on.

- System Administrator Roles are partitioned on network boundaries. A system administrator for the DMZ should not be a system administrator for the Internal Network.
- Data Center Administrators do not have Visual Modeler online administration roles.

Information Assets

Encryption of Persistent Data

This section describes the encryption of persistent data in the Visual Modeler.

1. What data is encrypted?

This is configurable within the Visual Modeler configuration files. Please see other sections for the details of which data fields and the contents of which data fields are encrypted.

2. What encryption libraries are used?

By default, the Sun implementation of JCE is used for encryption, with policy files enabling strong encryption. This is the recommended option.

3. What encryption algorithms are used?

This is configurable among secret key encryption algorithms supported by JCE. The recommended algorithm is AES. The Visual Modeler also supports 168 bit DES encryption.

4. Where are keys stored?

In a password protected JCE Key Store file. By default this file is located in the home directory of the user that initiates the Visual Modeler application, but this can be altered within a configuration file.

5. How is the Key Store password protected?

By default the key store password is encoded in Java. The Visual Modeler supports two phase initialization, if desired, in order to employ a password provided at run time. In this mode of operation, the Visual Modeler will come up partially and await a special HTTPS message containing the key store and key passwords to complete initialization and respond to requests. This technique means that the application server system administrator will not have access to the Visual Modeler password, but it requires more than one person to bring up the Visual Modeler.

6. How are passwords saved within the Visual Modeler?

Either by a cryptographically secure digest (MD5 or SHA-1) or by secret key encryption as described above.

Information Assets

This section introduces the critical information assets that are dealt with by the Visual Modeler.

Account profile

The user's account information is maintained in the Account profile. The user's account is identified by the (unique) account-id. The account profile contains personal financial details such as credit card numbers and BillTo and ShipTo addresses. Implementation can be customized, so this profile may contain additional or fewer details, encrypted fields, and so on.

Transaction and System Log

According to the type of data and level of role separation required, Visual Modeler logging can be configured to capture all transactional and non-transactional requests to file locations outside of the Visual Modeler deployment infrastructure. Certain aspects of order and account history can also be captured in the Visual Modeler database for administrative and end user interaction purposes but do not represent a secure audit trail.

Network considerations need to be included for remote and/or more secure logging.

Key Store File

The Key Store file stores the encryption keys that protect Visual Modeler data. This file normally resides in the home directory of the user that initiates the Visual Modeler Web application, but this can be altered within a configuration file.

<p>Attention: The Key Store file is a requisite to encrypt or decrypt data in the knowledgebase. This Key Store file must be well protected. With this Key Store file, the encrypted data can be read, so it must never be included on the same media as a database backup. If the Key Store file is lost, then data cannot be read from the database.</p>

WAR local key store file

The WAR local key store file stores the keys that protect the database username and password used by the Visual Modeler. It is contained in the Visual Modeler WAR file structure.

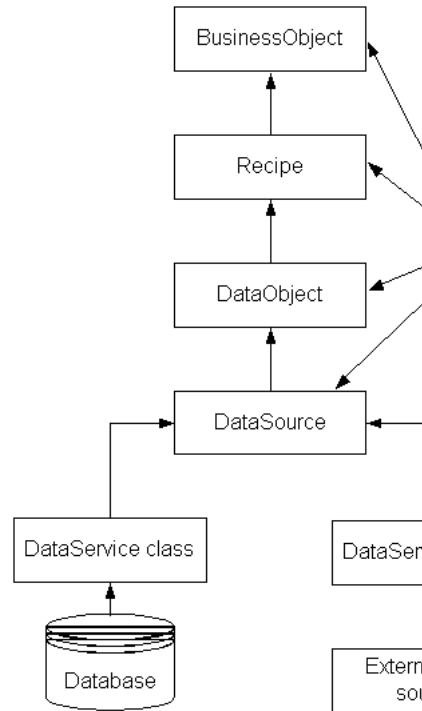
User passwords

Visual Modeler user names, passwords, and user profile information are stored in the user authentication and user contact tables. The passwords are protected as described in “Protection Mechanism for Information Assets” on page 369.

Roles Schematic

The following indicates the spheres of access for the following classes of administrators:

- NA: Network Administrators
- SA: Security Administrators
- WA: Web Administrators
- SA: System Administrators
- DBA: Database Administrators



Protection Mechanism for Information Assets

This section discusses the encryption mechanisms used to protect the integrity of the critical data fields of the Visual Modeler.

Credit Card Information

The following data fields are encrypted using the DES 168 bit encryption by default:

- Credit Card Number

- Credit Card Holder Name

<p>Note: The InlineDES encryption is not considered a strong encryption for purposes of a PCI compliance audit on a production server. For a system to meet that level of compliance, a specific cryptography configuration must be implemented for production.</p> <p>See the PCI Compliance Guide, http://www.pcicomplianceguide.org, for more general information about PCI compliance.</p>

User Passwords

User passwords are protected by 1-way encryption (MD5 or SHA-1 Message Digest).

Only the digests of the passwords are stored in the Visual Modeler database. Alternatively, at the discretion of the customer, the password can be encrypted with DES 168 bit encryption or other encryption as noted in the section “Credit Card Information” on page 369.

Protection of Critical Functions

Performing functions such as creating and setting database users and passwords and storing sensitive data in a database are critical from both a security and an operational perspective. Equally critical is protecting the performance of these functions from potential attacks or misuse. This section documents the process of performing these functions and explains how that process prevents attacks or misuse.

Setting Application’s Database User and Password

1. The DBA creates a user in the database, assigns a password, and sets appropriate database privileges.
2. The DBA informs the developer.
3. The developer encrypts the password using the key in the WAR local Key Store.
4. The developer initializes connection information in **WEB-INF/schema/DataSource.xml**.
5. The developer prepares the WAR file and gives it to the application server SA.
6. The application server SA deploys the WAR file
7. The application decrypts the password and uses it to connect to the database.

Assertions

- The developer knows the password but does not have access to the database, so cannot use the password.
- The application server SA does not know the password, so cannot access the database.
- The DBA has the password and can access data, but does not have access to the Web application. Sensitive data is encrypted. The DBA does not have access to the system without having access to encryption keys.

Storing Sensitive Data in the Database

1. The developer specifies encrypted fields for sensitive data in the database schema.
2. Where necessary, schema fields are redefined to incorporate correlative data to prevent relocation. For example, account balances may be converted strings combined with an account id to provide the ability to detect field copies.
3. The developer specifies the encryption algorithm and location and type of key store.
4. The developer specifies the key store password.
5. The developer prepares the WAR file and delivers it to the application server SA.
6. The application server SA deploys the WAR file and starts the application server.
7. When sensitive data fields are set, data services invoke the encryption service. Data is encrypted. Eventually, data is persisted to the database.

Assertions

- The DBA has access to the database, but does not have the ability to decrypt sensitive fields.
- The application server SA has access to the key store, but not to the key store password.
- The developer has access to the key store password, but not to the key store.

Threat Scenarios

Transport

Scenario: The Network Administrator intercepts clear text values in communication between the application server and database server, for example, over a JDBC transport stream.

Preventative Measures: Sensitive data is encrypted before transport and would not be usable. The encryption key is never transported.

Restores from Backup

Scenario: The DBA restores database tables from backup to some earlier state in order to create a vulnerability.

Preventative Measures: The Visual Modeler provides no means to prevent this type of attack. Rely upon Data Center practices and procedures.

Detection: It is theoretically possible to detect this attack by correlation to Visual Modeler debug logs and Web server access logs.

Log Files

Scenario: The application server SA edits the Visual Modeler debug log files to conceal some activity. We recommend that operational systems configure debug logging at the “INFO” level. At this level, Web request start and end events, JSP dispatch events, session events, login events, and CRON events are logged.

Scenarios such as an attack that could be concealed therefore involve an attack by a logged in and authorized user.

Preventative Measures: The application server SA has root access to the file system. One can not prevent this sort of activity.

Bogus Account to Access Customer Records

Scenario: The Visual Modeler administrator creates a super CSR who has access to all accounts. The Visual Modeler administrator uses super-CSR to get credit card info, change balances, charge customer credit cards, and so on.

Preventative Measures: CSRs should not be able to see the unmasked or unencrypted credit card numbers. Any activity that involves charges against the valid customer credit cards will be logged to the account transaction log.

Detection: All account activities are logged in the account transactions log table. Unusual activity from a specific CSR can be detected by reviewing the logs in a timely manner.

Credit Card Number Theft

Scenario: A CSR or CSR manager gets a customer's credit card number(s) and commits fraud.

Preventative Measures: When a customer's credit card number is displayed to the CSR, it is masked so that only the last four digits are displayed, and the Card Verification Value (CVV) number is not stored in the database or log files.

CSR's only see the entire credit card number and associated information when they create the account or edit the credit card information on behalf of the customer. There is no practical way to prevent the CSR from stealing this information.

DBA Password Theft

Scenario: A System Administrator obtains the DBA password and performs unauthorized activities on the database.

Preventative Measures: Use best practices rules for password creation and appropriate policies to expire and rotate the passwords periodically. As the Visual Modeler money fields are encrypted using keys and combinational fields, the DBAs will not be able to decrypt and modify the values.

Detection: Review of the database level audit files can detect such activities.

HTTP Sessions

Protecting HTTP sessions in Java servlet-based web applications is key to providing system security. If you do not take appropriate measures to protect sessions conducted on your e-commerce web site, you risk your site's security, expose confidential data, and permit unauthorized control of sessions.

The Visual Modeler is designed to be deployed in environments requiring varying degrees of security. For some environments, all HTTP or mixed HTTP/HTTPS page flow may be sufficient. However, if your system needs to protect confidential

information that has significant value, configure the Visual Modeler to require HTTPS from the login page forward.

<p>Note: If you are running the Visual Modeler in HTTPS mode, ensure that you appropriately set the Sterling eBusiness SSL System Server Port system property. To set this property, navigate to the Visual Modeler enterprise home page, click System Services in the System Administration panel, and then click Commerce Manager in the System Properties panel.</p>

Web server logs and the logs of any other server in the web application data path should be inspected to ensure that they are not capturing application server session IDs.

Many sites are set up to conduct sessions using mixed mode HTTP and HTTPS with the assumption that the user name and password are the only confidential pieces of information. However, in anything but a pure SSL environment, the session ID is what protects identity: it is a time-limited token for the user name and password that gives a user access to all the data in the session, including the authority acquired as part of that session. Everywhere that you protect the username and password, you should also protect the session ID. Robust session protection requires SSL from login to logout. If you use mixed mode, you cannot know that you are conducting a session with the same individual for the entire session.

In Release 7.2 and higher, the Visual Modeler hashes session IDs in logs to increase the security of sessions.

Please note the following information about HTTP sessions:

- An HTTP session is tracked either by a session ID (JSESSIONID:...) embedded in each URL or by a session cookie.
- Sessions expire after a configurable period of inactivity, usually 30 minutes.
- Sessions are not correlated to any particular IP address or hostname. If someone accesses your system with a valid session ID they will be given access regardless of the computer from which they make the request.
- If a session cookie is created from an HTTPS request, it is marked as a secure cookie. Properly written client applications will not transmit that cookie when issuing an HTTP request. However, the fact that a cookie was created as a secure cookie is not subsequently known to the server and client applications can transmit a supposedly-secure session cookie.

If it is important for your e-commerce site to guarantee the confidentiality of the interaction between a user and your server, then that session *must be* HTTPS from

the moment the user logs in until they log out. Mixed HTTP/HTTPS sessions are simply not secure.

Introduction

The best recovery plans focus on prevention. By setting up a robust environment, putting redundant systems in place, establishing regular backup and restore policies, and conducting regular tests that you can recover from backup, you can limit the effects of disasters by providing safeguards for each layer of your deployment infrastructure.

Some decisions about backup and restore policies must be made based upon business criteria. What is the value of the site being available per day for your business? How much data can you afford to lose? How long can your e-commerce Web site be unavailable to customers? What are the time and cost trade-offs for various backup and restore solutions? Answers to these questions will help to determine your backup and recovery requirements.

The simplest backup system might be saving data and a copy of your application to tape or other remote devices and storing that data at a remote data center. A better strategy is to put redundant systems in place for each part of your deployment so that if one system fails, another is already available. The most robust solution is to maintain a mirror image of your site at a remote location and synchronize the image with the live data regularly. The latter solution is the most costly, but is immediately recoverable. The former solutions are less expensive, but require more time and effort to perform recovery.

This chapter covers:

- “Deployment Architecture Overview” on page 378
- “Infrastructure” on page 378
- “Backup Strategies” on page 380

Deployment Architecture Overview

Setting up a rich development environment not only allows for easier site updates and maintenance, but also provides a quick path to rebuild a complete application as a recovery step, if needed. The deployment architecture includes the following elements:

- Build environment: a predictable, known build environment containing all the elements needed to build the deployment, including:
 - JDK’s
 - SDK’s
 - Code repository (such as CVS)
 - Libraries

The steps required to go from code to production vary and may be iterative, but the build environment should contain everything required to rebuild your deployment in a predictable way if necessary.

- QA area: a separate environment for performing quality assurance tasks. QA is the first environment in which work from (possibly) several engineers is integrated to run as a unit.
- Staging area: a separate environment in which the work integrated in QA now runs in a specific context that emulates production.

Infrastructure

A common strategy for establishing a robust infrastructure is “double everything”: put redundant systems in place so that when a primary system fails, a secondary system comes online as quickly as possible. The following figure shows a typical infrastructure consisting of three tiers:

- Web tier: all components handling requests from and serving content to Web browsers.

- Application tier: all components processing requests from and serving dynamic content to the Web tier, usually with data from the database tier.
- Database tier: all components serving data to the application tier.

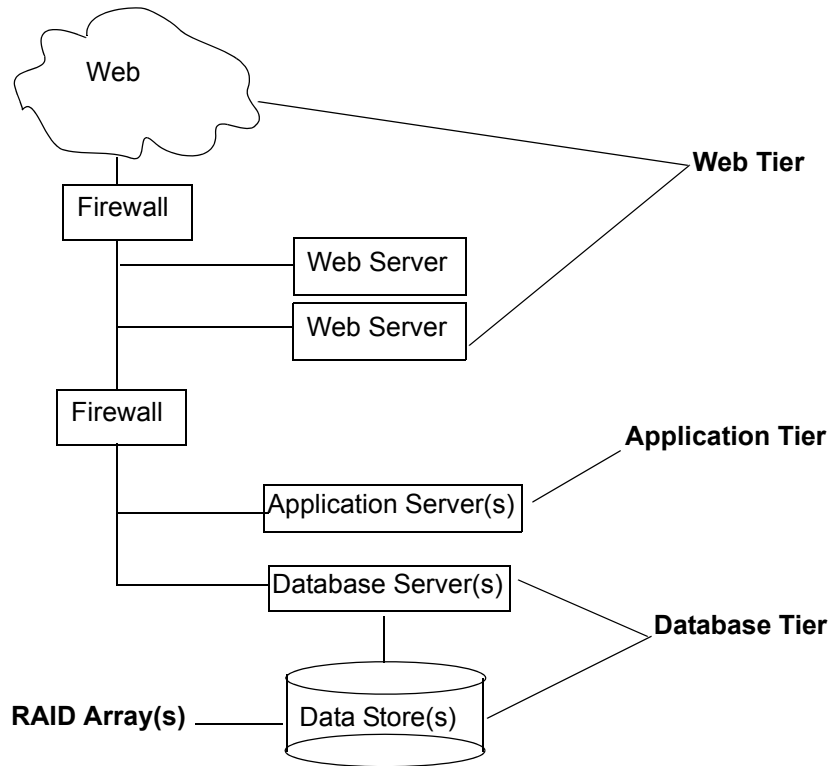


FIGURE 13. Typical Deployment Infrastructure

Two Web servers ensure that if one fails, the other can continue functioning. Having a second firewall in place to provide additional protection to data also satisfies certain regulatory requirements for securing data. See CHAPTER 29, “Security Best Practices” for more information about securing data.

One strategy for providing physical protection for your data is to store production data in a RAID device. If a single drive fails, then there is no loss of data. There is a mechanical threshold with such a strategy: depending upon your configuration, if

more than the threshold's number of drives fails, you will lose data. That is something to consider as you determine your requirements.

Backup Strategies

There are different backup strategies for protecting data and for recovering your application servers and Web servers. Most backup strategies come down to saving copies of what is already running on the application and Web tiers. The following sections describe backup strategies for databases, application servers, and Web servers.

Database Recovery

Your backup strategy determines how quickly you can recover data after a disaster. Determine your acceptable timeline for getting your database back up and running, including time needed to rebuild the OS and reload the database if necessary, and plan your backup policy accordingly.

The following are the types of database backups you should perform:

- **Checkpoint backups:** database servers log activity on the transaction level as transactions are committed. Checkpoint backups write a log of transactions since the last time a checkpoint backup was done. Write the checkpoint log to a separate physical device. This creates a snapshot of the database activity on a transaction level. If the database fails, there's a chain of records that enable reconstructing the activity.

The interval at which your deployment performs checkpoint backups is a business decision. If your site does millions of dollars of business each hour, doing checkpoint backups several times each hour would be advisable. If your site has only a few transactions each hour, you can do checkpoint backups less frequently. Determine an interval that allows data recovery at your level of business activity.

- **Daily incremental backups:** incremental backups save only those files that change each day. Performing daily incremental backups to a different facility, rather than using physical media, is a good strategy. The backup is effectively a disk-to-disk copy.
- **Weekly full backups:** full backups save the entire database, not just the files that have changed since the last backup. Performing full backups to a different facility is a good strategy.

A typical recovery scenario might be:

1. Perform the initial restore from the last full backup.
2. Next, perform restores in date order from the daily incremental backups.
3. Finally, reconstruct the last few hours' activity using the checkpoint backups.

Application Server and Web Server Recovery

When planning recovery policies for your application and Web servers, plan to be able to build exact replacements if either application or Web servers fail: this is the “build another one and go” principle.

Ensure that you have copies of everything that makes your application and Web server deployments unique: configuration files, properties files, the JVM, original source code from your CVS repository, and so on. Back up all static data kept on the Web tier, such as any custom JSP pages. Make sure that your backup process includes coverage for Web server and container configuration files or other files needed to operate your site, but which may not be considered as source code and are therefore not kept in a source code repository.

Use your QA and staging environments as starting points for rebuilding your servers and getting back to an operational state.

Part 3: Best Practices

The chapters in this section of the guide provide information about the best practices that may need to be followed while using the Visual Modeler.

Database Management Best Practices

Introduction

This chapter describes database management best practices and the activities that support those practices. Database management practices protect your data, ensure data integrity, and address database performance issues. Database management activities include:

- Archiving active data
- Monitoring database tables
- Sizing database tables and purging old data at intervals dictated by business requirements or regulatory audit compliance, such as Sarbanes-Oxley or HIPAA
- Updating indexes and managing database statistics

For recommended database archival practices, see CHAPTER 30, “Backup and Recovery Best Practices”.

This chapter covers:

- “Archiving Data” on page 386 presents general guidelines for archiving data.

- “Monitoring Database Tables” on page 386 presents guidelines for monitoring and sizing database tables, key tables to monitor, guidelines for purging data, and creating and using history tables to retain old data online in compliance with regulatory or business requirements.
- “Updating Statistics” on page 388 presents scripts for updating the database statistics for an Oracle and a SQL Server database.

Archiving Data

Managing your data is key to protecting your business. Archive your production data regularly and set up primary and secondary locations for storing your database archives, preferably off-site. Establish a regular schedule of archival activities, including daily incremental backups and weekly full backups (more often if your business volume demands it). See CHAPTER 30, “Backup and Recovery Best Practices” for more information about database backup strategies.

Monitoring Database Tables

Auditing compliance requirements such as Sarbanes-Oxley and HIPAA may dictate how long you must store production data online. In general, performance starts to degrade when the number of rows in a database table grows to between 50 million and 100 million rows. To avoid performance problems, monitor the size of your database tables and start transferring data to history tables when the number of rows in a particular table reaches about one million rows. Data that is no longer needed by your application, such as data that is over one year old, should be considered historical data and transferred to history tables. History tables are tables that are online but that are not accessed by the Visual Modeler application and therefore do not affect performance.

Key Tables To Monitor

While you should monitor the size of all the Visual Modeler database tables (including custom tables added during your implementation), the following tables tend to grow quickly and should be monitored on a daily basis.

CMGT_CART_CONFIGURATION

CMGT_CART_CONFIGURATION_LINES

CMGT_CART_LINES

CMGT_CATALOG_TO_CART_LOG

CMGT_CATALOG_TO_CONFIGURE_LOG
CMGT_NOTE
CMGT_OIL
CMGT_OIL_HEADER
CMGT_OIL_LI
CMGT_ORDER_ADDRESSES
CMGT_ORDER_EXTN
CMGT_ORDER_LI_EXTN
CMGT_ORDER_LI_SHIP
CMGT_ORDER_SERIAL_ITEMS
CMGT_QUOTE_EXTN
CMGT_RFQ_EXTN
CMGT_RFQ_LI_EXTN

Purging Data

Business requirements and regulatory compliance requirements will dictate how long you must retain data, whether online or in offline archives. Establish notification protocols and procedures to handle purging data appropriately for your requirements.

Creating and Using History Tables

The names of history tables are appended with `_H`. See “Key Tables To Monitor” on page 386 for more information about the key database tables to monitor.

Some history tables are created during the Visual Modeler implementation, such as `CMGT_INVOICE_LI_H`, `CMGT_OIL_LI_H`, and `CMGT_ORDER_LI_EXTN_H`. If you need to trim the size of a database table that does not already have an associated history table, then you can create one. To transfer historical data to a history table:

1. Create a new table with exactly the same structure as the table whose size you need to trim. The name must be of the following format:

TABLE_NAME_H

TABLE_NAME is the name of the table to contain the historical data and *_H* indicates that this is a history table, not to be used in indexing and not to be accessed by the Visual Modeler application. The name should be similar to the name of the original table. For example, the history table *CMGT_INVOICE_LI_H* contains historical data for the *CMGT_INVOICE_LINES* transactional table.

2. Transfer the historical data to the history table from the original (transactional) table, using business criteria to determine which data is no longer needed. For example, you might place data that is more than a year old into a history table.

The following sample SQL statement transfers historical data to a history table:

```
INSERT INTO CMGT_TABLE_NAME_H SELECT * FROM CMGT_TABLE_NAME WHERE  
ACTIVE_FLAG = 'N';
```

You can also use the DBMS export command, import the data to the appropriate history table, and then delete the transferred data from the original table.

The following sample SQL statement deletes transferred data from the original table:

```
DELETE FROM CMGT_TABLE_NAME WHERE ACTIVE_FLAG = 'N'
```

You can still see and access any history tables if necessary by doing SQL queries on the database. When the data ages past its required retention time, you can use standard tools to purge it. If you do not have auditing compliance requirements, other business decisions may determine when you should purge your data.

Updating Statistics

Updating the database statistics allows the database query optimizer to re-examine database indexes and re-compute the most efficient paths for retrieving data. This section presents two scripts: one to update statistics for an Oracle database, and one to update statistics for a SQL Server database.

Please consult your DBA to get the statistics updated on the tables properly or refer to your database documentation for further help.

Updating Statistics For an Oracle Database

The following example shows how to update statistics for an Oracle database at the schema level. Replace *schema name*, *owner name*, and *table name* with the appropriate schema, owner, and table names.

```
EXEC DBMS_STATS.GATHER_SCHEMA_STATS (
ownname=> 'schema name' ,
cascade=> TRUE,
estimate_percent=> DBMS_STATS.AUTO_SAMPLE_SIZE,
degree=> DBMS_STATS.AUTO_DEGREE,
granularity=>'AUTO',
method_opt=> 'FOR ALL COLUMNS SIZE AUTO');
```

The following example shows how to update statistics for an Oracle database at the table level.

```
exec dbms_stats.gather_table_stats(
ownname=> 'owner name',
tablename=> 'table name',
estimate_percent=> DBMS_STATS.AUTO_SAMPLE_SIZE,
cascade=> DBMS_STATS.AUTO_CASCADE,
degree=> null,
no_invalidate=> DBMS_STATS.AUTO_INVALIDATE,
granularity=> 'AUTO',
method_opt=> 'FOR ALL COLUMNS SIZE AUTO');
```

You can also update statistics at the database or indexes level depending on your requirements.

Updating Statistics For a SQL Server Database

The following example shows how to update statistics for a SQL Server database at the table level. Replace *table name* and *index name* with the appropriate table and index names.

```
UPDATE STATISTICS ON <table name> [ . <index name> ]
WITH FULLSCAN {, NORECOMPUTE }
```

Introduction

This chapter describes the following performance optimization items:

- “JVM Memory and Tuning Guidelines” on page 391 describes general guidelines for Java Virtual Machine (JVM) performance tuning. You should be familiar with your JVM and servlet container environment to apply these guidelines.
- “Log Analyzer Tool” on page 394 describes an open source log analysis tool, Log Analyzer.

JVM Memory and Tuning Guidelines

When you encounter memory-related issues, adjusting the JVM memory settings can get you back to a sane, working environment. This section presents guidelines for JVM memory settings and performance tuning. You should be familiar with your JVM and servlet container environment to apply these guidelines.

Adjusting JVM Memory Settings

In general, you should allocate as much memory as possible to the JVM running your application server. You can do this by setting the JVM memory configuration as follows:

- `-Xmx` should be between 80% and 100% of the machine's physical memory. If you set `-Xmx` too small, the application server may fail with an `OutOfMemory` error. If you set `-Xmx` too large, the memory's footprint is larger and you run the risk of the Java's heap being swapped out, causing other performance problems.
- `-Xms` should be approximately half of the `-Xmx` setting. If you have historical data about your application server's stable memory-usage point, then set `-Xms` to be around that value.

Another option is to set `-Xms` to the memory-usage value observed at the end of `InitServlet`. This will ensure that, at a minimum, DEBS initialization will complete with as little garbage collection as possible. To get the memory usage value, perform the following steps:

- a. Set `-Xms` to be the same as `-Xmx`
- b. Start up your Visual Modeler deployment and wait until initialization is complete
- c. Access your e-commerce site's home page
- d. Open the **debs.log** file in a text editor and examine the log entry that is similar to the following:

```
2003.03.18 ... END Request ... Mem=129380744/388726784/391291344 ...
```

- e. The first number after **Mem=** is the current memory usage after initialization. Set `-Xms` to that number: in the above example, use the value `-Xms128m`.
- `-XX:MaxPermSize` controls the allocation size for system-like reflective objects such as `Class` and `Method`. Its recommended initial value is `128m`.

For Web applications, the allocated space fills up quickly because the `*.jsp` files are converted into `*.java` files, then into `*.class` files which are loaded into the memory space specified by `-XX:MaxPermSize`. Starting with Java version 1.4.2, you can use `-XX:+PrintGCDetails` to monitor the details of this space, named permanent generation.

Be careful not to make memory-related changes that might contradict what the application server currently supports. DEBS needs to co-exist with the application server in the same VM so when in doubt, double-check the application server documentation or contact your application server's Support organization. For example, suppose that the current application server documentation states that the JVM setting `-server` is not supported. In that case, don't set `-server`.

As a last troubleshooting resort, start the VM with no additional arguments and incrementally add one argument, restart, observe the results, then add one more, continuing until you get good results.

Additional Performance Tuning

Additional performance tuning can be done around Java garbage collection activities and by adjusting memory settings for other areas, such as for threads, JVM stacks, or native structures or code. Use the Log Analyzer tool or check the **debs.log** file directly to make observations and determine performance problem areas.

Tracing Garbage Collection Activities

If you observe unexplained pausing, then it is possible that the VM is being paused for a full garbage collection. To confirm that this is what is happening, use the JVM setting `-verbose:gc` to enable recording of garbage collection events in the **debs.log**. Garbage collection events are of the following types:

```
[GC 325816K->83372K(776768K), 0.2454258 secs]
[Full GC 267628K->83769K(776768K), 1.8479984 secs]
```

A minor collection should be less than half a second. A major garbage collection should be less than three seconds. Anything more than three seconds indicates an out-of-range condition and should be looked into.

Other garbage collection trace settings you might want to look into are:

- The JVM `-server` setting: this setting adjusts some initial Java heap settings so that they are more appropriate for a server environment. Set the `-server` value unless your application server does not support it.

There is a known problem with the `-server` setting related to a bug in JIT (just-in-time) compilation which causes the value used by the data service to change unexpectedly. The result is that DEBS will fail to initialize

(InitServlet fails). Contact Sterling Commerce, Inc. or send email to ordercaptureapps_support@stercomm.com to learn how to disable JIT compilation for certain methods.

Some application servers recommend using the VM setting `-server`. In particular, the value of `-XX:NewRatio` for `-server` is 2 (the default value for the `-client` setting is 8). For more information about the `-server` and `-client` settings, see the Sun documentation at the following URL's:

http://java.sun.com/docs/books/performance/1st_edition/html/JAppHotspot.fm.html#998292

http://java.sun.com/docs/books/performance/1st_edition/html/JAppHotspot.fm.html#998359

- The `-Xincgc` setting: this setting enables incremental garbage collection. Setting `-Xincgc` reduces large pauses due to full garbage collection. When you use this setting, bear in mind that you are shifting the time spent to perform one major collection to several minor collections. There is an overhead cost associated with this shift, usually around 10%.
- If you are getting `OutOfMemoryError` messages, the you should first increase the value of `-Xmx`, ensuring that `-Xmx` is no more than the value of the machine's physical memory. If it appears that you are getting `OutOfMemoryError` messages when the current heap usage (where new objects are allocated) is nowhere near the value of `-Xmx`, then there is a possibility that other areas of memory allocation are exhausted. Examine the Log Analyzer report and check the following possible areas:
 - Due to Classes: try setting `-XX:MaxPermSize=128m`
 - Due to Threads: try adjusting the stack using `-Xss=512k`
 - Due to JVM Stacks: try adjusting the stack using `-Xss=512k`
 - Due to Native data structures: try adjusting the OS swap size
 - Due to Native codes: try adjusting the OS swap size

Log Analyzer Tool

The Log Analyzer is an open source tool that can help with your analysis of the Visual Modeler **debs.log** entries. The tool provides a view of key performance indicators: threads, memory, requests, and sessions, as well as response times sorted by user and request type.

Using the Log Analyzer as part of a daily routine of monitoring your deployment provides these advantages:

- Daily log analyzer reports add reliability and stability to your deployment. The generated data can provide an early warning about potential problems, making it possible to prevent outages. For example, using the daily log analyzer reports, you can pro-actively plan to re-start an application server when it reaches near-maximum memory usage.
- Daily log analyzer reports provide the basis for troubleshooting a current problem. By examining the reports, you can determine when the problem started and correlate it with events such as an OS upgrade.
- Daily log analyzer reports provide a focal point for making incremental improvements. By reviewing the log analyzer report daily, you can generate a to-do list to plan when to restart your application server, clean up any exception lists, track down hanging threads, or to provide feedback to developers about long-running requests or requests that are using substantial resources, such as returning large rows from a database.

Contact Sterling Commerce, Inc. to obtain the Log Analyzer tool. The Log Analyzer is a **.jar** file that can be saved and unjarred in any convenient location. The Log Analyzer expects that the format of DEBS log entries is similar to the following:

```
<YYYY.MM.DD HH:MM:SS:mss ThreadName:LogLevel:LogTag:messages>
```

For example:

```
2006.10.12 06:00:00:171 Env/http-8580-Processor48:INFO:WrappingFil-  
ter ...
```

Since the process of analyzing a log file can be memory-intensive, specify as much JVM memory as possible to avoid `OutOfMemoryError` messages. For example, start the Log Analyzer as follows:

```
java -Xms256m -Xmx512m -jar logAnalyzer-1.1.1-SNAPSHOT-app.jar
```

The following screen displays:

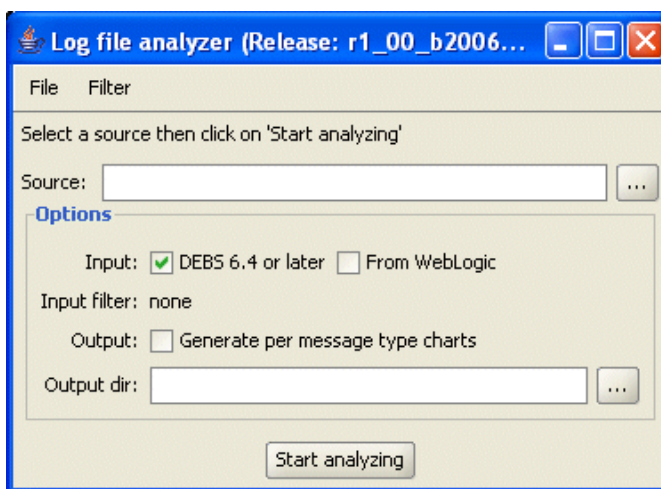


FIGURE 14. Log Analyzer Initial Screen

Enter the following information:

- Source: enter the full pathname of the location of a DEBS log file, or directory containing multiple DEBS log files.
- Input:
 - DEBS 6.4 or later is automatically checked. If you are analyzing a log file from a pre-Release 6.4 Visual Modeler, then uncheck this checkbox.
 - From WebLogic: click this checkbox to indicate that the log files are generated from a BEA WebLogic application server.
- Output: click the Output checkbox to generate a response-time chart grouped by message type.
- Output dir: enter the full pathname of the directory to contain the report output.

Click Start analyzing to start the log analysis process. The Log Analyzer displays messages as it progresses, then places the log analysis output in the specified output directory when it finishes.

Setting Up Log Analyzer Daily Reports

This section describes a procedure for automating daily log analyzer report generation. The procedure described here uses Ant, since Ant is portable, well-used, and has good documentation. Ant is available from <http://ant.apache.org>.

The goals of this procedure are to:

- Set up a cron job to run reports nightly, and organize output by date (year/month/day) to ease navigation.
- Compress log files when possible to save space.
- Set up the automation in a way that is easy to duplicate so that log files from multiple deployments can be hosted from a single log server.

To automate log analyzer report generation, you need:

- Java and Ant
- Read access to the DEBS log files
- Write access to the report output directory, `<out.dir>`. The contents of `<out.dir>` are accessible via a Web server.

Daily Reports Workflow

The following describes the general workflow for automating log analyzer daily reports.

1. DEBS generates log files to the application server or servlet container **logs** directory.

For example, the **logs** directory in a Tomcat deployment is `<tomcat-home>\logs`.

The log file is named **debs.log.n**, where *n* is a number. For example, `debs.log.1`, `debs.log.2`, and so on.

2. Set up a cron job to run daily (perhaps very early in the morning) to concatenate all the log files from the log directory into a temp file.
3. From the temp file, extract yesterday's log entries into the log analyzer output directory using the directory naming pattern `year/month/day/log.suffix`.
4. The **year/month/day/log.suffix** file is further compressed using **gzip** to save space.
5. Start the log analyzer to parse the **year/month/day/log.suffix.gz** file and generate the report to the **year/month/day/html/** directory.

Setting Up the Daily Reports

1. If you have not already done so, contact Sterling Commerce, Inc. to obtain the following files:
 - The Log Analyzer **.jar** file
 - **logAnalyzer-daily.xml**
 - **logAnalyzer-daily.properties**
2. Save the Log Analyzer files to a temporary location.
3. See “Configuration” on page 400 for information about configuration values.
4. Use the following command to run the daily log analyzer report:

```
ant -Dproperties.file.name="logAnalyzer-daily.properties" -f
logAnalyzer-daily.xml
```

5. Examine the output. The location is similar to the following:

```
sites/default/app-server/logAnalyzer-out.d/dailySplit/YYYY/MM/DD/
html/index.html
```

Recommended directory layout

The following figure illustrates the recommended log analyzer directory layout. This layout is especially recommended if you plan to host log files from multiple sites.

```
# where to keep the log analyzer jar file
bin/
  logAnalyzer-1.1.1-SNAPSHOT-app.jar

# ant script
logAnalyzer-daily.xml

# sites data
sites/
  site1/
  ...
  site2/
  ...
  siten/
    logAnalyzer-daily.properties
    app-server/
      logs/
        debs.log
        debs.log.1
        debs.log.2
```

FIGURE 15. Recommended Log Analyzer Directory Structure

Site information is kept under the **sites** directory, which contains a directory for each site. The site directory name can be any unique string; the example above uses *siten*, where *n* is a number: *site1*, *site2*, and so on.

Each site directory contains a **logAnalyzer-daily.properties** file that contains that site's specific settings.

Each site's log files are kept in the **siten/app-server/logs/** directory.

The **sites** directory is read-only. Output is written to the **siten/app-server/logAnalyzer-out.d** directory.

Using the above layout, you can start a cron job with just the site name. For example, for a site named *bbfb-01*:

```
# Tell Ant to set the site.name and use a build script name:
# logAnalyzer-daily.xml
ant -Dsite.name=bbfb-01 -f logAnalyzer-daily.xml
```

If you rename **logAnalyzer-daily.xml** to **build.xml**, then you can then skip the **-f logAnalyzer-daily.xml** argument. For example, for a site named *bbfb-01*:

```
ant -Dsite.name=bbfb-01
```

Configuration

Deployment-specific settings are set in a property file. The default property file is **sites/\${site.name}/logAnalyzer-daily.properties**. You can also set the property file name at the command line as follows:

```
ant -Dproperties.file.name="path_to_file.properties" ...
```

The following lists the configuration properties in the **logAnalyzer-daily.properties** file.

- **log.dir**: the full path to the location of the directory containing the DEBS log files. For example:

```
# default is ./logs  
log.dir=/home/hle/tomcat/logs
```
- **out.dir**: where to write the generated reports. For example:

```
# default is logAnalyzer-out.d  
out.dir=/home/hle/public_html/logAnalyzer-out.d
```
- **logAnalyzer.jar**: the location of the logAnalyzer **.jar** file. For example:

```
# default is ./logAnalyzer-1.1.1-SNAPSHOT-app.jar  
logAnalyzer.jar=target/logAnalyzer-1.1.1-SNAPSHOT-app.jar
```
- **is.weblogic**: true if the log files was generated by WebLogic. For example:

```
# default is false  
is.weblogic=true
```
- **genChart.perMessageType**: false to skip messageType charts generation. For example:

```
# default is true  
genChart.perMessageType=false
```
- **log.prefix**: the DEBS log prefix. You rarely have to change this. For example:

```
# default is debs.log  
log.prefix=Midwest.log
```
- **target.date.offset**: Auto-set the target.date. The default is 1, which means yesterday. For example, set target.date.offset to 7 to extract log files for a week ago:

```
# default is yesterday: 1  
target.date.offset=7
```


- `target.date`: Limit processing to log entries for this day. The most likely usage for this setting is to manually re-generate an old set of log files. For example:

```
# default is yesterday (auto-evaluated)
target.date=2006/07/24
```

Part 4: *Administration*

The chapters in this section of the guide provide information about administering the Visual Modeler.

This guide provides a comprehensive manual for administering the Visual Modeler. This chapter covers the following topics:

- "Using Storefronts" on page 406
- "Users, Roles, and Functions" on page 409
- "Configuring the Visual Modeler" on page 415

Terminology

Two types of users access the Visual Modeler:

- *enterprise users* manage enterprise data such as products, price lists, and partner profiles. Enterprise users belong either to the tenant enterprise or to the storefront partner of a storefront.
- *customer users* buy products from the enterprise. Customer users belong to customer partners of the tenant enterprise or to customer partners of a storefront.

The Visual Modeler supports creating storefronts. A storefront is a fully functional e-commerce site that is a child enterprise of the main (tenant) enterprise. The users of a storefront partner are the enterprise users for that storefront. In general, employees of the storefront organization log in as storefront enterprise users.

The tenant enterprise and its storefront partners are enterprises within the Visual Modeler.

Using Storefronts

Some of your selling partners may not have the infrastructure to maintain their own e-commerce Web site. You can create storefronts for your partners. The storefronts provide a complete e-commerce environment within which your partner can do almost all of the same things that you can do within the enterprise. Storefronts have their own URLs, look-and-feel, administrator users, customers, and so on. Storefront administrators manage their customers and partners, price lists, promotions, orders, define and administer customer segments, access customer service functions, and so on, just as tenant enterprise administrators can. Storefront product data can be shared with the enterprise product catalog and orders placed at the enterprise can be brokered from the enterprise to child storefronts.

Each storefront is created with a "storefront partner": this is the partner that represents the enterprise running the storefront. For example, suppose you want to create a storefront for an organization called Anderel. When you create the storefront, you provide details for the Anderel partner profile within the storefront. Anderel employees can log in as users of this storefront partner: when they do so, they act as enterprise administrators within their storefront. We refer to these users as storefront administrators.

Storefront Administrator Tasks

Administrators of each storefront perform administration tasks to manage their sites. These tasks correspond to the equivalent enterprise administration tasks and so the chapters in this guide that cover general enterprise administration are also applicable to storefront administrators.

Storefront administrators can perform nearly all of the same tasks for their storefront that tenant administrators can perform for the enterprise, including:

- Managing business rules for their storefront
- Performing user administration for their storefront
- Creating and managing storefront partners for their storefront
- Setting prices for their storefront
- Managing system properties for their storefront

- Assigning and unassigning feature types to feature categories that they manage for their storefront
- Assigning and unassigning features to the products that they manage for their storefront
- Defining customer segments for marketing campaigns for their storefront
- Defining acceptable payment methods for their storefront such as credit cards and gift cards

Storefront administrators cannot modify parts of the product catalog or perform catalog administration tasks that affect the entire enterprise. This means that they cannot perform the following tasks:

- Managing features with Sterling Product Manager
- Setting up guided selling with Sterling Advisor
- Performing product import/export
- Performing search administration
- Creating or modifying models using the Sterling Visual Modeler

Storefront Hierarchy

Storefront administrators at any level can create child storefronts beneath them. Over time, a hierarchy of storefronts can develop: the tenant storefront, the storefronts that tenant administrators create, the storefronts that storefront administrators of these storefronts create, and so on. With the exception of the tenant storefront, each storefront has one *parent* storefront, and may have zero or more child storefronts.

- When we refer to the ancestors of a storefront, we mean the parent storefront of the storefront, the parent of that parent, and so on up the hierarchy up to and including the tenant storefront.
- When we refer to the descendants of a storefront, we mean the child storefronts of the storefront, the child storefronts of these children, and so on.

Skins

A skin is a way to determine the look-and-feel of a storefront. The skin comprises a logo (a GIF file), and a cascading stylesheet. Together, these determine some aspects of the user experience as users access the storefront. One skin is created when an enterprise administrator creates the storefront: they specify the string for

the URL and this becomes part of the URL that is used when users access the storefront.

Users access a storefront using a URL such as the following:

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

The last component of the path (“matrix” in this case) is used to determine which skin of which storefront the user is accessing.

A storefront administrator can create more than one skin for their storefront: they must specify the URL string, and optionally a GIF image and cascading stylesheet, for each skin they want to create. For example, the enterprise administrator creates a storefront for Anderel, and they specify the URL string as “anderel”. To begin with, Anderel storefront administrators and end-users will access the Anderel storefront using a URL that looks like this:

```
http://server:port/Sterling/en/US/enterpriseMgr/anderel
```

If an Anderel storefront administrator logs in, and creates a new skin with the URL string “anderelStore” for their storefront, then both storefront administrators and end-users can access the Anderel storefront using:

```
http://server:port/Sterling/en/US/enterpriseMgr/anderelStore
```

Depending on the differences between the cascading stylesheets used for the two skins, users will experience a different look-and-feel depending on which of the two skins they use to enter the Anderel storefront.

Storefront Data

In general, any data that is created within a storefront (either by storefront administrators or by storefront end-users) is separate from data of the enterprise storefront. In particular, data created within a storefront cannot be seen from any other storefront. However, note that the following data is “shared” between the enterprise and storefronts:

- **Product data:** Enterprise product data is visible as read-only data to storefront administrators. Storefront administrators can view enterprise product details and add enterprise products to storefront price lists. Storefront end-users can buy enterprise products provided that they meet

the standard access criteria (using product entitlements and price lists defined at the storefront level).

Enterprise administrators can open a product category to one or more storefront partners immediately below them. When they do so, storefront administrators who have been granted access can create products and child product categories within this product category.

- Products created will be visible as read-only to storefront administrators of any ancestor storefronts, and will be visible as read-only to storefront administrators of any descendant storefronts.
- Product categories created will be visible as read-only to the storefront administrators of any ancestor storefronts. The storefront administrators at this level can open the product category to child storefront partners immediately below them in the storefront hierarchy.

This same principle is applied to product data that child storefronts of a storefront create. At any level within a hierarchy of storefronts, storefront administrators have full access to product data created at their level, and they have read-only access to product data created at ancestor and descendant storefronts.

- Enterprise administrators can see storefront price lists. This is so they can assign storefront price lists to enterprise partners for the purposes of supplier order-brokering.

Storefront Partners

Storefronts are created by an administrator of either the enterprise or an existing storefront. When an enterprise administrator creates a storefront, they provide profile information for the "storefront partner" and a URL that will be the access point for all users entering the storefront.

Users, Roles, and Functions

Users perform functions within the Visual Modeler. To perform their functions, users must have appropriate access privileges. First, assign the functions within your organization to administer the various parts of your e-commerce site. Next, create the users that perform these functions and assign appropriate access privileges to those users.

The tasks for this purpose are presented in CHAPTER 34, "User Administration".

Organizational Functions

Assign the following functions in your organization:

- *Accounts Receivable Representative*: Manages invoices for the e-commerce site.
- *Business Rules Manager*: Controls the business rules for the e-commerce site.
- *Channel Administrator*: Creates and maintains the profiles for each partner and creates a partner administrator for each profile. The partner administrator is an employee of a partner who is responsible for creating and maintaining their own partner users.
- *Commerce Administrator*: Monitors all cart activity at the e-commerce site or Sterling Partner.com partner site.
- *Customer Service Representative (CSR)*: Creates and updates orders on behalf of customers, monitors orders, and monitors any product return requests. Typically, products returns have internal rules that guide whether to approve or reject a return. When a decision must be made manually, the CSR has the authority to make that decision.
- *Enterprise Lead Administrator*: Creates and assigns leads to one or more partners. The enterprise lead administrator also closes the lead.
- *Promotion Administrator*: Manages promotions.
- *Product Administrator*: Manages all the products at the enterprise site: sets the correct prices for the products and associates them with the appropriate partners.
- *Sales Manager*: Manages the sales representatives and delegates leads to them.
- *Sales Representative*: Handles leads that are delegated to them.
- *System Administrator*: Maintains the system configuration using the System Administration module.
- *User Administrator*: Creates and maintains all the users at the e-commerce site.

Creating Users

The *user administrator* is the person at the Visual Modeler installation responsible for adding users to the system and giving them access to the areas appropriate for

them to perform their function. In general, user administrators do not have any privileges associated with partners. In particular, they cannot create partner users. (The only enterprise employees who can create partner users is the *channel administrator*.)

When you create users, you must assign them a username and password. The username you assign must be unique. Each username is checked for uniqueness when the user is created: if the username is already in use, then the user administrator must choose a different username. When a user is deleted from the Visual Modeler, their username is not: once a username is in use, it can never be reused.

As you create users, you must also assign access privileges by assigning one or more functions.

Assigning Functions

In the Visual Modeler, *entitlement functions and roles* explicitly define the access that users have to business objects and the functions they can perform such as updating users or creating price lists. These functions and roles are listed in the **Entitlements.xml** configuration file which is read by the Visual Modeler server on startup. The file comes with several entitlement functions and roles pre-defined (see "Pre-defined Functions" on page 412), but you can customize access by editing this configuration file to create more roles and edit the privileges of existing roles.

Roles are grouped into functions: functions are intended to correspond quite closely to the business functions within an organization: finance, sales, and so on. Each function has a label: the label displays in the browser when you perform user administration.

It is important to distinguish these *entitlement functions* from the *organizational functions* described earlier. Any person in your organization may have one or more organizational functions that they perform to complete their job responsibilities: system administrator, product manager, sales manager, and so on. These may or may not correspond to the entitlement functions defined in the Visual Modeler.

Consequently, the entitlement functions defined in your implementation of the Visual Modeler may serve as "umbrella" roles that cover more than one organizational function. For example, to provide them with the proper access, you may need to assign the same entitlement function to the *channel administrator* and the *user administrator*. At implementation time, your system integrators determine appropriate groupings of organizational functions into entitlements functions. These entitlement functions are defined in the **Entitlements.xml** configuration file.

However, note that only those roles present in the access policies and access control lists (ACLs) or in the **Entitlements.xml** file have any effect on the privileges users have.

Pre-defined Functions

The **Entitlements.xml** configuration file that is implemented with the Visual Modeler comes with the following pre-defined functions:

TABLE 22. Pre-defined Enterprise Functions

Function/Label	Description of Access
EnterpriseProgramManagement/ Program Management	Includes Pricing, Product, Model, Coupons, Advisor, and Promotion Management. Also includes reporting, job scheduling, and editing of system properties and business rules.
EnterpriseFinancials/Financials	Includes the ability to remove Credit Holds from Partners, Users and Orders. Also includes the ability to view and edit invoices.
EnterpriseCommerce/Commerce	Includes the ability to create carts, place orders, create quotes on behalf of customers.
EnterpriseSales/Sales	Includes the ability to work with opportunities and proposals as well as being able to create carts, quotes, and orders.
EnterpriseSalesExecutive/ Sales Executive	Adds the ability to act as sales manager to the EnterpriseSales function. Sales managers assign opportunities to other users and can also work opportunities themselves.
EnterpriseLeadAdministratorSales/ Lead Administration	Can manage leads for the enterprise.
EnterpriseBasicAdministration/ Basic Profile Maintenance	Performs limited user and profile administration at or below their node. Can only assign functions to other users that they have.
EnterpriseAdministration/ Profile Administration	Performs full user and profile administration at or below their node. You must ensure that at least one enterprise user has the EnterpriseAdministration function.

For partners, the following table summarizes their functions:

TABLE 23. Pre-defined Partner Functions

Function/Label	Description of Access
PartnerProgramManagement/ Program Management	Includes Pricing, Product, Promotion Management. Also includes creation of email templates, SKU and availability management.
DirectFinancials/Financials	The ability to view and edit invoices.
DirectCommerce/Commerce	Includes the ability to create carts, place orders and create quotes.
DirectCommerceExecutive/ Commerce Executive	Includes the ability to create carts, place orders, perform order approvals, and create quotes.
Commerce	Includes the ability to create and transfer carts.
DirectSales/Sales	Includes the ability to work with leads and opportunities apart from being able to create carts, quotes, and orders.
DirectSalesExecutive/Sales Executive	Includes the ability to work with opportunities apart from being able to create carts, quotes, and orders.
Sales	Includes the ability to work with leads and opportunities as well as being able to create and transfer carts.
Sales Executive	Includes the ability to work with leads and opportunities apart from being able to create and transfer carts.
PartnerBasicAdministration/ Basic Profile Maintenance	Performs limited user and profile administration at or below the node.
PartnerAdministration/ Profile Administration	Performs full user and profile administration at or below the node.
StorefrontCustomerBasicAdministration /Basic Profile Maintenance	Performs limited user and profile administration at the node. Can only assign functions to other users that they have.
StorefrontCustomerAdministration/ Profile Administration	Performs full user and profile administration at the node.

Managers

Users can be marked as managers. Managers are entitled to navigate to child nodes of their node to perform the same tasks as they can at their own node. They can also view and modify the activity of other users at their node. For example, an enterprise user with the Commerce function and marked as a manager can navigate to a child node in the enterprise hierarchy and view the orders created by EnterpriseCommerce users at the child node.

An enterprise user who is a manager can access all the accounts assigned to their enterprise node and nodes below this node. That is, managers do not have to explicitly draw accounts from the pool of accounts assigned to their node: they can work on any account assigned to their node.

User Statuses

Every user has a status. The status of the user and the profile status of their partner determine what the partner user can do.

The following are the possible user statuses:

- **Open:** no restrictions on the activities of a user.
- **On Credit Hold:** users can log in, but they cannot place orders on account. They can still place orders on credit cards.
- **On Hold:** users can log in, but they cannot place orders.
- **Closed:** users cannot log in.

When you set a user status to closed, or if their effective status becomes closed because you close their partner, this does not affect the carts, orders, returns, and other objects that the user has been working on. These remain in their current status until another partner user or enterprise user changes them.

Note that only partner users can be assigned the On Credit Hold and On Hold statuses. Only enterprise users with the Financials function can set partner users on On Credit Hold status. An enterprise user can re-open an On Hold partner user, but only enterprise users with the Financials function can re-open On Credit Hold partner users.

Inheriting Status

Each user belongs to a partner, and the effective status of a user is inferred from their user status and the effective status of their partner. For example, suppose that User 1 is a partner user of Partner B and that the effective status of Partner B is Open. If the status of User 1 is Open, there is no restriction on the activities of the

user. If you change the profile status of Partner B to On Hold, then even though you have not changed the status of User 1, their effective status changes to On Hold, and so they can log in, but they cannot place orders.

Suppose that Partner B is the child of Partner A, and that the status of Partner B is Open. If you set the status of Partner A to On Credit Hold, then even though you have not changed the status of Partner B, the effective status of Partner B is inherited to be On Credit Hold. Consequently, the effective status of Partner B users is On Hold, and so they cannot place orders.

The status of a partner overrides the status of a user if the partner status is more restrictive: In the example above, if the status of Partner B is set to Closed, then the effective status of User 1 is also Closed, irrespective of the status of User 1.

User Preferences

Partner users have *user preferences*: these are properties that influence the user experience as they use the Visual Modeler. In general, users will manage their own preferences through their user profile, but it is possible for their partner administrators to manage preferences for a user.

User preferences include:

- **Cart view:** offers a choice between a simple and a complex view of each shopping cart.
- **Cart mode:** offers a choice between a single cart or multiple carts. If a multiple cart user switches his preference to single cart, all his existing carts are hidden.

User preferences are set up as part of the implementation of the Visual Modeler, and so your installation may have more user preferences.

Configuring the Visual Modeler

The properties of the Visual Modeler are defined in a set of configuration files and in the Knowledgebase. When the servlet container is started, the Visual Modeler loads and accesses the configuration files in order to determine Visual Modeler settings.

System properties for each enterprise (tenant and storefronts) are managed in the CMGT_SYS_PROPERTIES database table: for each storefront default values are populated when the storefront is first created. When a system property is changed through the Web UI, then the new value is stored in the table.

After implementation, you can modify the settings using the System Administration page and the Business Rules Manager.

Site System Administration

Site system administration is performed by site system administrators. Site system administrators access the Visual Modeler through the site administration URL, which is of the form:

```
http://server:port/Sterling/en/US/enterpriseMgr/admin
```

The default login ID is admin, password admin. Site system administrators manage properties that are common to all storefronts, such as logging. See CHAPTER 42, "Site System Administration" for more information.

Enterprise System Administration

Enterprise system administration manages enterprise-level (either tenant or storefront) settings in the Visual Modeler. For example, you can specify the email settings in the System Properties page.

You can only modify system configuration settings if you have the appropriate access role. In the reference implementation provided with the Visual Modeler, only users with the Program Management function can access the System Administration pages. These users access the Visual Modeler through an enterprise administration URL which is of the form:

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

Enterprise system administrators manage properties that are specific to their enterprise: changes that they make do not affect other enterprises. See CHAPTER 39, "Enterprise System Administration" for more information.

Business Rules

Business Rules define the behavior of the Visual Modeler. For example, this includes punchin and punchout specifications, the behavior of imports and exports, cluster configuration, and other product management specifications. These business rules are specified in the properties files of the Visual Modeler, and are managed through the business rule administration interface. See CHAPTER 40, "Business Rules Administration" for the tasks involved in Business Rules management.

Job Scheduling

You can create cron jobs for different activities in the Visual Modeler. There are storefront-level and enterprise-level cron jobs. Each storefront manages its own set

of cron jobs. Only enterprise administrators can manage the enterprise level cron jobs. See CHAPTER 41, "Job Scheduling Administration" for more details.

You can schedule system-level or application-level cron jobs to run according to a specific frequency between a certain date and time range.

<p>Note: When a job is run using the Job Scheduler, its execution status is recorded. On occasion a job may execute successfully but the status is recorded as "Timed Out".</p>
--

This chapter covers tasks performed to manage users in the Visual Modeler: See "Managing Users" on page 419 for the tasks performed by Enterprise employees or the employees of a Sterling Partner.com partner.

"Users, Roles, and Functions" on page 409 contains an overview of user administration in the Visual Modeler.

This chapter covers the following topics:

- "Managing Users" on page 419
- "Defining Functions and Roles" on page 422

Managing Users

The *user administrator* (an enterprise employee with enterprise user management responsibilities) performs the following tasks:

- "To Create a New Enterprise User" on page 420
- "To Modify an Enterprise User Profile" on page 421
- "To Delete an Enterprise User" on page 421
- "To Search for an Enterprise User" on page 421
- "To Search for Any User" on page 422

- "Defining Functions and Roles" on page 422

Note that enterprise profile administrators can also create partner profiles and partner users for partners. See "To Create a New Partner User" on page 437 for more information on creating partner users.

To Create a New Enterprise User

1. Click **System Users** in the System Administration panel on the Visual Modeler enterprise home page. This displays the User List page.
2. On the User List page, click **Create User**. If this button is not visible, then you do not have the access privileges to create a new user.
3. On the User Detail page, enter the details of this new user. Note the following:
 - Username: This username must be unique throughout the Visual Modeler.
 - Password: Use letters and numbers from the keyboard with no spaces or other punctuation.
 - User Functions: Select those functions that this user will perform by checking the appropriate check boxes. The list of functions displayed here is determined at implementation time.

<p>Attention: Do not select the ERPAdministrator user type for standard users. Users of this user type cannot log in through the Web user interface.</p>

- Preferred Locale: Select the preferred locale which will apply when the user logs in. The drop-down list displays the names for the supported locales.
4. Enter any other pertinent details.
 5. Click **Save**.

Once you have saved the basic information for a new user, the User Detail page is displayed with new tabs.

6. You can update information on the Info tab and click **Save**. In particular, you can now set a maximum on the number of accounts that can be assigned to this user.
7. You can manage the assignment of accounts to this user.
8. You can also make notes regarding this user by clicking the Notes tab.

To Modify an Enterprise User Profile

You can change user profile information for another enterprise user or partner user if you have the right level of entitlement access to the user.

Attention: Do not change the username or password of the Anonymous User user.
--

1. Click **System Users** in the System Administration panel on the Visual Modeler enterprise home page. This displays the User List page.
2. On the User List page, click the link to the user whose profile you wish to modify. This displays the User Detail page. If you cannot see the user whose details you want to update, then you can search for the user.
3. On the User Detail page, modify user details as appropriate.
4. Click **Save All and Return to List**.

To Delete an Enterprise User

1. Click **System Users** in the System Administration panel on the Visual Modeler enterprise home page.
2. On the User List page, click **Delete (X)** in the Actions column next to the user you wish to delete.

The user is deleted from the Visual Modeler. However, note that the username that belonged to this user is still present in the system. No new user can re-use this username.

To Search for an Enterprise User

1. Click **System Users** in the System Administration panel on the Visual Modeler enterprise home page.
2. Select one of Username, First Name, Last Name, from the drop-down list, and enter the full or partial string for the search. You can use "*" as a wild card character.

For example, if you select First Name and enter "An*", then you will find all enterprise users whose first name begins with "An" such as Andrew and Anne.

3. Click **Go**.
4. You can click **Advanced Search** to perform a more detailed search.

To Search for Any User

You can view the user details for any user in the Visual Modeler. If you have the appropriate function, then you can also modify user details or delete them from the Visual Modeler. Note that in general, the administration of partner users should be left to partner administrators for each partner.

You can use the Search for User by Name panel to perform a quick search for a user, or you can use the advanced search capabilities as follows:

1. Click **Advanced Search** in the Search for User by Name panel.
2. Enter search criteria and click **Submit**.

The search results page displays the users that match your search criteria.

3. Click the link to the user that you were looking for.

Defining Functions and Roles

Functions and roles are defined using the **Entitlements.xml** file. Defining a function or role requires that you specify the functionality that you wish the function or role to perform.

This chapter describes the tasks required to administer the channel in the Visual Modeler. The enterprise employee who acts as the *channel administrator* can create, modify, and delete partner profile information. See "Profile Administration Tasks" on page 430. The partner employee who acts as partner administrator can maintain the partner profile once it has been created by the channel administrator.

This chapter covers the following tasks:

- "To Search for a Profile" on page 430
- "To Export Profile List Information" on page 431
- "To Create a New Profile" on page 432
- "To Create a Profile as a Child of a Parent Profile" on page 434
- "To Move a Child Profile to Another Parent" on page 435
- "To Create Profile Addresses" on page 435
- "To Delete Profile Addresses" on page 436
- "To Create a New Partner User" on page 437
- "To Move Users Between Levels in a Profile Hierarchy" on page 438
- "To Modify an Existing Profile" on page 440

Profile Detail Page

You maintain partner profile information on the Profile Detail page. The Profile Detail page provides a straightforward means to administer the information you need to work effectively with your partners. The information is organized by tabs that group related information:

The tabs are:

- Info Tab
- Addresses Tab
- Detail Tab
- Business Tab
- Hierarchy Tab
- Commerce Tab
- Assigned To
- Pricelists Tab
- Product Entitlements Tab
- Attributes Tab
- Notes Tab

Info Tab

This tab displays key partner information:

- Profile name: the display name for this profile. Profile names do not have to be unique. However, in several places in the user interface, profiles are listed by profile name. Distinguishing two profiles with the same name in any list of profile names can be confusing. Use a naming convention that ensures that profile names are effectively unique.
- Main telephone: the main telephone number of the partner.
- Main facsimile: the main facsimile number for the partner.
- Profile type: Each profile must be assigned a *type*. The choice of types is determined at the time of the implementation of the Visual Modeler. To make a profile available for selection as a distributor, you must select "Distributor" in the Profile Type drop-down list.

- Profile level: profiles can be assigned to one of several levels, such as Platinum, Gold, or Silver. If your implementation uses such a system, then select the correct level for this partner.
- XML Message version: the XML version is required to send messages to this partner's server.
- Login/Password required (for distributor partners only): if you check this box, a username and password is required to allow the enterprise employee to access the distributor's site.
- Company website address: the main home page for this profile. Although this field is not required, you must provide a Web site address if you wish the partner to be contacted through the Partner Selector function.
- Organization Email: the email address for the company. Although this field is not required, you must provide an Email address if you wish the partner to be contacted through the Partner Selector function.
- Distributors: each partner can have a business relationship with one or more distributors to place orders and obtain price and availability information. For the current profile, select those distributors with whom the partner has a business relationship. The list of profiles whose names are displayed in the drop-down list is determined by those profiles whose whose type is "Distributor".

<p>Note: In previous releases, the list of distributors was used to determine which distributors could be used for price-and-availability requests. In this release, this information is for display only.</p>

- Message URL: This field is required if the partner needs to send or receive Visual Modeler XML messages (such as price and availability checks and cart transfers). This field represents the URL to which messages for this partner are sent to or received from.

If you are creating a profile for a storefront partner, then the entry follows the format:

`http://<servername:port>/Sterling/msg/<partner name URL>`

- Custom field #1, Custom field #2, and so on: these fields can be customized to suit individual information needs.
- Profile Status: this panel enables you to change the status of the profile.

Profile Status
Please enter a reason when changing the status.
Status
Open
Reason

FIGURE 16. Profile Status Panel

- **Accounts:** this panel provides access to the payment accounts used by the partner to manage their program activities.

Accounts
Please enter a reason when changing the account limits.
Currency
U
Credit Limit
\$5000.00
Available Credit
\$5,000.00
Reason
View Account Details
MDF
Co-op %
15
Co-op Account Maximum
2000

FIGURE 17. Accounts Panel

This panel gives you access to the payment accounts for the partner: MDF and Co-op accounts. The Co-op % and Co-op Account Maximum fields are used to calculate how uploaded updates to Co-op accounts are added to the available balances in the accounts.

Addresses Tab

This tab displays the sold-to, ship-to, and bill-to addresses provided by the partner.

Detail Tab

This tab contains business information about the profile.

The Detail tab contains the following fields:

- Organization ID: This ID is used by the enterprise to identify each organization uniquely with whom they do business.
- Year founded: used for information only.
- Revenue: Estimates for this year's and next year's revenues.
- Fiscal year end month: used for information only.
- Number of employees: used for information only.
- Dun and Bradstreet ID: this ID must uniquely identify the partner in the commercial world.
- Services: this sub-tab provides information about the services that this partner offers.

Services	Skills
Delete Duplicate Previous 1 out of 3 Next	
Service name	
Configuration and Installation	
Service description	
Configurations and Installations	
Service level	

FIGURE 18. Services Sub-tab

- Skills: this sub-tab provides information about the skills and skill levels that this partner is assessed to have.

Services	Skills
Delete Duplicate Previous 1 out of 2 Next	
Skill name	
Novell Netware	
Skill description	
Novell Netware description	
Rating	
A	
Rating description	
Excellent	

FIGURE 19. Skills Sub-tab

Business Tab

This tab contains information relating to the business relationship between your enterprise and the current profile. Only enterprise employees who are channel administrators have the authority to modify information on this tab.

The information on this tab is for informational purposes only and has no effect on any other part of the Visual Modeler. The information comprises:

- Product categories: select one or more product categories to show categories of products that the partner may sell.
- Territories: select one or more territories for this partner.
- Customer types: select one or more customer types (vertical markets) for this partner.
- Contracts: there may be several business agreements between your enterprise and this partner. This sub-tab provides basic information about each agreement.

Hierarchy Tab

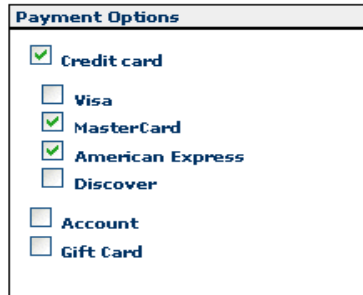
The Hierarchy tab enables you to manage a profile hierarchy. You can use this tab to create a complex organizational structure. For example, you can create management companies, divisions, locations, and departments. Then, by navigating down through the hierarchy of "children", you can create and view "children" within "children" to an infinite number of levels.

Commerce Tab

"Storefront Administration" on page 440The following are the supported type of payment options:

- Credit card
- Gift card
- Account

Support for credit card and gift card payment requires that you set up payment gateways. Before you set up a credit card or gift card payment gateway for a partner, the partner must have a business relationship with a payment processor. This generally involves establishing a merchant ID with the processor and obtaining a merchant key which is used to authenticate the merchant at the payment processor. The key must be stored on the file system accessible by the Visual Modeler.

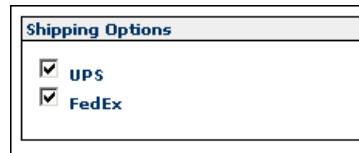


The Payment Options panel is a rectangular box with a title bar at the top. It contains a list of payment methods, each with a checkbox. The checked options are Credit card, MasterCard, and American Express. The unchecked options are Visa, Discover, Account, and Gift Card.

Payment Options	
<input checked="" type="checkbox"/>	Credit card
<input type="checkbox"/>	Visa
<input checked="" type="checkbox"/>	MasterCard
<input checked="" type="checkbox"/>	American Express
<input type="checkbox"/>	Discover
<input type="checkbox"/>	Account
<input type="checkbox"/>	Gift Card

FIGURE 20. Payment Options Panel

You can set up a credit card payment gateway and a gift card payment gateway for your partner profile. This involves selecting a payment processor and providing the merchant details for your partner.



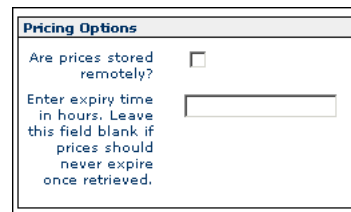
The Shipping Options panel is a rectangular box with a title bar at the top. It contains a list of shipping carriers, each with a checkbox. Both UPS and FedEx are checked.

Shipping Options	
<input checked="" type="checkbox"/>	UPS
<input checked="" type="checkbox"/>	FedEx

FIGURE 21. Shipping Options Panel

Pricing Options

Use this panel to specify if this partner maintains their pricing information remotely. Prices are retrieved using the Message URL specified on the Info tab (see "Info Tab" on page 424).



The Pricing Options panel is a rectangular box with a title bar at the top. It contains a checkbox for "Are prices stored remotely?" which is unchecked. Below it is a text input field for "Enter expiry time in hours. Leave this field blank if prices should never expire once retrieved." The input field is currently empty.

Pricing Options	
Are prices stored remotely?	<input type="checkbox"/>
Enter expiry time in hours. Leave this field blank if prices should never expire once retrieved.	<input type="text"/>

FIGURE 22. Pricing Options Panel

If you check the **Are prices stored remotely?** check box, then you can also specify how long prices may be cached on the Visual Modeler before needing to be retrieved again.

Assigned To

You use this tab to see which enterprise users have been assigned to this partner.

Pricelists Tab

You use the Pricelists tab to assign price lists to this partner.

Product Entitlements Tab

You use the Product Entitlements tab to assign product entitlements to this partner. You can also manage the order in which the product entitlements are evaluated.

Attributes Tab

You can use this tab to view and assign attributes that have been pre-defined as available for assignment to the partner during the partner's creation and profile maintenance.

Notes Tab

You can make notes as you work on partner profiles using the Notes tab.

Profile Administration Tasks

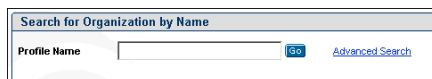
All of the tasks described here are initiated by an enterprise administrator.

To Search for a Profile

You can perform searches on your existing profile to access a given profile.

Note: In the case of a profile hierarchy, you can only search for the profile at the top-level of the hierarchy.

1. To perform a quick search, enter the profile name in the Profile Name text field of the Search for Organization by Name panel, and click **Go**. You can use "*" as a wild card. For example, searching for "Af*" will find "Affine Systems", "AffinityNet", and so on.



The image shows a search interface titled "Search for Organization by Name". It contains a text input field labeled "Profile Name", a blue "Go" button, and a link labeled "Advanced Search".

FIGURE 23. Search for Organization by Name Panel

You can perform a more advanced search by clicking **Advanced Search**. The Profile Search page displays.

2. Click **Search** to view all the profiles, or enter search criteria and click **Search**.

You can use the asterisk (*) in your searches. For example, "Ander*" in the Profile Name field will find any profile whose name begins with "Ander". Likewise, "*erel" will find any profile whose name ends in "erel".

The list of profiles satisfying your search criteria is displayed.

3. In the displayed list, find and click the name of a profile to display the Profile Detail page for that partner.

If the list is too long to efficiently locate the profile or if the profile is not in the list, then you can return to the main search page and attempt a new search.

On the Profile List page, you can also click the check box next to a profile and then do one of the following:

- Export a list of selected profiles. See "To Export Profile List Information" on page 431.
- Click **View Account Activity** to view cart activity for selected partners.

To Export Profile List Information

There may be times when you want to review profile information offline or using an analysis tool such as a spreadsheet. You can export profile information as a text file for this purpose.

1. Using the Search for Organization by Name panel, enter search criteria to help you locate a profile or set of profiles. You can specify criteria to limit the scope of the search as described in "To Search for a Profile" on page 430.

The Profile List page is displayed. It presents all of the profiles that meet your search criteria.

2. Select those profiles whose details you wish to export by checking the check boxes next to each profile.

You can click **Select All** to select all the profiles on the current page.

3. Click **Export List**.

A new browser window is displayed showing the selected profile data in text format.

You can save this file to your machine. When you open this saved file in a spreadsheet application, you must specify that it has been created in a tab-delimited format.

If the file has been generated by a UNIX installation of the Visual Modeler, then the file has UNIX end-of-line characters. This means that some Windows text-editing applications such as Notepad may display the file as one continuous line. You can still open the file as a spreadsheet.

To Create a New Profile

Perform this task to create a profile either as a standalone profile (no child profile, no parent profile) or as the top-level profile in a profile hierarchy. To create a child profile of a parent in a profile hierarchy, use "To Create a Profile as a Child of a Parent Profile" on page 434.

1. Click **Go** in the Search for Organization by Name panel on the Visual Modeler home page.

2. Click **Create Profile** on the Profile List page.

The Profile Detail page displays.

3. At the **Create** tab, enter the pertinent profile information.

See "Info Tab" on page 424 for a description of these fields. At a minimum, you must enter information in the fields marked (*).

<p>Note: Although the Organization website address and the Organization Email address are not required, you must provide these if you wish the partner to be contacted through the Partner Selector function.</p>
--

You must enter at least one address and define the address as Sold-to, Bill-to, or Ship-to by checking the appropriate boxes.

Enter an Organization ID: this should be a unique identifier for the profile.

4. Click **Save**.

Once you have saved the required information for the new profile, then you can continue through the other profile tabs to enter additional information.

5. (Optional) For enterprise nodes, on the **Info** tab, you can specify the maximum number of users of this node who can be assigned to a particular account.
6. (Optional) At the **Addresses** tab, enter additional sold-to, bill-to, and ship-to addresses for the profile.
 - a. Create a new address by clicking **New**, or duplicate an existing address by clicking **Duplicate**.
 - b. Enter the pertinent address information.
 - c. Define the address as Sold-to, Bill-to, or Ship-to by checking the appropriate box.
 - d. Define the address as the default Sold-to, Bill-to, or Ship-to address by checking the appropriate box.
7. (Optional) At the **Detail** tab, enter pertinent information. You can specify what services and skills the profile offers by using the Services and Skill sub-tabs.
At minimum, you must enter information in the fields marked (*).
8. (Optional) At the **Business** tab, enter the product categories, territories, and approved customer types (vertical markets).
9. (Optional) At the **Hierarchy** tab, create any desired profile hierarchy.
See "To Create a Profile as a Child of a Parent Profile" on page 434.
10. (Optional) At the **Logo** tab, upload the logo that will appear on the partner's storefront. Typically, the task of uploading a logo is done by a partner administrator of each partner who is enabled for Sterling Partner.com.
11. Save the information you have entered.
12. If you want to assign price lists to this profile, then click the Pricelists tab.
13. You can add notes about this partner by clicking the **Notes** tab.
You can create partner users for the new partner by clicking **View Partner Users**.
14. You can assign attributes available for assignment to this partner by clicking the **Attributes** tab.

To Create a Profile as a Child of a Parent Profile

Perform this task if you want to create a profile as part of an existing profile hierarchy. Typically, you do this if you are creating an organizational hierarchy for a partner to match its organization into departments or divisions.

1. Click **Organization Lookup** in the Channel Management panel on the Visual Modeler home page.
2. Search for the top-level profile within whose hierarchy you want to create the child.

See "To Search for a Profile" on page 430. The list of profiles satisfying your search criteria is displayed.
3. Find the profile in the search results list, then click their name to be taken to the Profile Detail page.
4. Click the **Hierarchy** tab.
5. Find the profile that you want to be the parent.

Note: Skip this step if you want the child profile to be child to the top-level, parent profile.

- a. Find and click the parent profile in the list of child nodes.
- b. Click **Go To Child**.

This displays the Profile Detail page for the child.
- c. Click the **Hierarchy** tab for the child.
- d. Repeat these steps until you find the appropriate node in the hierarchy.
6. When you find the desired parent profile, click **Create Child**.

This displays the Profile Detail page for the new partner. Notice that certain information (for example, Profile type) is copied from the parent profile.
7. Enter the information for the partner.

See "To Create a New Profile" on page 432.
8. Save the information you entered.
 - Click **Save All** to save the information and remain at the Profile Detail page.

- Click **Save All and View Partner Users** to save the information and view the partner users for this partner. See "To Create a New Partner User" on page 437 for further information about creating partner users.
- Click **Save All and Return to List** to save the information and display the Profile List.

To Move a Child Profile to Another Parent

You can move a child profile to another location in a profile hierarchy. For example, you might want to re-arrange the divisional organization of a profile hierarchy if the partner undergoes a re-organization.

1. Search for the profile that is the parent in the hierarchy.
See "To Search for a Profile" on page 430.
2. In the Profile Detail page, click the **Hierarchy** tab.
3. In the Hierarchy tab, navigate the hierarchy until you find the child that you want to move.
4. Click on the child you want to move.
5. Click **Move Child**.

This displays a list of the other nodes to which you can move the child.

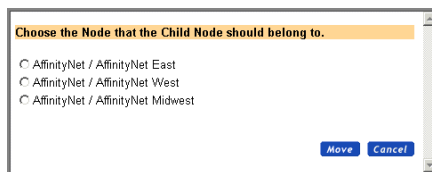


FIGURE 24. Move Child Popup Window

6. Click **Move**.

The child becomes a child of the selected parent.

To Create Profile Addresses

1. Click **Organization Lookup** in the Channel Management panel on the Visual Modeler home page.

2. Search for a profile.

Enter the profile name, or enter search criteria such as the profile type or the first few letters of their name, then click **Search**, or click **Show All** to view all the profiles.

The list of profiles satisfying your search criteria is displayed.

3. Find the profile in the list, then click their name to be taken to the Profile Detail page.
4. Click the **Addresses** tab.
5. Create a new address by clicking **New**, or duplicate an existing address by clicking **Duplicate**.
6. Enter the pertinent address information.
7. Define the address as Sold-to, Bill-to, or Ship-to by checking the appropriate box.

When you check the appropriate box, a check box appears that enables you to define the address as the default.

8. Save the information you entered.
 - Click **Save All** to save the information and remain at the **Addresses** tab.
 - Click **Save All and View Partner Users** to save the information and view the partner users for this profile. See "To Create a New Partner User" on page 437 for further information about creating partner users.
 - Click **Save All and Return to List** to save the information and display the Profile List.

To Delete Profile Addresses

1. Search for the profile who has the address you want to delete.

See "To Search for a Profile" on page 430.
2. Find the profile in the list, then you can click their name to be taken to the Profile Detail page.
3. Click the **Addresses** tab.
4. Find the address you want to delete by clicking **Next** or **Previous**.
5. Click **Delete** to delete the address.
6. Click **Save All**.

To Create a New Partner User

In general, each partner is responsible for managing the partner employees who may log in to the Visual Modeler. These people are known as partner users.

When a profile is created, the channel administrator should create at least one partner user with partner administrator privileges so that this partner administrator may manage their profile and users for that partner: you do this by assigning the user the Profile Administration function.

1. Click **Organization Lookup** in the Channel Management panel on the Visual Modeler home page.
2. Find the profile for whom you want to create the user.
See "To Search for a Profile" on page 430.
3. From the list of search results, click the profile.
4. On the Profile Detail page, click **View Users**.
5. On the User List page, click **Create User**.
6. Enter a username in the **Username** field.
All usernames must comprise standard keyboard characters. Do not use punctuation marks or spaces in a username. Usernames in the Visual Modeler must be unique within a storefront, so your first choice of username may be already taken. If a username is taken, then a dialog box prompts you to try again with a different username.
7. Enter a password for this new user. The system verifies that the same password has been entered in both fields.
8. If this user is to be a partner user with the partner administrator function, then check the **Profile Administration** check box.
9. Select the attributes you want to assign to this user from the drop-down lists in the **Attributes** panel.
10. Enter any other required information (indicated with an asterisk (*)), as well as any optional information you want to enter.
11. Click **Save**.

When the User Detail page is re-displayed, additional tabs are available.

12. Optionally, enter a spending limit and designate one or more approvers if the user exceeds the spending limit.

<p>Note: The fields for spending limits and approvers appear only if this feature has been enabled. This is done with a business rule. See CHAPTER 40, "Business Rules Administration".</p>
--

13. Click **Save**.
14. Click the **Addresses** tab to enter the addresses.
You can create as many addresses as you like for the user.
15. Click the **Preferences** tab to set user preferences for the user.
16. You can assign attributes that are available for assignment to this user by clicking the **Attributes** tab.
17. You can make notes about this user by clicking the **Notes** tab.
18. Contact the partner to let them know that a partner user has been created.

To Move Users Between Levels in a Profile Hierarchy

Perform this task if you want to move one of your partner's users between the partner nodes in their profile hierarchy. In general, you can only move users to nodes to which you have access: typically, this means that you can move users to your node or to nodes below your node.

<p>Note: Moving a user does not move any carts, orders, and so on, associated with the user. When you move a user to another level in the partner hierarchy, notify the administrator for the level. The administrator for the level can recover these lists, orders, and so on.</p>

1. Click **Organization Lookup** in the Channel Management panel on the Visual Modeler home page.
2. Find the partner that contains the user you want to move.
See "To Search for a Profile" on page 430.
3. From the list of search results, click the partner that contains the user.
4. Find the user you want to move.
If the user belongs to the top level in the partner hierarchy, then click **View Partner Users** on the Profile Detail page. This displays the User List page.
If the user belongs to a level below the top level:

- a. Click the **Hierarchy** tab on the Profile Detail Page.
- b. Find and select the level that contains the user you want to move.
- c. Click **Go To Child**.

This displays the Partner Profile Detail Page for that partner. If the user you want belongs to a level below this one, then repeat these steps until you reach the desired level.

- d. Click **View Partner Users** to display the User List Page for that level.
5. In the User List page, find the user you want to move.
6. Click the **Move** icon in the Actions column.

This displays a window with a selection of levels in the profile hierarchy. The levels are displayed as fully-qualified paths. For example, in Figure 25 on page 439, the first selection is AffinityNet East, a division of AffinityNet. The third selection is AffinityNet West - San Jose, a division of AffinityNet West, which is itself a division of AffinityNet.

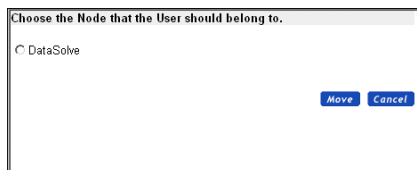


FIGURE 25. Level Selection Window

7. Click the radio button next to the level to which you want to move the user.

If you move a user between two node levels of a profile hierarchy, then the functions assigned to the user before the move are retained.

<p>Note: Moving a user does not move any carts, orders, and so on, associated with the user. When you move a user to another level in the partner hierarchy, notify the administrator for the level. The administrator for the level can recover these lists, orders, and so on.</p>

8. Click **Move**.

The user is moved to the selected level.

After you move a user, you should inform the partner (or the node) administrator so that they can examine and modify the information as necessary. This ensures that

the information is correct for the new location. For example, the addresses (ship-to, bill-to, and so on) might need to be corrected for the new location.

To Modify an Existing Profile

Over time, your relationship with a partner may change and profile information will need to be updated as contacts and addresses change. As channel administrator, your responsibility is to keep profile information up-to-date by modifying the profile.

As a channel administrator, you can create, modify, and delete partner users. However, once a profile administrator has been created, primary responsibility for partner user administration rests with the partner administrators.

1. Search for the profile as described in "To Search for a Profile" on page 430 and click their name to display their Profile Detail page.
2. Enter the revised information in the appropriate fields.
3. As you enter the information, click **Save** to save the information that you have entered so far.

Storefront Administration

When you create a storefront, you create a partner profile that serves as the enterprise partner within the storefront. This partner is the storefront administrator partner and users who belong to this partner are *storefront administrators*.

The storefront that you create using the Visual Modeler is meant only for administration purposes. To ensure that the correct product information is used for defining the models in the Visual Modeler, you must create a storefront with the same Skin ID as the Organization Code of the catalog organization.

To Create a Storefront

1. Click **Go** in the Search for Organization by Name panel on the Visual Modeler home page.
2. Click **Create Storefront** on the Profile List page.
This displays the Organization Detail: New Profile page.
3. Enter basic information for the storefront administrator partner as you would do for any other partner. * denotes required fields.

4. In the Detail tab, in the **External Partner ID** field, enter the Organization Code of the catalog organization as defined in the Applications Manager. This is required to ensure that the correct Organization Code is passed while making XAPI calls to the Selling and Fulfillment Foundation.
5. Enter a skin URL for the new storefront. This should be a simple string and must be unique within the Visual Modeler. For example, you can use "andere1" or "storefront". This string will be used in URLs used to access the storefront. For example:

```
http://server:port/Sterling/en/US/enterpriseMgr/andere1
```

6. Click **Save**.

You must create at least one user to act as the first storefront administrator for the new storefront. See "To Create a New Partner User" on page 437 for details. Notify the organization for whom you have created the storefront and provide them with their new storefront URL and their storefront administrator userid.

Managing the Enterprise Profile

In addition to managing partner profiles, some enterprise users are responsible for managing the enterprise profile (that is, the profile of the tenant or storefront partner). Almost all of the profile fields are the same as for partner profiles, and so are covered in their respective sections above. However, some fields are used only by enterprise profiles and child nodes. This section documents these fields.

Info Tab

For enterprise profiles, an additional field is displayed:

- **Max Reps Per Account:** enter the maximum number of users belonging to this profile that may be assigned to any particular partner profile account.

Commerce Tab

You can manage the skins for the tenant and storefront partners.

Current Accounts

Use this tab to assign partners to each node of the enterprise. This allows you to manage which partner accounts are managed by which enterprise nodes.

This chapter covers the tasks involved in creating and modifying models.

<p>Attention: Models are compiled to XML files. Consequently, do not use the following characters when naming models and model entities such as groups, properties, and rules: "&", "/", "@", "!", and the quote characters " and '.</p>

Visual Modeler Interface

The Visual Modeler page consists of three frames:

- **Model Groups:** When you first access the Visual Modeler page, this frame displays the root model group, that is, the highest group in the model group hierarchy.

You can expand a model group to display the model groups within it by clicking the plus (+) sign.

- **Models and Groups:** This displays the models and groups that are children of the model group selected in the Model Groups frame.
- **Content:** This displays information about the model group selected in the Model Groups frame. The information is collected into the following tabs:

- **General Info:** Displays the children of the model group (where you can select, delete, and reorder children). You can also create new model groups, new models, and new groups. You can also upload models or model groups to the currently selected model group.
- **Properties:** You can define new properties in this tab which you can then attach to any model, option class, or option item within the model group for which the property was defined. In the same way, you can also use the property in rules defined for any model or model group in the hierarchy below the model group for which it is defined.
- **Rules:** You can define rules for the model group. These rules can be attached to any models, option classes, or option items in the hierarchy below the model group for which it is defined.
- **Lists:** The list you define here can be used in any properties in the hierarchy below the model group for which it is defined.

In addition, the Visual Modeler page contains a *toolbar* across the top with access to the following tasks:

- **Edit:** This enables you to edit a model, option class group, or option item group highlighted in the Models and Groups frame.
- **Compile:** This enables you to compile a model, option class group, or option item group into an XML file. See "Compiling a Model" on page 486. Only compiled models can be associated with configurable products.
- **Test:** This enables you to test a model that you are creating or modifying. See "Testing a Model" on page 485.
- **Copy:** This enables you to copy a selected entity (model group, model, option class group, and so on).
- **Import:** This enables you to import an entity into your library of entities. See "Importing Model Groups and Models" on page 553.
- **Export:** This enables you to export an entity. See "Exporting Model Groups and Models" on page 554.
- **Report:** This enables you to produce a report on some entity in the model library. See "Reporting" on page 558.
- **Search:** This enables you to search for entities based on selected search parameters. See "Searching" on page 556.

When you build a model, you use the model detail page. The detail page contains the following frames:

- **Toolbar:** as described above.
- **Navigation:** Click the plus (+) sign to expand the model or group and display the elements of the model or group: the sub-models, option classes, option items, or groups.
- **Content:** This displays information about the model selected in the Navigation frame. By navigating to a particular node in the model, you can create and update information on that node. This information is collected into the following tabs:
 - **General Info:** Displays general information about the model or group, as well as the children. You can delete or re-order children in this frame. You can translate the model, assign a product ID to the model, create option classes and (for option item groups) option items, and attach groups. You can also download models from here.

Note: Only the General Info tab appears for option class groups or option item groups.
--

- **Display:** This tab enables you to define display properties at the model level. These properties include things like constant guiding text, as well as pre- and post-pick guiding text. Some display properties have default values which can be overridden by display values set at the option class or option item levels. Note that all the properties displayed on the **Display** tab can also be set by setting UI properties on the **Properties** tab. See "Working with Display Properties" on page 562 for more information about display properties and UI properties.
- **Properties:** If the current node is a model, then this tab consists of two tabs: **Attach** and **Define**; otherwise you can use this tab to attach properties to the node. In the **Attach** tab, you can attach to the model properties to which the model has access. (The model has access either to properties defined specifically for the model itself or to properties defined for any model group above the model in the model group hierarchy.) In the **Define** tab, you can define new properties for use locally, in the model's structure.
- **Rules:** If the current node is a model, then this tab consists of two tabs: **Attach** and **Define**; otherwise you can use this tab to attach rules to the node. In the **Attach** tab, you can attach to the model rules to which the model has access. (The model has access either to rules defined

specifically for the model or rules defined for any model group above the model in the model group hierarchy.) In the **Define** tab, you can define new rules for use locally, in the model's structure.

- **Lists:** If the current node is a model, then the lists you define here can be used locally, in any properties you define for the model.
- **Tables:** If the current node is a model, then you create constraint tables here.
- **Tabs:** If the current node is a model, then you can create a tab-based configuration for your customers here. See "Working with a Tabbed User Interface" on page 488.
- **Worksheets:** If the current node is a model, then this tab enables you to manage properties using worksheets. These provide you with a quick way to view and manage related properties and option items. See "Using Worksheets" on page 502.

To Access the Visual Modeler

1. Click **Configuration Models** in the Product and Catalog Administration panel on the Visual Modeler home page.

This displays the Visual Modeler page.

2. In the Models and Groups frame, click on a model or a group.

This displays the current structure (option classes, option items, and groups) for the selected model or group. Click the plus (+) sign to the left to expand the structure of the model.

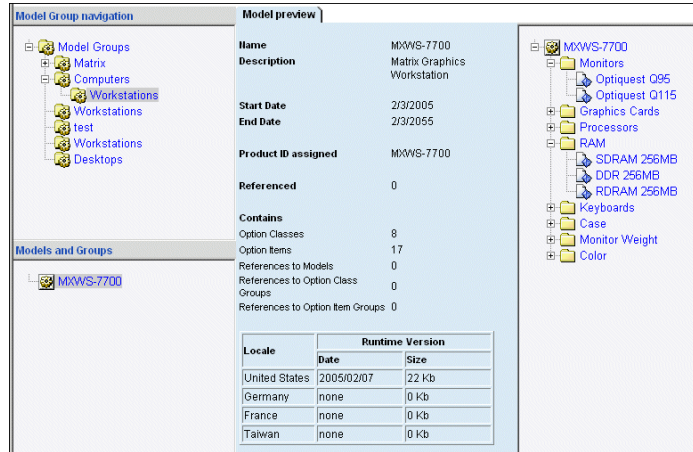


FIGURE 26. Model Structure Panel

3. Click **Edit** in the taskbar.

This displays the detail page for the model, option class group, or option item group.

4. In the Navigation frame, click on the plus (+) sign to expand the model or group (see Figure 27 on page 447).



FIGURE 27. Navigation Frame

5. Click an option class.

This displays the following tabs in the Content frame:

- **General Info:** This tab provides general information about the selected option class. A list box displays the children belonging to this option class. You can also assign a product ID here, define a ratio for the class (the number by which the option item quantity will be multiplied to get the necessary option item quantity). You can create nested option classes and option items as well as attach groups.

- **Display:** This tab enables you to set display property values specific to the selected option class.
 - **Properties:** You can associate with the option class properties to which the option class has access. (The option class has access either to properties defined specifically for the model to which the option class belongs or to properties defined for any model group above the option class in the model group hierarchy.)
 - **Rules:** You can attach rules defined for the model, as well as for the model group to which it belongs (or to any ancestor model group).
6. In the Navigation frame, click the plus (+) to expand the option class.
- This displays the children of the option class: these may be option items or option classes.
7. Click an option item.
- This displays the following tabs in the Content frame:
- **General Info:** This tab provides general information about the selected option item: name and description, effectivity dates, and a field for assigning a product ID.
 - **Display:** You can set display property values specific to the selected option item.
 - **Properties:** You can associate with the option items properties to which the option item has access. (The option item has access either to properties defined specifically for the model to which the option item belongs or to properties defined for any model group above the option item in the model group hierarchy.)
 - **Rules:** You can attach any accessible rules to the option item. (The option item has access to any rules defined at any level above it in the model group hierarchy.)

Working with Model Groups

Model groups provide you with a way of organizing related models into appropriate sections.

To Create a Model Group

1. Navigate to and select the model group under which you wish to create the new model group.

See "To Access the Visual Modeler" on page 446 for information on how to display the model group.

2. Click **New Model Group**.

This displays the **New Model Group** tab.

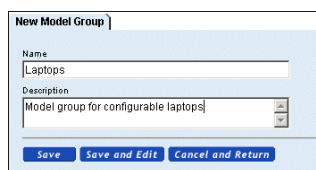


FIGURE 28. New Model Group Tab

3. Enter a name and description for the new model group.

<p>Note: On Windows platforms, there is a 256 character limit for a fully-qualified pathname (this includes the pathname <i>and</i> the filename). Therefore, in Visual Modeler, take care not use long names for either model groups or models, particularly if you are using non-ASCII characters. When you compile a model, Visual Modeler recreates the model group structure as directories in the file system and, in the process, expands any non-ASCII characters.</p>

4. Click **Save** or **Save and Edit** to save the new model group.

The new model group appears in the Model Groups frame. If you clicked **Save and Edit**, then the Visual Modeler page appears, ready for you to edit the new model group. See "To Modify a Model Group" on page 449.

To Modify a Model Group

1. Navigate to and display the model group you want to modify.

See "To Access the Visual Modeler" on page 446 for information on how to display the model group.

This displays the **General Info** tab where you can modify the name and description of the group. You can also do one or more of the following:

- Delete model groups, models, or groups that are children of the selected model group (see "To Delete the Children of a Model Group" on page 450).

Attention: Click **Save All Changes** to save your changes before you leave the **General Info** tab.

- Create a model group as a child of this group. See "To Create a Model Group" on page 449.
 - Create a model as a child of this group. See "To Create a Model" on page 453.
 - Create either an option class group or an option item group. See "Working with Option Class Groups and Option Item Groups" on page 469.
2. Click the **Properties** tab to create or modify properties for this model group. See "Properties" on page 492.

Attention: Click **Save All Changes** to save your changes before you leave the **Properties** tab.

3. Click the **Rules** tab to create or modify rules for this model group. See "Rules" on page 510.
4. Click the **Lists** tab to create or modify lists for this model group. See "Lists" on page 506.

To Delete the Children of a Model Group

To delete one or more children in a group (a model group, a model, an option class group, or an option item group), use the following procedure:

1. Navigate to and select the parent model group that contains the child you want to delete.
See "To Access the Visual Modeler" on page 446 for information on how to display the model group.
2. In the list box, select one or more model groups (MG), models (M), option class groups (OCG) or option item groups (OIG) to be deleted.
 - You cannot delete a model group if the group has children. You must delete the children first.

- You cannot delete a model if it is attached as a sub-model elsewhere in the model group hierarchy.
- You cannot delete an option class group if it is attached to another model or option class group.
- You cannot delete an option item group if it is attached to another model, option class group, or option item group.

3. Click **Delete**.

4. Click **Save All Changes**.

The model group hierarchy will no longer display the deleted items.

To Copy a Model Group

You can copy a model group and its components into another model group.

1. Navigate to and select the model group you wish to copy.

See "To Access the Visual Modeler" on page 446 for information on how to display the model group.

2. In the taskbar, click **Copy**.

This displays the Copy window.

3. Enter the Destination Model Group.

a. Click **Browse...**

This displays a Hierarchy Browser.

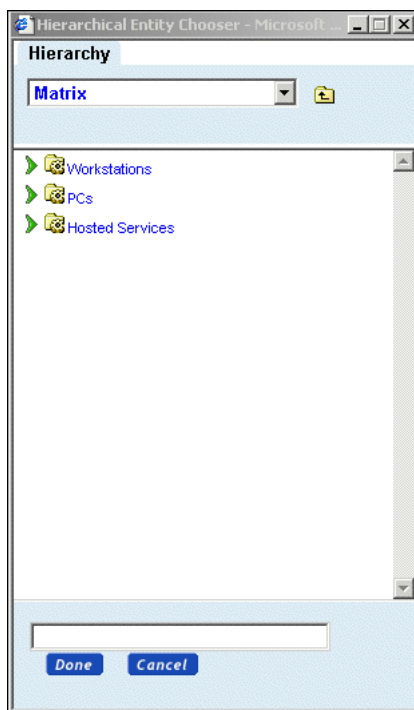


FIGURE 29. Hierarchical Entity Chooser

- b. Browse the model group hierarchy until you find the destination model group.
- c. Select the destination model group.
- d. Click **Done**.

The model group appears in the Destination Model Group field.

4. As desired, modify the Destination Name field.

The name defaults to the name of the model group being copied.

5. Click **Copy** in the Copy window.

The model group is copied to the destination model group.

Working with Models

To Create a Model

1. Navigate to and display the model group under which you wish to create a model.

See "To Access the Visual Modeler" on page 446 for information on how to display the model group.

2. In the **General Info** tab, click **New Model**.

This displays the **New Model** tab.

FIGURE 30. Creating a New Model

3. Enter a name and description for the new model.

If you plan to associate the model with a product ID, consider skipping this step. If the name and description match the name and description of the product ID, you can auto-fill these fields when you assign the product ID in Step 5.

Note: On Windows platforms, there is a 256 character limit for a fully-qualified pathname (this includes the pathname *and* the filename. Therefore, in Visual Modeler, take care not use long names for either model groups or models, particularly if you are using non-ASCII characters. When you translate a model, Visual Modeler recreates the model group structure as directories in a file system and, in the process, expands any non-ASCII characters.

4. Select the Start Date and End Date for the model.

These are the dates within which the model is available for configuration. If the current date is outside these dates, the model is not available for configuration for any product with which it is associated.

5. If applicable, assign a product ID.

See "To Associate a Product with a Model, Option Class, or Option Item" on page 456.

6. Click **Save** or **Save and Edit** to save the new model.

If you click **Save**, the **New Model** tab remains and the new model appears in the Models and Groups frame. You can create another model in this group.

If you click **Save and Edit**, the Model Detail page appears with the new model in the Navigation frame. You can now add properties, rules, lists, and constraint tables for this model. You can also associate the model with a product. See "To Modify an Existing Model" on page 454.

To Modify an Existing Model

1. In the model group hierarchy, navigate and display the Model Detail page for the model you want to modify.

See "To Access the Visual Modeler" on page 446 for information on how to display the Model Detail page.

2. In the **General Info** tab, you can do one or more of the following:
 - Modify the name, description, and/or the start and end dates.
 - Delete one or more of the option classes or groups associated with the model. See "To Delete the Children of a Model" on page 455.
 - Arrange the order of the children in the list.
 - Assign a product to the model, or change the current product assignment.

See "To Associate a Product with a Model, Option Class, or Option Item" on page 456.

Attention: Click Save All Changes to save your changes before you leave the General Info tab.
--

- Create one or more option classes. See "To Create an Option Class" on page 461.
- Attach an option class group. See "Working with Option Class Groups and Option Item Groups" on page 469.

- Modify display properties. See "Working with Display Properties" on page 562.
- 3. Click the **Properties** tab to define properties for or to attach properties to this model.
See "Properties" on page 492.
- 4. Click the **Rules** tab to define rules for or attach rules to this model.
See "Rules" on page 510.
- 5. Click the **Lists** tab to create lists for this model.
See "Lists" on page 506.
- 6. Click the **Tables** tab to create or modify constraint tables.
See "Option Constraints" on page 545.

To Delete a Model

You delete a model by finding the model group that is its parent, then deleting the model from that group. You cannot delete a model if it is attached as a sub-model elsewhere in the model group hierarchy.

See "To Delete the Children of a Model Group" on page 450 for the procedure.

To Delete the Children of a Model

Use this procedure to delete one or more option classes or groups that are children of a model:

1. Navigate to and display the Model Detail page for the model with the elements you want to delete.
See "To Access the Visual Modeler" on page 446 for information on how to display the model.
The **General Info** tab contains a list box showing the option classes (OC), option class groups (OCG) or option item groups (OIG) that are children to the model.
2. In the list box, select one or more objects to be deleted.
3. Click **Delete**.

Note: Attached sub-models and groups are not deleted by this action. Only the attachment to those models and groups is removed. See "To Delete a Group" on page 482.

4. Click **Save All Changes**.

The model hierarchy no longer displays the deleted children.

To Associate a Product with a Model, Option Class, or Option Item

You can reference a model, option class, or option item to a product ID in the product catalog. If the product ID has been assigned to one or more price lists in Sterling Pricing, then this enables you to associate a price with the entity. In addition, if the item associated with a product is selected as part of a configuration, then when the user adds the configured product to their cart, the item is displayed with associated product ID and product information.

1. In the model group hierarchy, find the entity that you want to associate with a product ID.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. In the **General Info** tab for the model, option class, or option item, click **Browse...** to search for the product ID in the product catalog.

- The product ID must exist in the product catalog. You create the product using Sterling Product Manager.
- See "Searching the Product Catalog for a Product ID" on page 488 for help in browsing for a product ID. When you select the product ID, the product ID is displayed in the Assigned Product ID field and its product name and description are auto-filled into those fields.
- You can manually enter the product ID in the Assigned Product ID field, but the Product Name and Product Description fields are not auto-filled until you save the information.
- You can use the product name as the name of the new model. If the Name field is blank, then the field will be auto-filled with the product name. If the field has an entry already, then you will be prompted to use the product name.
- If you are modifying a model, then you can click **Product Detail** to view the details of the assigned product.

3. Click **Save All Changes**.

To Copy a Model

You can copy a model and its components into a model group.

1. In the Model Groups frame, navigate to and select the model group that contains the model you want to copy. The model name is displayed in the Models and Groups frame.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. In the Models and Groups frame, click on the model you want to copy.

This displays the current structure of the model.

3. Click **Copy** in the taskbar.

This displays the Copy window.

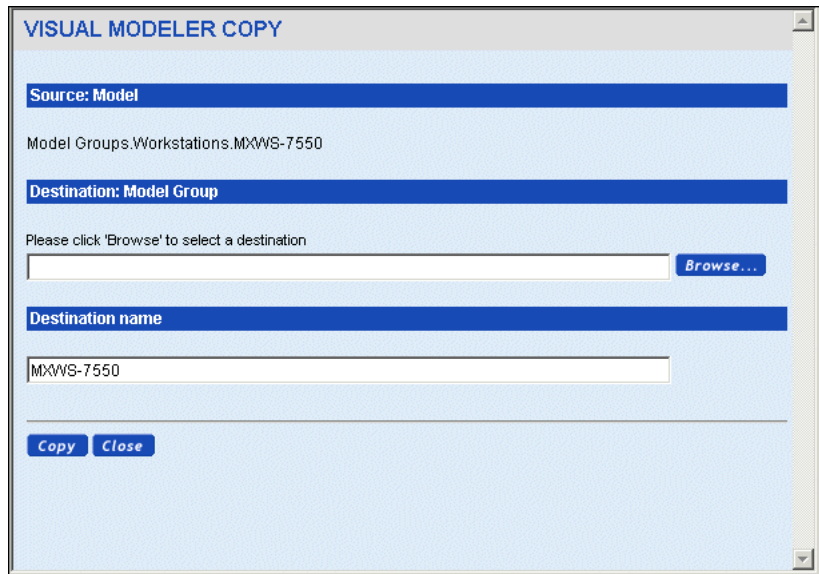


FIGURE 31. Copy Window for Models

4. Enter the Destination Model Group.
 - a. Click **Browse...**

This displays a Hierarchy Browser.
 - b. Browse the hierarchy until you find the destination model group.
 - c. Select the destination model group.

- d. Click **Done**.

The model group appears in the Destination Model Group field.

5. As desired, modify the Destination Name field.

The name defaults to the name of the model being copied.

6. Click **Copy** in the Copy window.

The model is copied to the destination model group.

To Copy a Model Reference

You can re-use a model as part of another entity without having to recreate the model. You do this by attaching the model to the entity. The attachment then becomes a model reference. You can copy this model reference; that is, instead of copying the actual model, you can copy the reference to a model that is attached.

1. In the Model Groups frame, navigate to and select the model group that contains the entity with the model reference you want to copy. The entity name is displayed in the Models and Groups frame.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. In the Models and Groups frame, click on the entity that contains the reference you want to copy.

This displays the current structure of the entity.

3. Click **Edit** in the taskbar.

This displays the **General Info** tab.

4. In the Navigation frame, find and select the model reference you want to copy.

5. Click **Copy** in the taskbar.

This displays the Copy window.

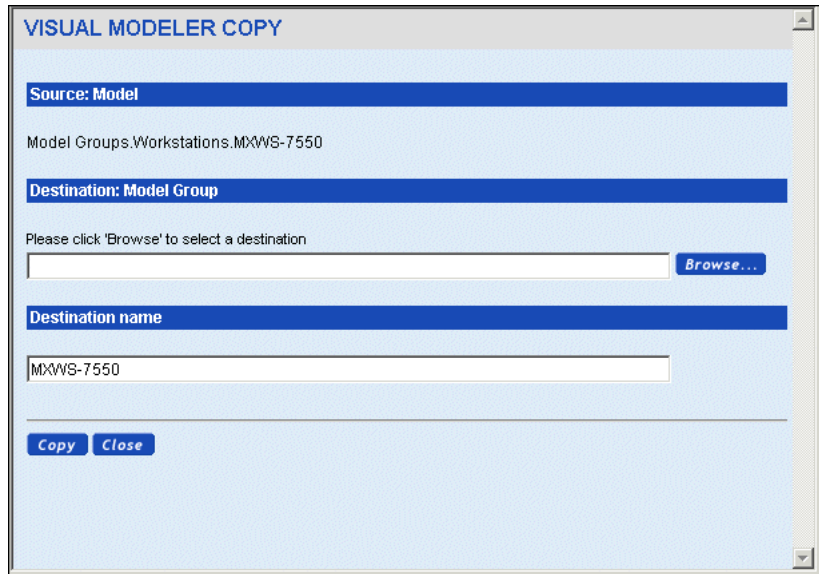


FIGURE 32. Copy Window for Copying Model References

6. Enter the Destination Option Class.
 - a. Click **Browse...**

This displays a Hierarchy Browser.
 - b. Browse the model group hierarchy until you find the destination option class.
 - c. Select the destination option class.
 - d. Click **Done**.

The option class appears in the Destination Option Class field.
7. As desired, modify the Destination Name field.

The name defaults to the name of the model reference being copied.
8. Click **Copy** in the Copy window.

The model reference is copied to the destination option class.

To Embed a Model

You can embed a model within an option class.

1. In the Model Groups frame, navigate to and select the model group that contains the model structure you want to embed. The model name is displayed in the Models and Groups frame.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. In the Models and Groups frame, click on the model whose structure you want to embed.

This displays the current structure of the model.

3. Click **Edit** in the taskbar.

This displays the **General Info** tab.

4. Click **Copy** in the taskbar.

This displays the Copy window.

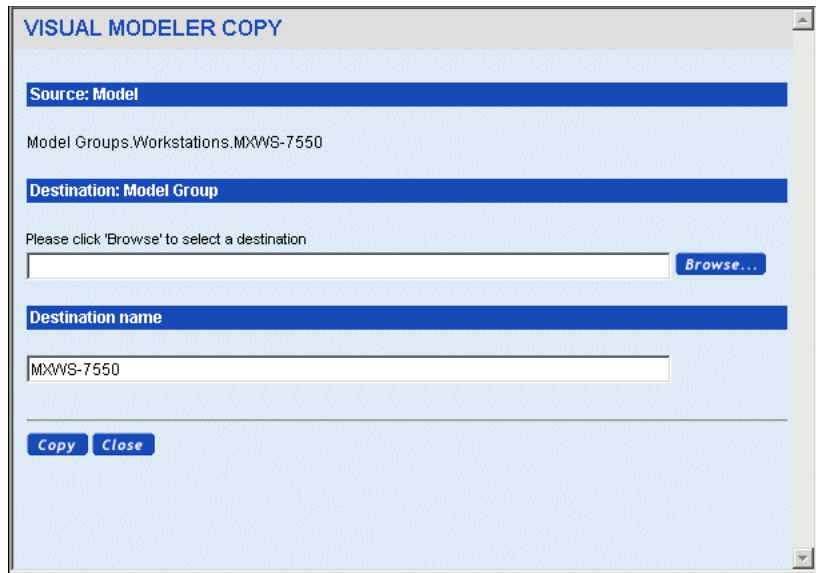


FIGURE 33. Copy Window for Embedding Models

5. Enter the Destination Option Class by typing or by browsing.

To browse for the option class:

- a. Click **Browse...**

This displays a Hierarchy Browser.

- b. Browse the model group hierarchy until you find the destination option class.
- c. Select the destination option class.
- d. Click **Done**.

The option class appears in the Destination Option Class field.

6. As desired, modify the Destination Name field.

The name defaults to the name of the model being embedded.

7. Click **Copy** in the Copy window.

Working with Option Classes and Option Items

Option classes and option items comprise configurable parts or services of a model. You can think of option classes as representing questions or components that need to be configured, while option items represent answers or choices of components. Sometimes the answer to a question can give rise to further questions. In these cases it is useful to nest option classes within other option classes to help guide a user to the configuration that best meets their needs.

To Create an Option Class

1. In the model group hierarchy, navigate to and display the model, option class group, or option class in which you want to create the option class.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

You can create an option class within another option class, within a model, or within an option class group.

2. To create option classes as children of the model or option class group:
 - a. Click **New Option Class**.

This displays the **New Option Class** tab.

- b. Proceed to Step 4.
3. To create nested option classes:
 - a. In the Navigation frame, navigate to and select the option class where you want to nest the new class.
 - b. Click **New Option Class**.
This displays the **New Option Class** tab.
 - c. Proceed to Step 4.

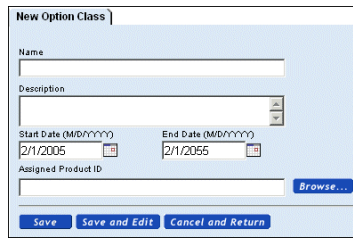


FIGURE 34. New Option Class Tab

4. Enter a name and description for the new option class.
If you plan to associate the option class with a product ID, then you might consider skipping this step. If the name and description match the name and description of the product ID, then you can auto-fill these fields when you assign the product ID in Step 6.
5. Define the effectivity dates by modifying the start and end dates.
You can click the calendar icon to select the dates from a calendar.
6. If applicable, assign a product ID.
See "To Associate a Product with a Model, Option Class, or Option Item" on page 456.
7. Click **Save** to save the new option class and remain at the **New Option Class** tab (to create additional option classes); click **Save and Edit** to save the new option class and display the option class tabs for editing.

The new option class appears in the Navigation frame. The new option class is selected, ready to be modified.

To Modify an Option Class

1. In the model group hierarchy, navigate to and display the model, option class group, or option class that contains the option class.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. In the Navigation frame, find and click on the option class that you want to modify.

This displays the **General Info** tab for the option class.

3. Modify Name, Description, and Start and End Dates as applicable.

4. Enter a ratio in the **Ratio** field, if applicable.

The ratio field determines the quantity of option items that are added to a customer's order. The quantity of any child item selected is multiplied by this ratio to compute the "extended" quantity of the child item. For example, a bicycle model may have a wheel option class defined with a ration of "2". When a user selects a particular wheel item from this option class, then two wheels will be added to the configured product.

You can enter the **Ratio** as either a whole number or a decimal.

5. As applicable, modify the order of the children or delete the children.

See "To Delete the Children of an Option Class" on page 468.

6. If applicable, assign a product ID or modify the current assignment.

See "To Associate a Product with a Model, Option Class, or Option Item" on page 456.

7. Before you click the other tabs, click **Save All Changes**.

8. Click the **Display** tab to modify the display properties for this option class.

See "Working with Display Properties" on page 562.

9. Click the **Properties** tab to attach properties to this option class.

See "To Attach a Property" on page 494.

10. Click the **Rules** tab to attach rules to this option class.

See "To Attach a Rule" on page 514.

When you have completed modifying the option class, click **Save All Changes**.

You can also create option items for this option class. See "To Add Option Items to an Option Class" on page 464.

To Add Option Items to an Option Class

1. In the model group hierarchy, navigate to the option class to which you want to add the option items.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. In the **General Info** tab, click **New Option Item** to display the **New Option Item** tab.

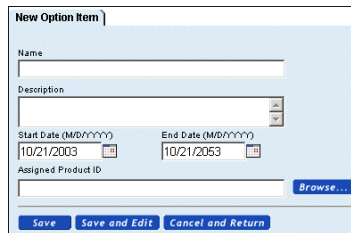


FIGURE 35. New Option Item Tab

3. Enter a name and description for the new option item.

If you plan to associate the option item with a product ID, then you might consider skipping this step. If the name and description match the name and description of the product ID, then you can auto-fill these fields when you assign the product ID in Step 5.

4. Define the effectivity dates by modifying the start and end dates.
5. If applicable, assign a product Id.

See "To Associate a Product with a Model, Option Class, or Option Item" on page 456.

6. Click **Save** or **Save and Edit**.

The new option item appears in the model hierarchy in the Navigation frame.

To Copy an Option Class

You can copy an option class and its components into a model, an option class group, or another option class.

1. Navigate to and select the parent model group for the model or option class group that contains the option class.
See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.
2. In the Models and Groups frame, click on the model or option class group that contains the option class.
This displays the current structure of the model or option class group.
3. Click **Edit** in the taskbar.
This displays the **General Info** tab for the model or option class group.
4. In the Navigation frame, find and click on the option class that you want to copy.
This displays the **General Info** tab for the option class.
5. Click **Copy** in the taskbar.
This displays the Copy window.

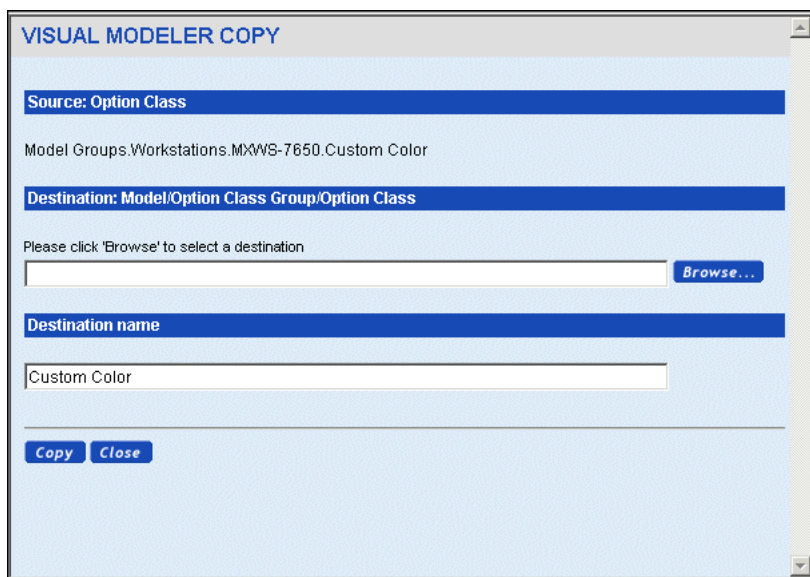


FIGURE 36. Copy Window for Option Classes

6. Enter the destination model, option class group, or option class as follows:

a. Click **Browse...**

This displays a Hierarchy Browser.

b. Browse the model group hierarchy until you find the destination model, option class group, or option class and select it.

c. Click **Done**.

The model, option class group, or option class appears in the Destination Model/OCG/Option Class field.

7. Enter the Destination name.

The name defaults to the name of the option class being copied.

8. Click **Copy** in the Copy window.

The option class is copied to the destination model, option class group, or option class.

To Modify an Option Item

1. Find the option item that you want to modify.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

When you click the option item, the **General Info** tab is displayed.

2. If applicable, modify the name, description, start/end dates.

3. If applicable, assign a product Id.

See "To Associate a Product with a Model, Option Class, or Option Item" on page 456.

Attention: Before you click the other tabs, click Save All Changes .
--

4. Click the **Display** tab to modify the display properties for this option item.

See "Working with Display Properties" on page 562.

5. Click the **Properties** tab to attach properties to this option item.

See "To Attach a Property" on page 494.

6. Click the **Rules** tab to attach rules to this option item.

See "To Attach a Rule" on page 514.

To Copy an Option Item

You can copy an option item into an option item group or an option class.

1. Find the option item that you want to copy.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

When you click the option item in the Navigation frame, the **General Info** tab is displayed.

2. Click **Copy** in the taskbar.

This displays the Copy window.

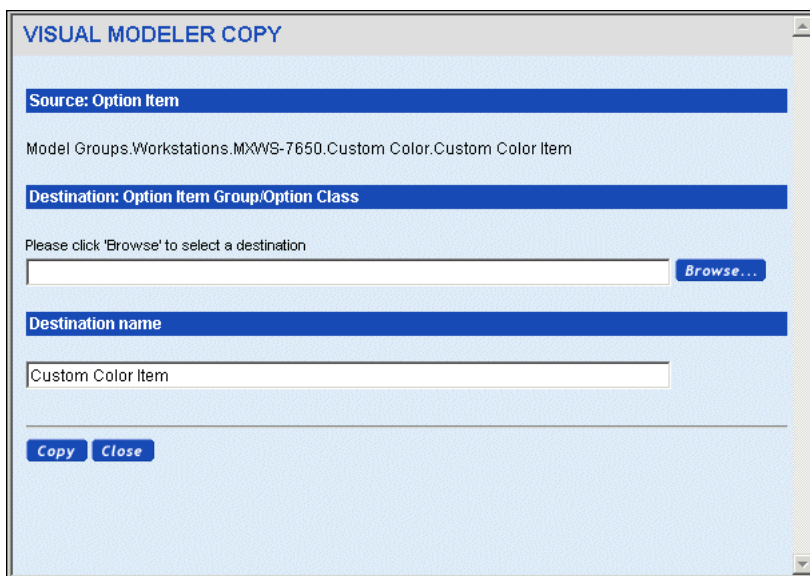


FIGURE 37. Copy Window for Option Items

3. Enter the destination option item group or option class.

- a. Click **Browse....**

This displays a Hierarchy Browser.

b. Browse the model group hierarchy until you find the option item group or option class and select it.

c. Click **Done**.

The option item group or option class appears in the Destination: Option Item Group/Option Class field.

4. Enter the Destination name.

The name defaults to the name of the option item being copied.

5. Click **Copy** in the Copy window.

The option item is copied to the destination option item group or option class.

To Delete an Option Class

You delete an option class by deleting the option class as a child of the parent to which it belongs. This can be one of the following:

- A model. See "To Delete the Children of a Model" on page 455.
- An option class. See "To Delete the Children of an Option Class" on page 468.
- An option class group. See "To Delete the Children of a Group" on page 483.

Deleting the option class automatically deletes any option items, nested option classes, or attachments to groups.

Note: Nested groups are not deleted when you delete an option class, only the attachment to those groups.
--

To Delete the Children of an Option Class

You can delete option items and nested option classes, as well as any attachments to groups.

1. Navigate to and display the detail page for the model or option class group that contains the option class.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. In the Navigation frame, navigate to and select the option class.

This displays the **General Info** tab which contains a list box showing the children of the option class.

3. Click the item to be deleted: option item (OI), option class (OC), model, option class group (OCG), or option item group (OIG).

Note: Nested groups are not deleted. However, the attachment to those groups is removed.

4. Click the **Delete** button.
5. Click **Save All Changes**.

The items are no longer displayed in the Navigation frame.

Working with Option Class Groups and Option Item Groups

To Create a Group

1. In the Model Groups frame, navigate to and select the model group for which you are creating the option class group or option item group.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

This displays the **General Info** tab for the group. Make sure you are creating the group within the appropriate model group. The group will be available for attachment to any items below this model group in the model group hierarchy.

2. Click **New Option Group**.

This displays the **New Option Class/Item Group** tab (see Figure 38 on page 469).

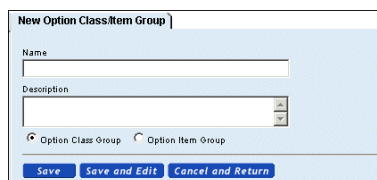


FIGURE 38. New Option Class/Item Group tab

3. Enter a name and description for the group.
4. Select the type of group (Option Class Group or Option Item Group).

5. Click **Save** or **Save and Edit**.

The group appears in the hierarchy. You can now begin to build the group. The first step is to create one or more option classes. See "To Create an Option Class" on page 461.

To Modify a Group

When you modify a group and then compile it, the modifications are reflected in any model to which the group is attached, once the model is recompiled.

1. In the model group hierarchy, navigate to and select the option class group or option item group that you want to modify.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

This displays the **General Info** tab for the group.

2. Modify the name and description, reorder or delete the children.

See "To Delete the Children of a Group" on page 483 for information about deleting the children of a group.

3. (Option item groups only) If applicable, define start/end dates.

4. Click **Save All Changes**.

You can also do the following:

- Add option classes to an option class group. See "To Create an Option Class" on page 461.
- Attach groups to the group. See "To Attach a Group to a Model or Another Group" on page 476.

To Copy an Option Class Group

You can copy an option class group to a model group.

1. In the Model Groups frame, navigate to and select the model group that contains the group you want to copy. (See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.)

2. In the Models and Groups frame, click on the group you want to copy.

The current structure of the group, if any, appears in the content frame.

3. Click **Copy** in the taskbar.

This displays the Copy window.

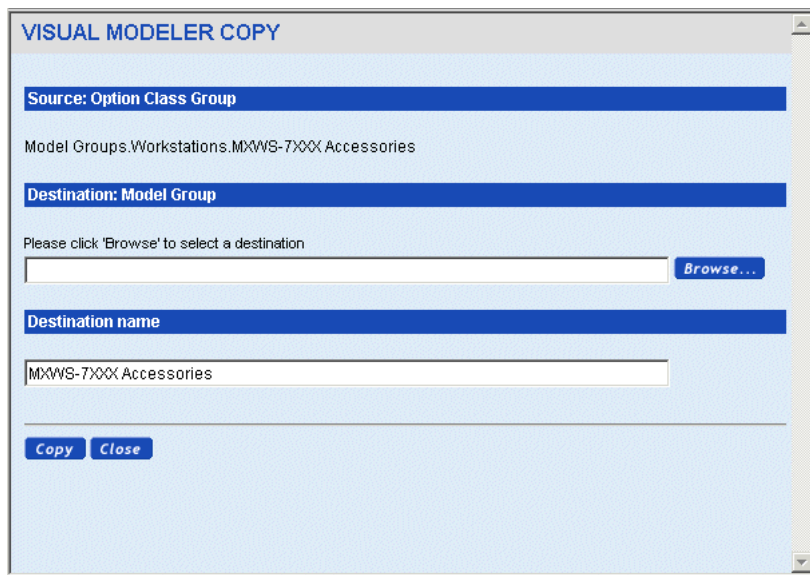


FIGURE 39. Copy Window for Option Class Groups

4. Enter the Destination Model Group.
 - a. Click **Browse...**
- b. Browse the model group hierarchy until you find the destination model group and select it.
- c. Click **Done**.

The model group appears in the Destination Model Group field.

5. Enter the Destination name.

The name defaults to the name of the option class group being copied.

6. Click **Copy** in the Copy window.

The option class group is copied to the destination model group.

To Embed an Option Class Group

You can embed an option class group within a model, another option class group, or an option class.

1. In the Model Groups frame, navigate to and select the model group that contains the option class group you want to embed. The group name is displayed in the Models and Groups frame.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. In the Models and Groups frame, click on the group you want to embed.
This displays the current structure of the group.

3. Click **Edit** in the taskbar.
This displays the **General Info** tab.

4. Click **Copy** in the taskbar.
This displays the Copy window.

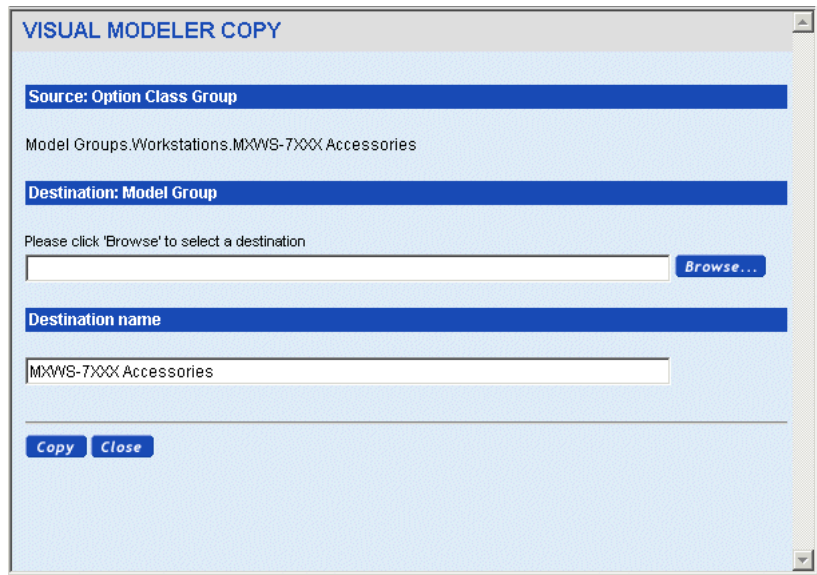


FIGURE 40. Copy Window for Embedding Option Class Groups

5. Enter the destination model, option class group, or option class as follows:
 - a. Click **Browse...**
This displays a Hierarchy Browser.

- b. Browse the model group hierarchy until you find the destination model, option class group, or option class and select it.
- c. Click **Done**.

The model, option class group, or option class appears in the Destination Model/Option Class Group/Option Class field.

6. Click **Copy** in the Copy window.

The option class group is embedded in the destination model, option class group, or option class.

To Copy an Option Item Group

You can copy an option item group to a model group.

1. In the Model Groups frame, navigate to and select the model group that contains the option item group you want to copy. (See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.)
2. In the Models and Groups frame, click on the option item group you want to copy.

The current structure of the group, if any, appears in the content frame.

3. Click **Copy** in the taskbar.

This displays the Copy window.

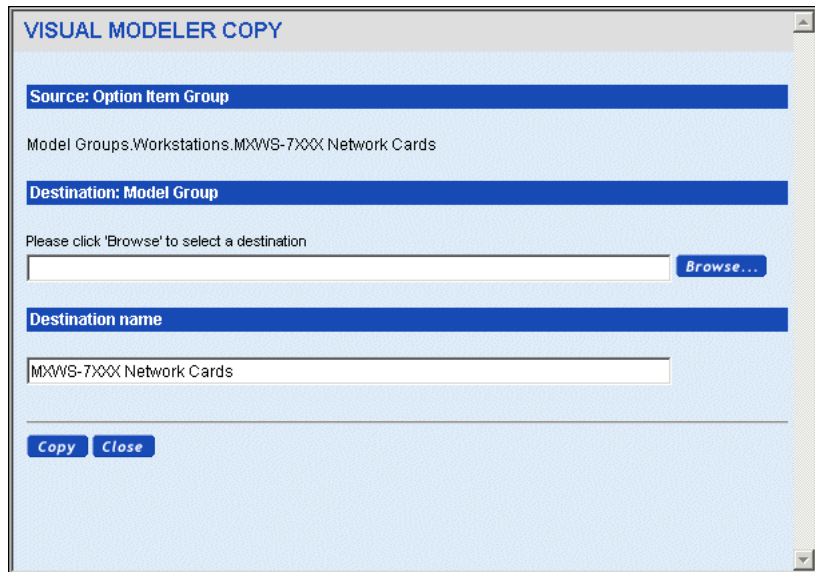


FIGURE 41. Copy Window for Option Item Groups

4. Enter the Destination Model Group.
 - a. Click **Browse...**

This displays a Hierarchy Browser.
 - b. Browse the model group hierarchy until you find the destination model group.
 - c. Select the destination model group.
 - d. Click **Done**.

The model group appears in the Destination Model Group field.
5. Enter the Destination name.

The name defaults to the name of the option item group being copied.
6. Click **Copy** in the Copy window.

The option item group is copied to the destination model group.

To Embed an Option Item Group

You can embed an option item group within another option item group or option class.

1. In the Model Groups frame, navigate to and select the model group that contains the option item group you want to embed. The group name is displayed in the Models and Groups frame.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. In the Models and Groups frame, click on the group you want to copy.

This displays the current structure of the group.

3. Click **Edit** in the taskbar.

This displays the **General Info** tab.

4. Click **Copy** in the taskbar.

This displays the Copy window.

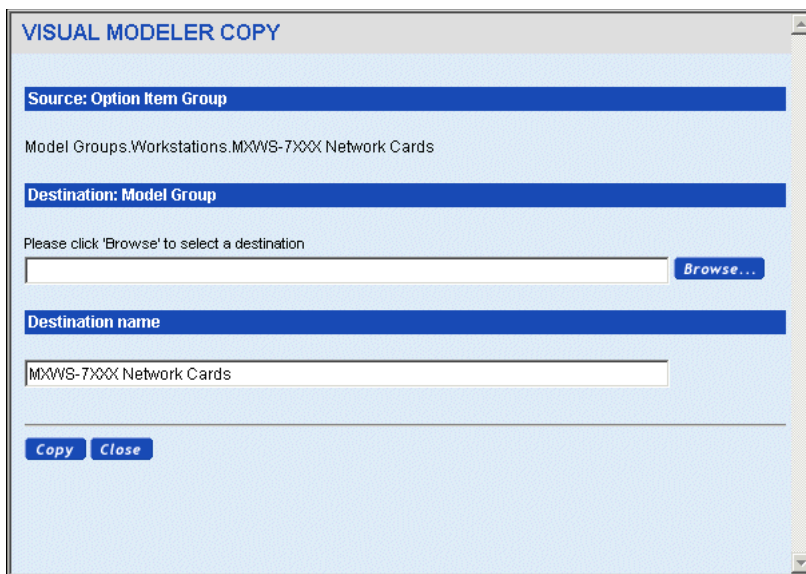


FIGURE 42. Copy Window for Embedding Option Item Groups

5. Enter the destination option item group or option class.

a. Click **Browse...**

This displays a Hierarchy Browser.

b. Browse the model group hierarchy until you find the destination option item group or option class and select it.

c. Click **Done**.

The option item group or option class appears in the Destination OIG/Option Class field.

6. Click **Copy** in the Copy window.

The option item group is embedded in the destination option item group or option class.

To Attach a Group to a Model or Another Group

You can attach a model only to an option class (see "To Attach a Model, Option Class Group, or Option Item Group to an Option Class" on page 477). You can attach an option class group to a model, an option class, or another option class group. You can attach an option item group to an option class or to another option item group.

1. In the Model Groups frame, navigate to and select the model group that contains the model or group to which you want to attach the group.

See "To Access the Visual Modeler" on page 446 for information about how to navigate the model group hierarchy.

2. In the Models and Groups frame, click on the model or group to which you want to attach the option class group or option item group.

3. Click **Edit**.

This displays the **General Info** tab for the model or group.

4. In the **General Info** tab, click **Attach**.

This displays the **Attach** tab.



FIGURE 43. Attach Tab

5. Enter a name and description for the attachment to the group or model.
6. Select the option class group or option item group to be attached.

- a. Click **Browse...**

This displays a Hierarchy Browser.

- b. Browse the model group hierarchy until you find the option class group or option item group.
- c. Select the group.
- d. Click **Done**.

The group appears in the selection field.

7. Click **Assign**.

You can click **Return to General** to return to the **General Info** tab.

The name you entered for the attached group or model appears in the model hierarchy in the Navigation frame.

To Attach a Model, Option Class Group, or Option Item Group to an Option Class

You can attach a model, an option class group, or an option item group to an option class.

1. In the Model Groups frame, navigate to and select the model group that contains the model with the option class.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. In the Models and Groups frame, click on the model or option class group that contains the option class.

The current structure of the model or group, if any, appears in the content frame.

3. Click **Edit**.

This displays the **General Info** tab for the model or group.

4. In the Navigation frame, navigate to and select the option class to which you want to attach the group.

This displays the **General Info** tab for the option class.

5. In the **General Info** tab, click **Attach**.

This displays the **Attach** tab.

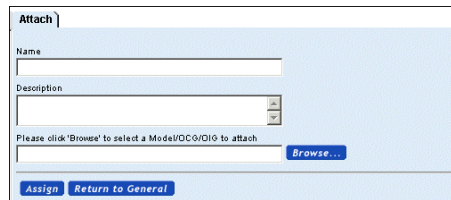


FIGURE 44. Attach Tab

6. Enter a name and description for the attached group or model.
7. Select the model, option class group, or option item group to be attached.

- a. Click **Browse...**

This displays a Hierarchy Browser.

- b. Browse the model group hierarchy until you find the model, option class group, or option item group.
- c. Select the model or group.
- d. Click **Done**.

The model or group appears in the selection field.

8. Click **Assign**.

You can click **Return to General** to return to the **General Info** tab.

The name you entered for the attached model or group appears in the model hierarchy in the Navigation frame.

To View the Structure of an Attached Group

Once a group is attached, you can view the group's structure by clicking **Show Detail**.

1. Navigate to the level in the hierarchy (model, option class or option item) where the group is attached.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. Click **Show Detail**.

This displays a read-only view of the group's structure.

To Copy an Option Class Group Attachment

You can copy a reference to an option class group; that is, rather than copy the group itself, you copy the reference to the group. You can copy the reference into either a model, an option class group or into an option class.

1. In the Model Groups frame, navigate to and select the model group that contains the entity with the option class group attachment you want to copy. (See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.)

2. In the Models and Groups frame, click on the entity that contains the attachment you want to copy.

The current structure of the model appears in the content frame.

3. Click **Edit** in the taskbar.

This displays the model in the Navigation frame and the **General Info** tab for the group.

4. In the Navigation frame, navigate the model until you find the attached group you want to copy.

5. Click the attached group.

6. Click **Copy** in the taskbar.

This displays the Copy window.

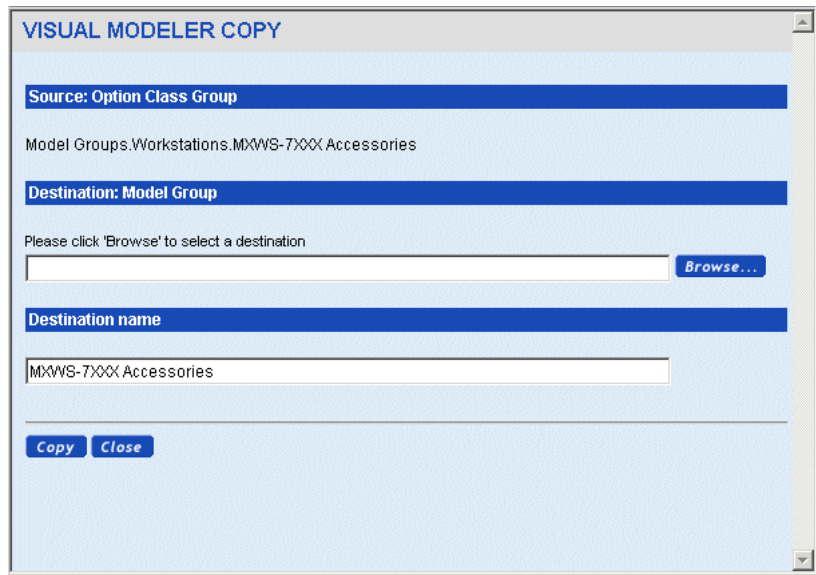


FIGURE 45. Copy Window for Option Class Group Attachments

7. Enter the destination model, option class group, or option class as follows:
 - a. Click **Browse...**
This displays a Hierarchy Browser.
 - b. Browse the model group hierarchy until you find the destination model, option class group, or option class and select it.
 - c. Click **Done**.
The model, option class group, or option class appears in the Destination Model/Option Class Group/Option Class field.
8. Enter the Destination name.
The name defaults to the name of the option class group being copied.
9. Click **Copy** in the Copy window.
The attachment is copied to the destination model, option class group, or option class.

To Copy an Option Item Group Attachment

You can copy a reference to an option item group; that is, rather than copy the group itself, you copy the reference to the group. You can copy the reference into either an option item group or into an option class.

1. In the Model Groups frame, navigate to and select the model group that contains the entity with the option item group attachment you want to copy. (See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.)

2. In the Models and Groups frame, click on the entity that contains the attachment you want to copy.

The current structure of the entity appears in the content frame.

3. Click **Edit** in the taskbar.

This displays the entity in the Navigation frame and the **General Info** tab for the group.

4. In the Navigation frame, navigate the entity until you find the attached group you want to copy.

5. Click the attached group.

6. Click **Copy** in the taskbar.

This displays the Copy window.

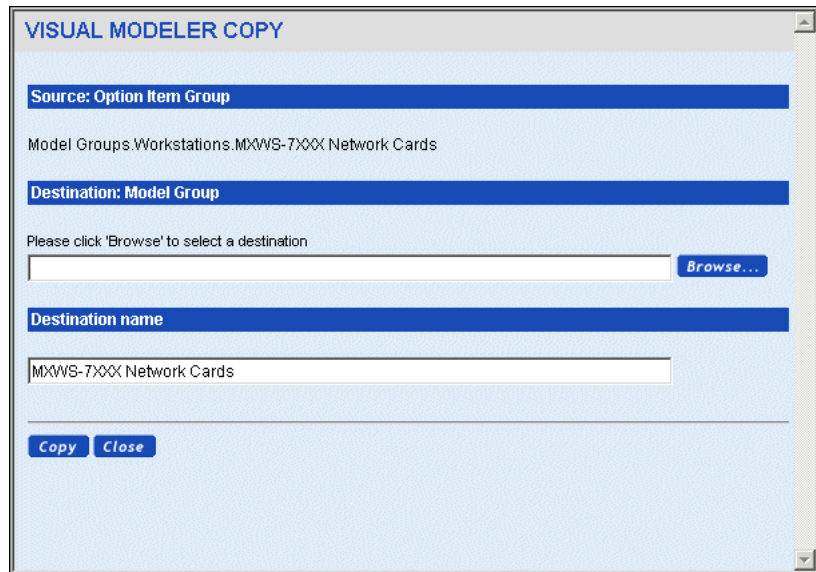


FIGURE 46. Copy Window for Option Item Group Attachments

7. Enter the destination option item group or option class.
 - a. Click **Browse...**
This displays a Hierarchy Browser.
 - b. Browse the model group hierarchy until you find the option item group or option class and select it.
 - c. Click **Done**.
The option item group or option class appears in the Destination Option Item Group/Option Class field.
8. Click **Copy** in the Copy window.

The attachment is copied to the destination option item group or option class.

To Delete a Group

You delete a group by finding the model group that is the parent of the group you want to delete, then deleting the group from that model group. See "To Delete the Children of a Model Group" on page 450 for the procedure.

- You cannot delete an option class group if it is being referenced from another model or option class group.
- You cannot delete an option item group if it is referenced from another model, option class group, or option item group.

To Delete the Children of a Group

Use this procedure to delete one or more option classes or groups that are children of a group:

1. Navigate to and select the parent model group that contains the group with the children you want to delete.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. In the Models and Groups frame, click on the group.

This displays the current structure of the group.

3. Click **Edit**.

This displays the **General Info** tab that contains a list box showing the children belonging to the group. This can include option classes (OC) and option class groups (OCG).

4. In the list box, select one or more elements to be deleted.

5. Click **Delete**.

Note: Groups are not deleted by this action. Only the attachment to those groups is removed. See "To Delete a Group" on page 482.
--

6. Click **Save All Changes**.

The model hierarchy no longer displays the deleted elements.

Including Sub-Models in Models

You can include one model in another so that a sub-component of the parent model can be modeled and configured separately.

To Include a Sub-Model in a Model

Suppose that you have a model A, and you want to use Model B as an option item in Model A, so that end-users can configure the Model B component as part of a session to configure model A.

1. Create Model B as a model in its own right, and compile it. Make a note of the location of this model in the model group and model hierarchy. For example: Matrix/Computers/Workstations/Configurable Monitors/Matrix Monitor.
2. Navigate to Model A and to the location in the Model hierarchy at which you want to include Model B as an option item.
3. Create the option item and enter a name, description, and effectivity dates for it. Click **Save**.
4. Click the **Properties** tab.
5. Select CONFIG: SUBMODEL NAME in the Unattached Properties drop-down list.
6. In the Value field, enter the fully qualified name to Model B. For example, Matrix/Computers/Workstations/Configurable_0020Monitors/Matrix_0020Monitor. Note the use of escape characters to encode special characters such as spaces. See "Special Characters Encoding" on page 484 for more information.
7. Click **Attach**.
8. Click **Save All Changes**.
9. A separate property called CONFIG: SUBMODEL RETURN controls whether end-users return to the main model after configuring the child model.
 - a. If you want to have end-users return to the main model when they have finished configuring Model B, then set the value of CONFIG: SUBMODEL RETURN to "true".
 - b. If you want to have end-users return to directly to the calling application when they have finished configuring Model B, then set the value of CONFIG: SUBMODEL RETURN to "false".
10. Click **Attach**.
11. Click **Save All Changes**.
12. Click **Compile** to re-compile Model A.
13. To test the model, click **Test**.

Special Characters Encoding

You must encode any special characters in model group and model names when you provide model group path names and model names.

The following table lists some common special character encodings:

TABLE 24. Character Encodings

Character	Encoding
" " (blank)	_0020
"_"	_002D
"/"	_002F
"!"	_0021
"@"	_0040
"#"	_0023
"\$"	_0024

Testing a Model

You can test the model at any point while you are creating it. The test model feature performs the following steps:

1. Compiles the model into an XML file.
2. Launches the browser.
3. Displays the model as a HTML page.

To Test a Model

1. Navigate to the model that you want to test.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. Click **Compile**.

A dialog box reports that compilation is successful.

3. Click **Test Model**.

This displays a configuration window as the end-user will see it, based on the current model.

Note that if you click **Compile and Test**, then both actions are taken.

4. To change some of the environmental variables that affect how a model displays, click **Set Defaults**.

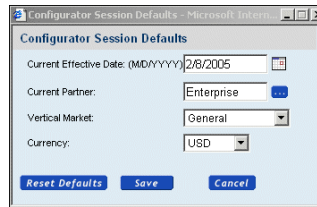


FIGURE 47. Set Defaults Window

Model display environmental variables include:

- **Current Effective Date:** by changing the date in this field, you can view the model as it would be viewed by a customer on the specified date. This means that you would see only option classes and option items that are effective on that date, and the prices that you see are based on price lists effective on that date.
- **Current Partner:** By selecting a specified a partner, you view the model as it would be seen by a user of that particular partner. Depending on the assignment of price lists to the partner, this may affect which option classes and option items are displayed.
- **Vertical Market:** When customers create carts and orders, they can specify a customer type: this is used to filter which price lists are to be used in calculating prices. By selecting a customer type, you can check how the model will be seen by customers who select the same customer type.
- **Currency:** When customers create carts and orders, they can specify a currency: this is used to filter which price lists are to be used in calculating prices. By selecting a currency, you can check how the model will be seen by customers who select the same currency.

Compiling a Model

Before a model can be associated with a configurable product and a customer can use the model you have created to configure a product, you must compile the model into XML format and store the model in a location accessible by Sterling Configurator. Only compiled models can be associated with configurable products. After you create the model, you click a button to compile the model into an XML file. USD is not considered the default currency when you test a model. Currency and Organization Code are the mandatory parameters required for pricing items.

The value of the Currency is fetched based on the preferences set by the user in the Applications manager. For more information about Organization Code and defining currency definitions, refer to the Selling and Fulfillment Foundation: Application Platform Configuration Guide. The Organization Code set in the Visual Modeler for the current storefront is used.

Note: To compile all the models for the locales configured in Visual Modeler, select the “Compile All Locales” check box.

To Compile a Model

1. Navigate to the model that you want to compile.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. Click **Compile**.

The model is compiled into an XML file. This XML file is placed in the following location in *debs_home/Sterling/WEB-INF/data/config/*. This directory contains several directories, one for each locale. The model resides within the directory representing your preferred locale in either the folder representing the root model group folder or in one of the folders representing model groups within the root model group. They are stored in the shared location of a clustered deployment of the Visual Modeler.

<p>Attention: If your implementation of the Visual Modeler makes use of a staging and a production system, then bear in mind that the XML files may have to be moved over to the production environment or the model directories must be shared between the systems.</p> <p>In addition, the product records in the Knowledgebase for configurable products may have to be updated to point to the location of the XML files.</p>
--

If your model group and model hierarchy include special characters (that is, non-alphanumeric characters), then these are encoded in the directory and files names that correspond to them. See "Special Characters Encoding" on page 484 for more information.

To Compile All Models

Rather than compile models one by one, you can also compile all the models in a model group at once.

1. Navigate to the model group whose models you want to compile. This can be the top-level model group.

2. Click **Compile All**.
3. In the Compile All Models window, click **Compile All Models**.
4. The Compile All Models Status window is displayed.
5. When it reports that all the models have been compiled, then click **Close**.

Searching the Product Catalog for a Product ID

When you assign a product ID, you can click **Browse...** to display the Hierarchical Entity Chooser.

You can use this window to navigate through the hierarchy until you find the product ID that you want to assign to the model object. You can click the **Search** tab to search through products unassigned to any product category.

Click **Done** when you find the product ID that you want to assign. The product ID appears in the Assigned Product ID field.

Working with a Tabbed User Interface

You can design your end-user interface so that, rather than being displayed in a single frame, the option classes appear within a series of tabs. You do this by first selecting the Tabbed Configurator JSP template at the model level, which sets the display property UI:JSP Filename (see "Working with Display Properties" on page 562). You then design the end-user interface using **Tabs** tab.

To Create a Tabbed User Interface

1. Navigate to the model for which you want to create the tabbed interface.
See "To Access the Visual Modeler" on page 446 for information about how to navigate the model group hierarchy.
2. Click the **Display** tab.
The **Display** tab displays.
3. Select Tabbed Configurator from the JSP Template drop-down list.
This automatically sets the UI: JSP FILENAME property to Configurator_Tabbed.jsp.
4. Click the **Tabs** tab.
This displays the **Tabs** tab.

5. Enter a name for the tab in the Tab Name field.
6. Click **Add**.
The content frame displays an area for editing the new tab.
7. Select the option classes or option class groups for the tab.
 - a. Select an option class or option class group from the drop-down list.
 - b. Click **Add**.
8. Repeat the last step for each option class or option class group you want in the tab.

<p>Note: If you are creating a tabbed UI, then not all option classes must be accounted for in the tabs. Any option classes not included in a tab will not be displayed to the end-user.</p>

9. Click **Move Up** or **Move Down** to arrange the order of the entities. To remove an entity, click the entity, then click **Remove**.
10. Click **Save All Changes**.

To Modify a Tab

1. Navigate to the model with the tabbed interface.
See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.
2. Click the **Tabs** tab.
This displays the **Tabs** tab.
3. Find the tab element you want to modify.
4. To rearrange the order of the entities within the tab:
 - a. Find and select the entity you want to move.
 - b. Click **Move Up** or **Move Down**.
5. To remove an entity:
 - a. Find and select the entity you want to remove.
 - b. Click **Remove**.
6. To rearrange the location of the tab within the list of tabs, click the up or down arrows in the far right.

7. Click **Save All Changes**.

To Delete a Tab

1. Navigate to the model with the tabbed interface.
See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.
2. Click the **Tabs** tab.
This displays the **Tabs** tab.
3. Find the tab element you want to delete.
4. On the far right, click the **Delete** icon (X) for that tab.
5. Click **Save All Changes**.

The basic concepts and tasks of modeling are covered in CHAPTER 36, "Using the Visual Modeler". This chapter and the next, CHAPTER 38, "Visual Modeler UI Concepts", describes the more advanced concepts associated with building complex models. This chapter covers:

- "Properties" on page 492
 - "Working With Properties" on page 493
 - "Using Worksheets" on page 502
 - "Properties as Variables" on page 504
 - "Visual Modeler Properties" on page 505
- "Lists" on page 506
 - "Working With Lists" on page 507
- "Rules" on page 510
 - "Working With Rules" on page 510
 - "Working With Rule Fragments" on page 523
 - "Working with Rule Actions" on page 537
- "Fragments" on page 523

- "Working with Rule Actions" on page 537
- "Option Constraints" on page 545
 - "Working With Constraints" on page 545
- "Importing and Exporting Models" on page 553
 - "Importing Model Groups and Models" on page 553
 - "Exporting Model Groups and Models" on page 554
- "Using Dynamic Instantiation" on page 555
- "Searching" on page 556
- "Reporting" on page 558

Properties

A property is an attribute of a model, option class, or option item. It is used as a basic building block for rule creation.

The Visual Modeler provides a set of built-in properties which are understood by the Sterling Configurator engine. These control the behavior of the engine and the presentation of the model to the end-user. These properties are summarized in "Visual Modeler Properties" on page 505.

You can also define properties and they are available for use in any part of the model group and model hierarchy beneath the point at which they are defined. These defined properties are used to describe the product so that the Sterling Configurator engine can ensure that the user-configured model is valid.

You can also use properties as variables and write rules that reason on the properties' values using functions such as value and expand.

Working With Properties

To Define a Property

1. Navigate to and select the location in the model group hierarchy where you want to create the property.

See "To Access the Visual Modeler" on page 446 for information about how to navigate the model group hierarchy.

This is important since where you create the property determines what objects in the hierarchy can use the property.

2. When you reach the appropriate level, click the **Properties** tab.
3. If you are working in a model, then within the **Properties** tab, click the **Define** tab.
4. Enter a name for the property.

Note: Do not begin a property name with "UI:" or "CONFIG". Do not include a period (.) in a property name.

5. Select a property type from the drop-down list.
 - **Number:** use this for any property whose value is determined by a number. For example, the weight of an item could be expressed as a real number of grams (including decimals).
 - **String:** use this for any property that is expressed as a word or phrase. For example, you can use a string-valued property to indicate the color of an option item.

If you select this type, then the Localize field is enabled. If you check this box, then you can enter values for this property in any of the supported locales. In other words, if you enter the original value in English, then you can change the system locale to German and then modify the property's value in German. The German value will appear for those users whose locale is German; the English value will appear for those users whose locale is English.

- **List:** use this for any property where the value of the property must be selected from a list. For example, the availability of an item might be limited to specifying one or more days of the week. You can capture this in the form of a property by defining a list called "Weekdays" whose values are Sunday, Monday, and so on, concluding with Saturday.

6. If applicable, define a default value that this property takes. You can override this value when you apply the property to an item or class.

If you selected "List" as the property type, then the Value field displays a drop-down selection of the current lists available. Select a list. See "Lists" on page 506 for information about creating lists.

7. Click **Add**.

The new property appears in the boxes below the fields.

8. Click **Save All Changes** to save the new property.

To Attach a Property

You define a property at the model group or model level (see "To Define a Property" on page 493). You attach a property to a model, an option class, or an option item.

1. Click **Configuration Models** in the Product and Catalog Administration panel on the Visual Modeler home page.
2. In the Navigation frame, navigate to the object to which you want to attach the property.
3. Click the **Properties** tab.

This displays two sets of fields: one called Unattached Properties for selecting properties and defining values for them, and one called Attached Properties that shows the properties that are currently attached.

Note: The Properties tab for a model contains two tabs: Attach and Define . You use the Define tab to define properties. See "To Define a Property" on page 493.

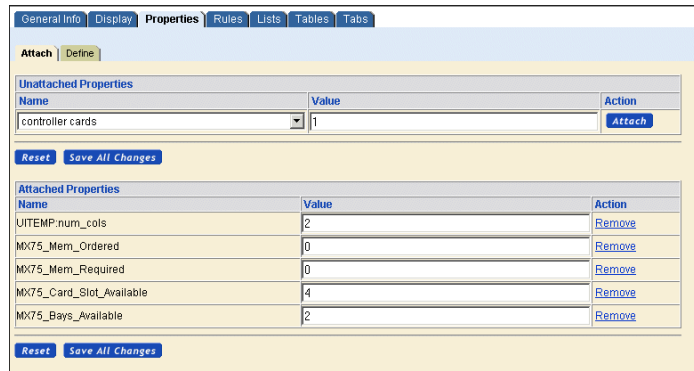


FIGURE 48. Properties Tab

4. Select a property from the Unattached Properties drop-down list.
The property will display any default value defined for it.
5. Enter a value for the property. You can set the value of a property simply by entering its value in the text field, or you can use a property editor window to set a value. See "To Use the Property Editor Window" on page 496 for details.
6. Click **Attach**.
The newly-attached property appears among the Attached Properties.
7. Click **Save All Changes**.

Attention: You must perform this last step. Otherwise the property will not be attached.

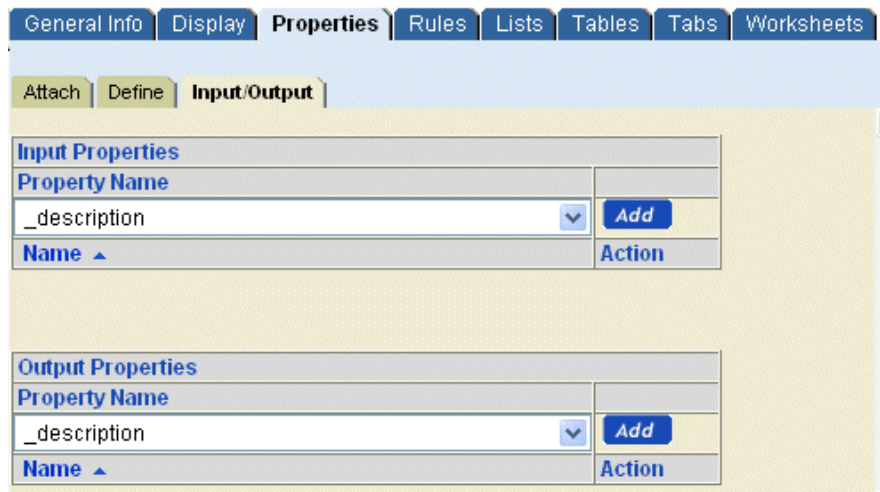


FIGURE 49. Input/Output Properties Tab

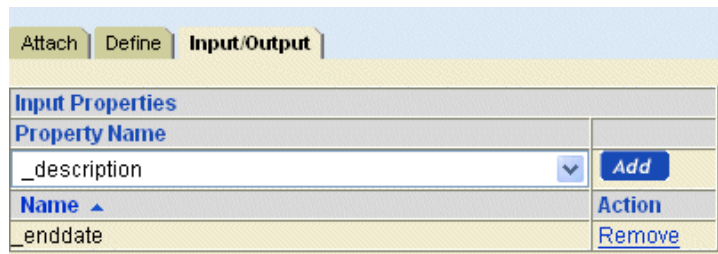


FIGURE 50. Input/Output Properties Tab Remove Button

To Use the Property Editor Window

The Numeric Property Editor window and the String Property Editor window are used to edit property values.

1. You can invoke the property window editor simply by clicking the **Edit** button next to any property.

When you do so, a Property Editor window is displayed.

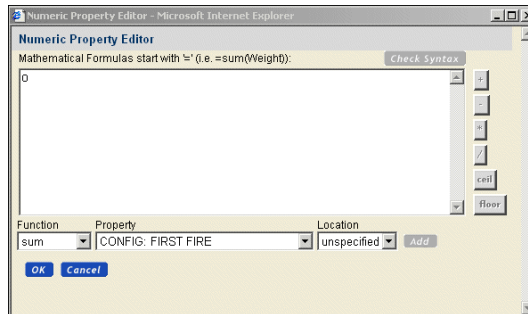


FIGURE 51. Numeric Property Editor Window

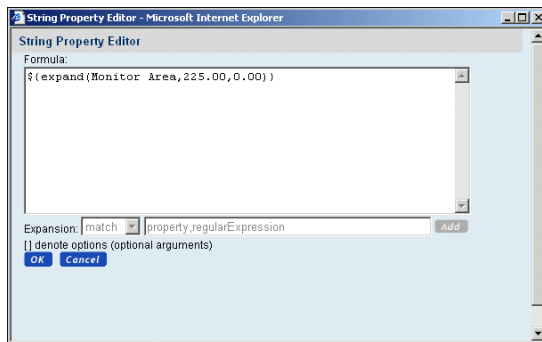


FIGURE 52. String Property Editor Window

2. You can use this window to specify a fixed value of a property or to specify a formula that is used to calculate a value at runtime. If the first character of the text area is "=", then the editor window assumes that you want to create a formula, and the expansion fields are activated to help you define the formula.
3. The syntax of a formula depends on whether you are editing a numeric or a string property:
 - a. If you are working on a numeric property, then when you specify a formula, use the drop-down lists as follows:
 - Function: select one of the defined functions.
 - Property: specify the property whose values should be used to calculate the function.

- Location: specify where the named property (or properties) should be located. You can select an option item or select one of the following values for the location:
 - unspecified: select this to use the named property anywhere it is defined in the model. First, the current position is checked to see if the property is defined at that location, if not, then the standard algorithm is followed to see if the property is defined anywhere else in the model.
 - relative: select this to use the named property at the current location.
- a. If you are working on a string property, then when you specify a formula, use the drop-down lists as follows:
 - Choose among gather, match, and expand:
 - gather: use this in assigning actions to a string property. It finds all occurrences of the specified property in the property pool and creates a string with the semicolon separating the values of these occurrences.
 - match: use this in writing rule fragments. It provides a mechanism to compare a string to the value of a property.
 - expand: use the expand function as described in "Working with Display Properties" on page 562.

To Modify or Remove an Attached Property

You can only modify the value of an attached property at the local level to which it is attached. To modify the name or default value, see "To Modify or Delete a Property Definition" on page 499.

1. Click **Configuration Models** in the Product and Catalog Administration panel on the Visual Modeler home page.
2. In the Model Groups frame, navigate to the element to which the property is attached.

If the property is attached to a model:

- a. In the Model Groups frame, click the model group that contains the model.
- b. In the Models and Groups frame, click the model to which the property is attached.
- c. Click **Edit** in the toolbar.

If the property is attached to an option class or option item:

- a. In the Model Groups frame, click the model group that contains either the model or group with the option class or option item.
 - b. In the Models and Groups frame, click the model or group.
 - c. Click **Edit** in the toolbar.
 - d. In the Navigation frame, find and click the option class or option item.
3. Click the **Properties** tab.

This tab displays two sets of fields: one called Unattached Properties for selecting properties and defining values for them, and one called Attached Properties that shows the properties that are currently attached.

Note: If the property is attached to a model, you will see two tabs within the Properties tab: **Attach** and **Define**. The **Attach** tab is automatically displayed.

4. Find the property you want to modify or remove.
5. Modify or remove the attached property:
 - If necessary, change the value of a property.

Note that this only changes the value locally, at the level it is attached. To change the default value of the property, see "To Modify or Delete a Property Definition" on page 499.
 - To remove an attached property, click **Remove**.
6. Click **Save All Changes**.

To Modify or Delete a Property Definition

1. Click **Configuration Models** in the Product and Catalog Administration panel on the Visual Modeler home page.
2. Navigate to and select the location in the model group hierarchy where the property is defined.
 - At the root model group level:

The Visual Modeler page automatically displays the root model group when you access Visual Modeler. If the root model group is not selected, then click on the root model group.
 - At the model group level, navigate to and click the model group in the Model Groups frame.

- At the model level, navigate to and click the model group that contains the model. Then, in the Models and Groups frame, click the model. Now click **Edit Model** in the toolbar.

In all of these cases, this displays the **General Info** tab for the group or model.

3. Click the **Properties** tab.

At the model group level, this displays the properties defined at that level.

At the model level, this displays two tabs: **Attach** and **Define**. If the property is attached anywhere in the model group hierarchy, you will not be able to modify the property type. If the property is attached anywhere in the model group hierarchy, you will not be able to delete the property definition.

4. If you want to modify an unattached property, click the **Define** tab. Within the **Define** tab, find the property you want to modify or delete.

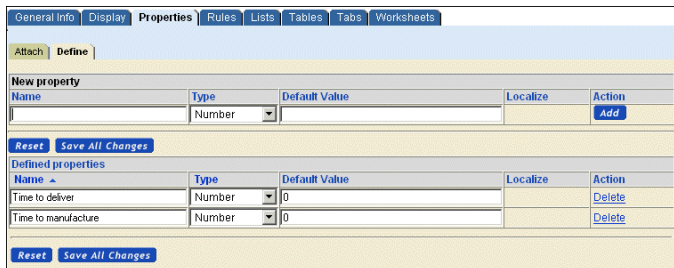


FIGURE 53. Model Properties Define Tab

5. If you want to modify an attached property, click the **Attach** tab. Within the **Attach** tab, find the property you want to modify or remove.

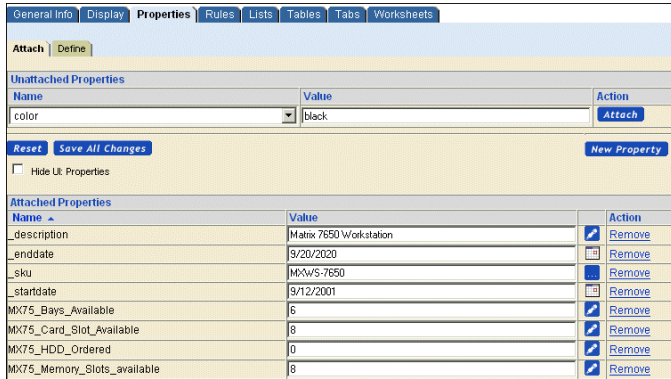


FIGURE 54. Model Properties Attach Tab

6. Modify or delete the property definition (property type or value).

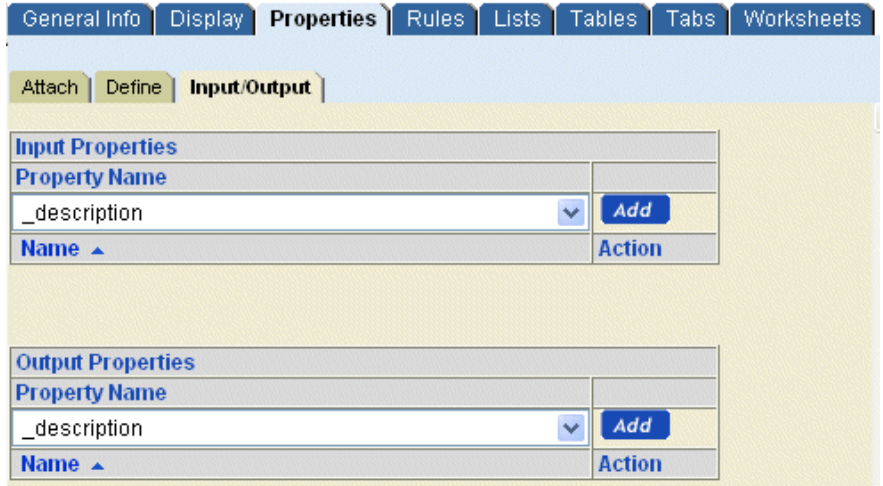


FIGURE 55. Model Properties Input/Output Tab

7. Click **Save All Changes**.

Name changes and value changes will be propagated to anywhere the property is attached. The value change is the default value for the property. It will not override any values set for the attached property.

Using Worksheets

Worksheets provide quick access to a group of properties, enabling you to easily maintain all of a model's properties in one place. A worksheet is a table that assigns property values to option items:

- Rows represent option items
- Columns represent properties

Each worksheet belongs to a model and can be used to set the values of properties of that model. You can still set the values for properties as described in "To Attach a Property" on page 494.

For example, suppose that a model of a computer has an option class for hard drives. Each hard drive option item has a number of properties such as capacity, RPM, latency, and buffer cache. You can create a worksheet to maintain the hard drive properties, similar to the following table:

TABLE 25. Hard Drive Worksheet

Option Item	Capacity	RPM	Latency	Buffer Cache
WD Protege	160	5400	5.00	2
WD Caviar	250	7200	4.20	2
WD Caviar SE	250	7200	4.20	8
WD Essential	250	7200	4.20	2

To Create a Worksheet

1. Navigate to the model for which you want to create a worksheet.
2. Click the Worksheets tab.
3. Click **New...**
4. In the New Worksheet window, enter a name for the worksheet, and click **Create**.
5. Add the option items whose properties you want to set using this worksheet. You do this by clicking **Add Row**, and then navigating to each option item in turn using the entity picker window.
6. Add the properties to the worksheet by clicking **Add Column** and in the Add Column dialog box, select each property from the drop-down list of properties defined for this model. You can create a new property by clicking

New Property in the Add Column window, and then entering the new property details in the Define New Property window.

7. After you have added the rows and columns for your worksheet, you can enter values for each option item and property.
8. Click **Save All Changes**.

To Modify a Worksheet

You can modify a worksheet at any time. Changes to property values are effective immediately, and will be compiled with the other model details when you next compile the model.

1. Navigate to the model to which the worksheet belongs.
2. Click the Worksheets tab.
3. Select the name of the worksheet form the drop-down list.
4. Click **Select**.
5. In the worksheet, you can do the following:
 - Change the name of the worksheet: click the worksheet name and enter a new name for the worksheet.
 - Add a new row: click **Add Row** and select option items as required.
 - Move a row: click the link to the row, and select its new position from the drop-down list of rows.
 - Remove a row: click the link to the row, and click **Delete**.
 - Add a new column: click **Add Column**, and select the property from the drop-down list.
 - Move a column: click the column name, and select its new position form the drop-down list of columns.
 - Delete a column: click the column name and click **Delete**.

To Export a Worksheet

There are times when it is more convenient to manage the values of properties when you have the worksheet in the form of a spreadsheet that you maintain on your local machine. You can export a worksheet in the form of a comma-separated values (CSV) file, and then open this file in your preferred spreadsheet program to manage the values. You can then import the modified spreadsheet to update the

values in the worksheet: see "To Import a Worksheet" on page 504 for details on importing a worksheet.

1. Navigate to the model to which the worksheet belongs.
2. Click the Worksheets tab.
3. Select the name of the worksheet from the drop-down list.
4. Click **Select**.
5. Click **Export...**
6. In the File Download window, click **Save**.
7. In the Save As window, navigate to the directory on your local machine to which you want to save the file, and then click **Save**.

The file is saved to your local machine.

To Import a Worksheet

When you have finished editing a spreadsheet for a worksheet, save it as a comma-separated values (CSV) file. Follow these steps to import the worksheet into the Visual Modeler.

1. Navigate to the model to which the worksheet belongs.
2. Click the Worksheets tab.
3. Click **Import...**
4. In the Worksheet Import window, click the **Browse...** button.
5. In the Choose File window, navigate to and select the spreadsheet that you want to import.
6. Click **Open**.
7. In the Worksheet Import window, click **Import Now**.

The spreadsheet is imported into the Visual Modeler.

Properties as Variables

You can evaluate the value of a property in defining rules and properties using this syntax: $\${function(...)}$. This enables you to define a property as a function of another property. This can be useful in defining display properties and in defining mathematical formulae for rules. For example, you can use $\${expand(property[,default[,format]])}$ to display properties of models.

For example, suppose that you have a numerical property called "Monitor Size" defined on a series of monitors that expresses the screen size in inches and suppose that you want to present this information in a table in the form "17.00 inches". You can define a property called Display Monitor Size by `"${expand("Monitor Size","n/a",0.00)} inches"`. Now use this new property in the display of the model and users will see the size expressed as "17.00 inches" if the underlying Monitor Size property has the value "17". Note that if the Monitor Size property is not defined, then "n/a inches" is displayed.

Visual Modeler Properties

The following table summarizes the properties that are built in to the Visual Modeler. Note that UI properties are covered in CHAPTER 38, "Visual Modeler UI Concepts".

TABLE 26. Visual Modeler Properties

Property	Type	Comments
		""""
CONFIG: FIRST FIRE	numeric	1 if this is the first time firing rules, 0 otherwise.
CONFIG: POOL SIZE	numeric	Number of copies of a model to keep in the model pool.
		Determines whether or not the price of a particular line item in a bill of materials is locked. Set CONFIG: PRICE LOCKED > 0 to lock the price; 0 to unlock the price.
CONFIG: REPEAT FIRING	string	"yes" or "true" turns on looping in the rule engine, causing rules to fire as long as the current state keeps changing. Since rules are removed from the rule list whenever they fire, this is not an infinite loop.
CONFIG: SUBMODEL NAME	string	The encoded name of another model. Encoding replaces potentially unsafe file system characters with <code>_XXXX</code> where XXXX is the hex representation of their Unicode character code. For example, a space is represented by <code>"_0020"</code> . See "Special Characters Encoding" on page 484 for more information.

TABLE 26. Visual Modeler Properties (Continued)

Property	Type	Comments
CONFIG: SUBMODEL RETURN	string	"yes" or "true" implies that when we punch into a submodel specified by the previous property we will be returning with that models BOM as a child of this model.
_cacheKey	string	Used on a model node to contain the key used to store the model in the model cache.
_description	string	The description of an item.
_errorCount	numeric	Number of errors encountered during rule firing.
_fileSize	string	String representation of a Long value, size of the XML file for a model.
_lastModified	string	Last modified date for a model as a string (number of seconds since some important date).
_modelTabs	list	List of tab names for the model.
_name	string	The name of an option item, option class, or model.
_parent.<item names>	varies	Properties inherited by a submodel from the parent.
_pickItems	list	Internally used to keep track of picked items.
_pickmap.<itemKey>	string	Mapping of an item to an option class.
_picks	list	Internally used to keep track of picked items.
_quantity	integer	Quantity selected, if >0 the item is picked.
_sequence	numeric	Rule firing sequence, if 0 this is the first time through the loop, 1 is the second, and so on.
_tabMembers<#>	list	Where <#> is a tab number (0...N), these properties contains the names of the root level option classes that are part of the tab whose index is <#>.

Lists

In many cases, the values a property may take can be expressed as a number or as a string of characters. In some cases however, a property has to take one of a certain number of pre-specified values such as the days of the week, or one of a set of manufacturer-specified formats such as SM, M, L, or XL.

In these situations, the best approach to take is to define a property, of List type. Then you can write rules to test whether the value of the first property is in the list that is the value of the List property.

Thus, if you have a property called `ShirtSize` and you want to restrict the choices that a user can select to SM, M, L, or XL, then the steps are:

1. Create a list called `ShirtSizeList`. Enter values for the list: in this case SM, M, L, and X.
2. Create a property called `AvailableShirtSizes` whose type is List and assign it the value `ShirtSizeList`.
3. Create the `ShirtSize` property and assign it to option items as appropriate.
4. Create a rule that specifies that the value of the `ShirtSize` property must be in the list of the `AvailableShirtSizes` property.

Working With Lists

To Define a List

1. Navigate to the model group or model for which you want to define the list.
See "To Access the Visual Modeler" on page 446 for information about how to navigate the model group hierarchy.
2. Click the **Lists** tab.
This displays any lists already defined.
3. Click **New...**
This displays the **New List** tab.

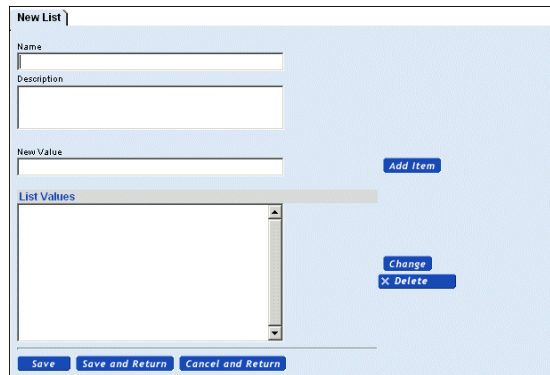


FIGURE 56. New List Tab

4. Enter a name and description for the list.
5. Define the values for the list.
 - a. Enter a value in the New Value field.
 - b. Click **Add Item**.
6. Repeat the last step for each value you want to add.
7. Click **Save** to save the values and remain at the **New List** tab.

When you click **Save and Return**, you save the values and return to the **Lists** tab. The new list appears among the defined lists.

To Modify a List

1. Navigate to the model which contains the list you want to modify.

See "To Access the Visual Modeler" on page 446 for information about how to navigate the model group hierarchy.
2. Click the **Lists** tab.

This displays any lists already defined.
3. Click the name of the list you want to modify.

This displays the **Edit List** tab.

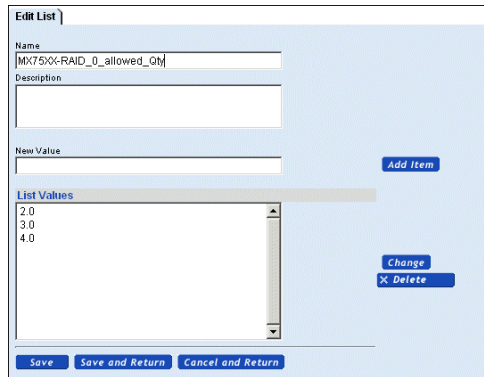


FIGURE 57. Edit List Tab

4. Modify the name or description.
5. Delete values from the list.
 - a. Select one or more values in the list.
 - b. Click **Delete**.
6. Add values to the list.
 - a. Type a value in the New Value field.
 - b. Click **Add Item**.
7. Modify values in the list.

There is no way to modify a value in a single step. You must delete the old value and add the new one.
8. Click **Save** to save the values and remain at the **Edit List** tab.

When you click **Save and Return**, you save your changes and return to the **Lists** tab.

To Delete a List

1. Navigate to the model which contains the list you want to delete.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. Click the **Lists** tab.

This displays any lists already defined.

3. Among the defined lists, find the list you want to delete.
4. Click **Delete** on the same line as the list you want to delete.

The list disappears from among the defined lists.

Attention: This last step is important! If you click Delete , but do not click Save All Changes , then the list will not be deleted.

5. Click **Save All Changes**.

Rules

Working With Rules

To Define a Rule

1. Navigate to the detail page for the model group or model where you want to create the rule.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.
2. Click the **Rules** tab.
 - a. If you are defining the rule at the model level, then the **Rules** tab displays two tabs: **Attach** and **Define**. Click the **Define** tab.
 - b. The model group level contains a single tab for defining the rule.

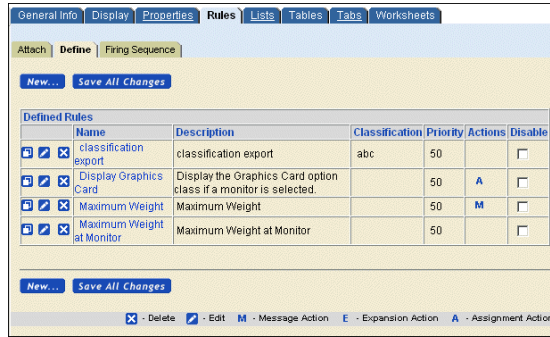


FIGURE 58. Define Tab for a Model Rule

3. Click **New...**

This displays the **New Rule** tab.

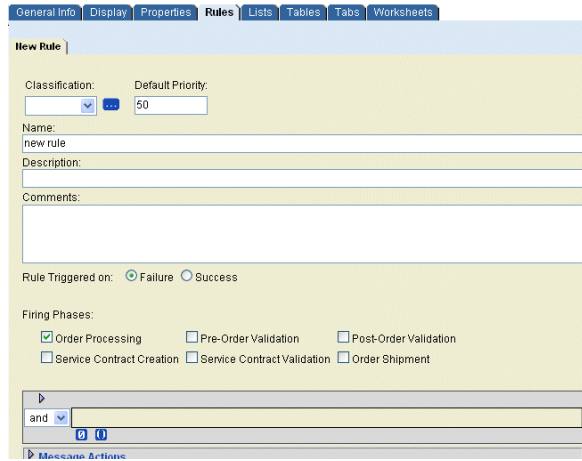


FIGURE 59. Model Page: New Rule Tab

FIGURE 60. Model Group Page: New Rule Tab

4. Select a classification for the rule and specify a priority.

You can create your own rule classifications: see "To Create a Rule Classification" on page 514. Rule priorities are used to determine the order in which rules are fired: lowest numbers fire first. You should use values between 0 and 100: 50 is the default value.

5. Enter a name and description for the rule. Also, select whether the rule is triggered when the rule's conditions are met (success) or not met (failure).
6. Define the fragments of the rule.

See "Fragments" on page 523.

7. Define the rule actions.

You can define messages to be displayed, a rule expansion formula, or you can assign properties and values. See "Working with Rule Actions" on page 537.

Note: No syntax checking is performed on rules. The configurator engine will fail to load a model if there is a syntax error in any of the assigned rules.

8. Click **Save**.

To Modify a Rule

1. Navigate to the model group or model where the rule was created.
See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.
2. Click the **Rules** tab.
The **Rules** tab for the Model level displays two tabs: **Attach** and **Define**. To modify the rule, click the **Define** tab. The model group level contains a single tab for defining the rule.
The **Rules** tab displays a table with the currently defined rules.
3. Find the rule you want to modify, then click the **Edit** icon.
The **Edit Rule** tab displays.
4. Modify the Name and Description as necessary.
5. Add or modify Comments as necessary.
6. Modify whether the rule is triggered when the rule's conditions are met (success) or not met (failure, as necessary).
- 7.

Edit Rule

Classification: Default Priority

60

Name:
ADV_up to RAID level 5

Description:
Sets option "RAID 5 Parity" on model, if end user selected answer "up to RAID level 5" in Questionnaire.

Comments:

Rule Triggered on: Failure Success

Fragments	If Not Specified	Actions
and	value(up to RAID level 5) = Yes	Rule is false

Message Actions

Error

Type Message

FIGURE 61. Edit Rule Tab

8. Modify the rule fragments in the Fragments table.
See "Fragments" on page 523.

9. Add or modify actions in the Actions area.

You can define messages to be displayed, a rule expansion formula, or assign properties and values. See "Working with Rule Actions" on page 537.

Repeat these steps for each rule you want to modify. You can click **Where Used** at the bottom of the tab to view the entities to which the rule is attached. See "To View Rule Attachments" on page 517.

To Create a Rule Classification

You can create new rule classifications as follows.

1. Navigate to the rule creation page: see "To Define a Rule" on page 510.
2. Click ... next to the Classification drop-down list.

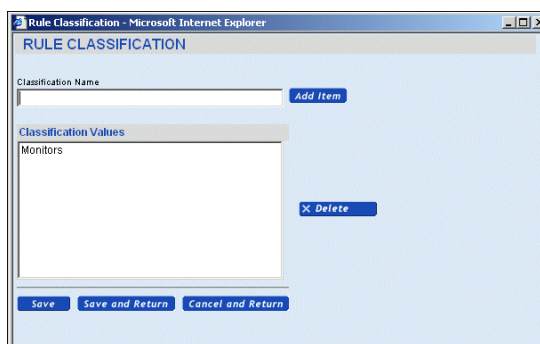


FIGURE 62. Rule Classification Window

3. In the Rule Classification Window, enter a name for the classification, and click **Add Item**.
4. Click **Save and Return**.

To Attach a Rule

1. Navigate to the level in the model hierarchy (model, option class or option item) where you want to attach the rule.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

- Click the **Rules** tab.

At the model level, the **Rules** tab contains two tabs: **Attach** and **Define**. At the option class and option item levels, the **Rules** tab looks like the **Attach** tab.

The **Attach** tab displays a drop-down list of the unattached rules, as well as a table showing the rules that are currently attached.

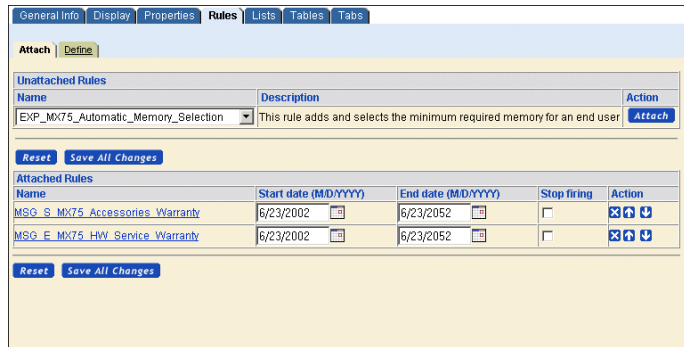


FIGURE 63. Rules Tab

- Select a rule from the drop-down list in the Unattached Rules table.
- Click **Attach**.

The rule is appended to the end of the current rules in the Attached Rules table.

- Define the start and end dates for the rule.
- If you want this rule to be a checkpoint, check the box the Stop Firing column.

When checked, this rule acts as a checkpoint: if any errors have occurred up to this point in the rule firing, then processing will stop at this point and the errors will be displayed. If no errors have occurred, then rule firing will continue until all the rules are fired or the next checkpoint is hit.

- Determine the sequence.

The rules will fire within the element to which they are attached in the order they appear in the list. You can modify the order using the up or down arrows to the right of the rule.

- Click **Save All Changes**.

To View the Details of an Attached Rule

Once you have attached a rule, you can view the details of the attached rule by clicking the rule's name in the **Attach** tab.

1. Navigate to the level in the hierarchy (model, option class, or option item) where the rule is attached.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. Click the **Rules** tab.

At the model level, the **Rules** tab contains two tabs: **Attach** and **Define**. At the option class and option item levels, the **Rules** tab looks like the **Attach** tab.

The **Attach** tab displays a drop-down list of the unattached rules, as well as a table showing the rules that are currently attached.

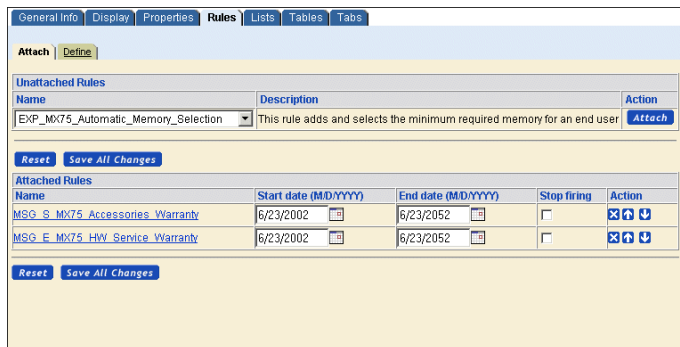


FIGURE 64. Attach Tab

3. Find the rule among the list of attached rules in the lower part of the frame.
4. Click the name of the rule.

This displays the Rule Detail Viewer.

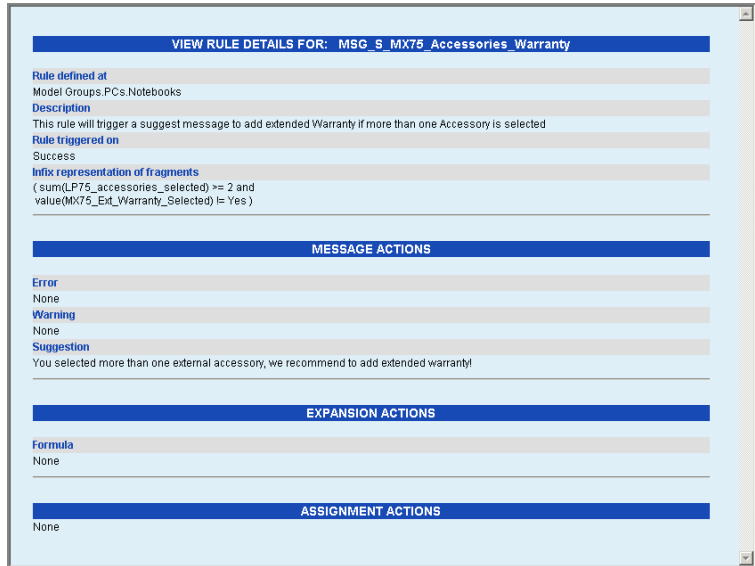


FIGURE 65. Rules Detail Viewer

To View Rule Attachments

You can use this procedure to see where a rule is attached.

1. Navigate to the model group or model where the rule was created.
See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.
2. Click the **Rules** tab.
The **Rules** tab for the Model level displays two tabs: **Attach** and **Define**. To modify the rule, click the **Define** tab. The model group level contains a single tab for defining the rule.
The **Rules** tab displays a table with the currently defined rules.
3. Find the rule you want to modify, then click the **Edit** icon.
The **Edit Rule** tab displays.

Edit Rule

Classification: Default Priority:

Name:
 ADV_up to RAID level 5

Description:
 Sets option "RAID 5 Parity" on model, if end user selected answer "up to RAID level 5" in Questionnaire.

Comments:

Rule Triggered on: Failure Success

Fragments	If Not Specified	Actions
and value(up to RAID level 5) = Yes	Rule is false	

Message Actions

Type	Message	Delete
Error		

FIGURE 66. Edit Rule Tab

4. Click **Where Used...**

The Rule Usage window displays.

VIEW RULE USAGE FOR: MSG_E_MX75_Memory_Software_Check.		
Rule Usage		
Model Group	Model Name	Attached At
Workstations	MXWS-7650	MXWS-7650.Software
Workstations	MXWS-7550	MXWS-7550.Software
PCs.Desktops	MXDS-7500	MXDS-7500.Software
PCs.Desktops	MXDS-7500	MXDS-7500.Memory
PCs.Desktops	MXDS-7550	MXDS-7550.Memory
PCs.Desktops	MXDS-7550	MXDS-7550.Software
Workstations	MXWS-7550	MXWS-7550.Memory
Workstations	MXWS-7650	MXWS-7650.Memory
PCs.Desktops	MXDS-7550	MXDS-7550

FIGURE 67. Rule Usage Window

To Unattach a Rule

1. Navigate to the level in the model hierarchy (model, option class or option item) where the rule is attached.

See "To Access the Visual Modeler" on page 446 for information about how to navigate the model group hierarchy.

2. Click the **Rules** tab.

At the model level, the **Rules** tab contains two tabs: **Attach** and **Define**. At the option class and option item levels, the **Rules** tab only contains attachments.

The Rules tab displays a drop-down list of the unattached rules, as well as a table showing the rules that are currently attached.

3. Find the rule in the Attached Rules table.
4. Click the **Delete** symbol (**X**) at the end of the rule's row in the table.

The rule returns to the Unattached Rules table.

5. Click **Save All Changes**.

To Delete a Rule

You can delete rules when they are no longer required.

Note: You cannot delete a rule if the rule is currently attached to any node in the model hierarchy.

1. Navigate to the model group or model where you created the rule.

See "To Access the Visual Modeler" on page 446 for information about how to navigate the model group hierarchy.

2. Click the **Rules** tab.

At the model level, the **Rules** tab contains two tabs: **Attach** and **Define**. At the model group level, you can only define. If the rule you want to delete was created at the model level, then click the **Define** tab.

3. Find the rule you want to delete.
4. Click the **Delete** icon next to the rule you want to delete.
5. Click **Save All Changes**.

To Move or Copy a Rule

It is sometimes necessary to re-organize your model hierarchy and in doing so, you may need to move the rule definitions too. You can move or copy a rule: *moving* means that the rule definition is deleted from its previous location whereas *copying* means that you create a copy of the rule without deleting its original definition.

1. Navigate to the rule definition that you wish to move or copy.
2. Click **Copy Rule**.

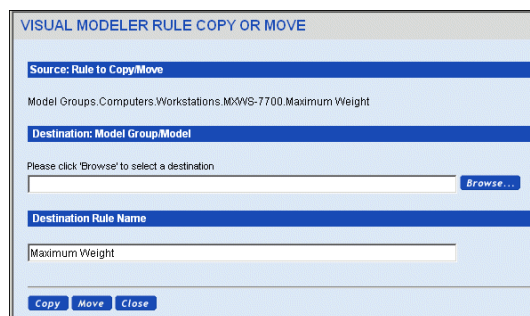


FIGURE 68. Rule Copy or Move Window

3. Click **Browse...** to open the entity picker window.
4. Navigate to the model group or group to which you want to move or copy the rule definition, select it and click **Done**.
5. If you want to, you can then change the name of the rule definition.
6. Click **Move** or **Copy** as appropriate.

An error message is displayed if a rule with the same name already exists in the target location. If a property referenced in the rule does not exist in the new location, then it is created at the same time as the rule is.

7. Click **Close**.

Rule Firing

Each time a model is validated, the rules are fired to determine whether each rule succeeds or fails. You can control the order in which rules fire by setting a priority for each rule: see "To Specify the Rule Firing Sequence" on page 520. When you are testing a model, you can review the model firing behavior: see "To Review Rule Firing" on page 521. You can also specify whether rules are fired just once or possibly multiple times: see "To Force Multiple-Pass Rule Testing" on page 523.

To Specify the Rule Firing Sequence

1. Navigate to the model.
2. Click the Rules tab.
3. Click the Firing Sequence sub-tab.

Attach Define Firing Sequence			
Reset Save All Changes			
Attached Rules			
Priority	Classification	Name	End date
50		ASG Service offerings visible	MX-7500 Service.Warranty
50		MSG_E_MX75_HW_Service_Warranty	MX-7500 Service.Hardware
50		ASG Service offerings visible	MX-7500 Service.Hardware
50		MSG_E_Available_Card_Slots_Check	MXDS-7500.Accessory Car
50		EXP_MX75_Fire_Wire	MXDS-7500.Software Appli
50		MSG_VV_Num_OS_Check	MXDS-7500.Software

FIGURE 69. Firing Sequence Tab

4. Enter a priority for each rule: this should be an integer between 0 and 100.

The higher the value the lower the priority: that is rules with lower priority value will fire before rules with a higher priority value. The default value is 50.

To Review Rule Firing

1. Navigate to the model whose rule firing you want to review.
2. Click **Test**.
3. In the Product Configurator window, click **Show Trace Log**.

#	Result
0	Firing phase [0]:begin
1	Firing rules on MXWS-7550.Memory
2	
3	MSG_E_max_memory_slots_exceeded ==> fires on FALSE
4	Property not found [MX75_Memory_Slots_available or MX75_Memory_Slots_required], taking null action
5	
6	MSG_E_MX75_Memory_Software_Check ==> fires on TRUE
7	Property not found [MX75_Mem_Ordered or MX75_Mem_Required], taking null action
8	
9	Firing rules on MXWS-7550.Disk Drives
10	
11	MSG_E_Available_HDD_Slots ==> fires on TRUE
12	TESTING:sum(MX75_HDD_Ordered) >value(MX75_Bays_Available) [nullreturn=false]
13	FALSE: 1.0>MX75_Bays_Available:4.0
14	FALSE: sum(MX75_HDD_Ordered) >value(MX75_Bays_Available) [nullreturn=false]
15	
16	Firing rules on MXWS-7550.Software.Application
17	
18	EXP_MX75_Fire_Wire ==> fires on TRUE
19	Left side property [MX75_Video_Editing] not found, taking null action
20	

FIGURE 70. Configurator Rule Firing Trace Window

- The secondary window displays a trace of the results of the rule firing. You can review this to determine if rules are firing as you expect.

Controlling Rule Firing

When Sterling Configurator validates a model and the current set of picks, it tests each rule in turn to evaluate them for success or failure, and performs expansion and assignment actions as appropriate. There are different ways in which rule firing can behave:

- Each rule is tested only once when a model is validated. This is referred to as single-pass rule firing.
- You can configure a model so that if any rules are fired, the fired rules are removed from the rules list and the remaining rules are tested again. This

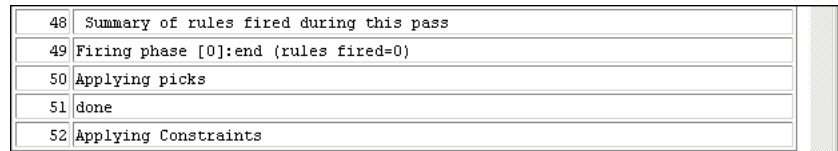
process continues until no more rules are fired. This is referred to as multiple-pass rule firing.

The property CONFIG: REPEAT FIRING controls this behavior. By default, only single-pass firing is performed.

To Force Multiple-Pass Rule Testing

1. Navigate to the model whose rule firing you want to control.
2. Click **Properties**.
3. Select CONFIG: REPEAT FIRING from the **Unattached Properties** drop-down list.
4. Set its value to “true” and click **Attach**.
5. Click **Save All Changes**.

You can verify that the rules fire only once by following the steps described in "To Review Rule Firing" on page 521. In the summary section of the trace log, you should see that there was one firing phase.



48	Summary of rules fired during this pass
49	Firing phase [0]:end (rules fired=0)
50	Applying picks
51	done
52	Applying Constraints

FIGURE 71. Trace Log Summary Section

Fragments

As you create rules, you must create rule fragments that perform the rule logic. This section describes in detail how to create and work with rule fragments.

Working With Rule Fragments

This section describes the procedures to define or modify the fragments of a rule when you are creating ("To Define a Rule" on page 510) or modifying ("To Modify a Rule" on page 513) a rule. The Fragment area appears as in Figure 72 on page 524.

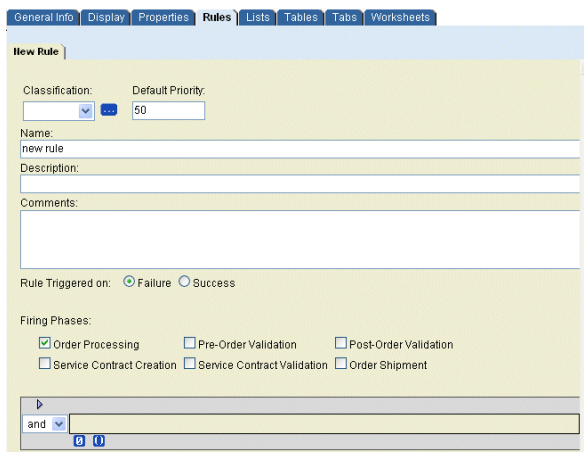



FIGURE 72. New Rule and Defining Fragments

Click the arrow icon, , to toggle the visibility of sections of the Fragments area and enable working with them. For example, click the arrow to display the foreach section:

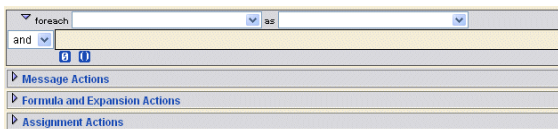


FIGURE 73. New Rule and Foreach Section Display

The following table describes the buttons that display in the Fragments area of the New Rule tab.

TABLE 27. Rule Buttons





Button	Name
	New Operator button
	New Fragment button

TABLE 27. Rule Buttons (Continued)

Button	Name
	Delete button
	Edit button

Click the **New Operator** icon to create a nested level for creating fragments, as shown in Figure 74 on page 526. The new level displays a new set of **New Fragment** and **New Operator** links. You use this **New Fragment** link to create the fragments at this nested level. If you click **New Operator** at this level, then you will create another nested level below this one with another set of **New Fragment** and **New Operator** links for that level.

Click the **New Fragment** icon to create a fragment at the currently displayed level (in this case, the top level) in the rule structure. Click **New Fragment** again to create a second fragment at the currently displayed level. In other words, the rule would be:

FragmentA AND FragmentB

Click the **Delete** button to delete the fragment.

Click the **Edit** button to modify the fragment.

The screenshot shows the 'New Rule' dialog box with the following sections:

- General Info:** Name, Description, and Comments text boxes.
- Rule Triggered on:** Radio buttons for Failure (selected) and Success.
- Fragments:** A visual tree structure showing nested 'and' conditions. The first level has two 'and' nodes, and the second level has three 'and' nodes. A toolbar below includes Delete, Edit, New Operator, and New Fragment buttons.
- Message Actions:** A table with columns for Type (Error, Message) and an Add Item button.
- Formula:** A text box for entering a formula.
- Expansion Actions:** A table with columns for Min, Max, Qty, and Item, with a Browse... button and a Delete button.
- Assignment Actions:** A table with columns for Property, Value, Assign To, and Delete, with a Browse... button and an Add Item button.
- Buttons:** Save, Save and Return, and Cancel and Return buttons at the bottom.

FIGURE 74. New Rule Tab with Nested Levels

Foreach

A rule condition can have a property called the foreach property. Use the foreach property to loop through the property pool, identify all instances of a specified property, and act on a set of found values. You associate the foreach property with a property defined in the model. Each occurrence of the foreach property is bound to an "as" property.

For example, to increase the price of any node found selected in the model that has the SKU MXDS-7500 and for which the rackMountable property is true, you could use foreach in the following way:

```
foreach sku as tempSku
  IF value(tempSku) == literal(MXDS-7500)
  AND propval(itemType) == literal("rackMountable")
  THEN UI: PRICE = value(UI: PRICE) * 1.1
```

Example: To Create a Simple Level of Fragments

In this example, you are attempting to create a rule consisting of two fragments, joined by a single operator, with no nested levels: FragmentA AND FragmentB. When you access the tab, the **New Rule** tab appears as in Figure 75 on page 527.

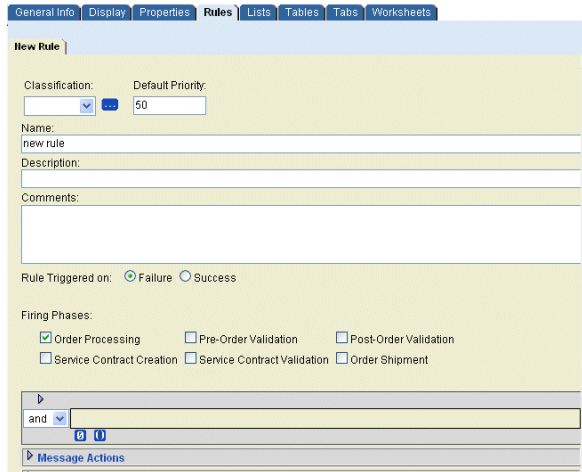


FIGURE 75. New Rule Tab for a Single-Level Rule

1. Select the boolean operator you want for these fragments.
2. Click the **New Fragment** icon

This displays the **New Fragment** tab.

FIGURE 76. New Fragments Tab

3. Define the fragment.
 - a. Check the **Not** check box if you want to define the fragment as a negative: "NOT (sum(PropertyA <= 250))".
 - b. Select the first function from the Function1 drop-down list.
 - c. Select a property from the Property1 drop-down list.
 - d. Select the operator.
 - e. Select the second function from the Function2 drop-down list.
 - f. In the Property2 field, select a property from the drop-down list or enter a literal value in the field (if you selected "literal" as the function).
 - g. Select the value for the If Not Specified drop-down list.
4. Click **Save and Return**.

This re-displays the **New Rule** tab with the new fragment, as shown in Figure 77 on page 529. Also, notice the infix representation.

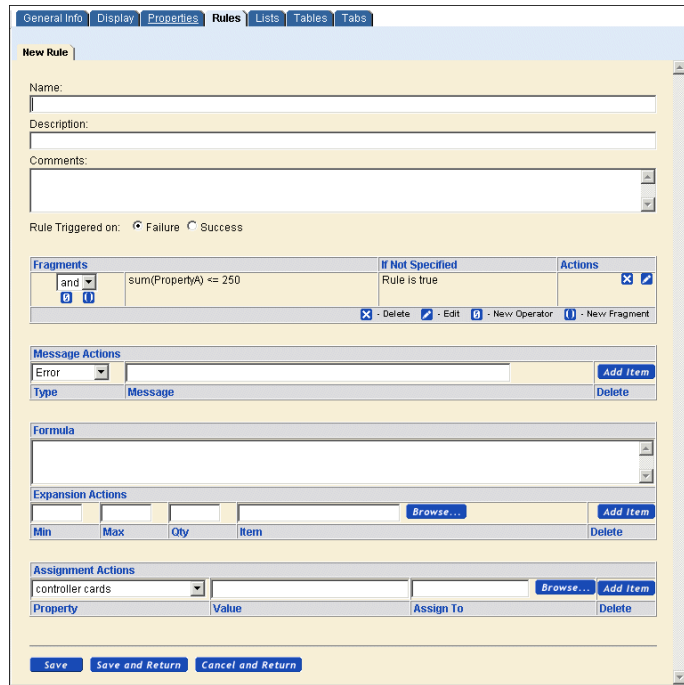


FIGURE 77. New Rule Tab with New Fragment

5. Click the **New Fragment** icon to create the next fragment in the rule.
6. Repeat Step 3 to define the second fragment.
7. Click **Save and Return**.

This re-displays the **New Rule** tab with the fragment you created, as shown in Figure 78 on page 530. Notice that the rule has two fragments now.

You can click **Save** to save the rule and continue defining the rule. Click **Save and Return** to return to the list of rules in the **Define** tab.

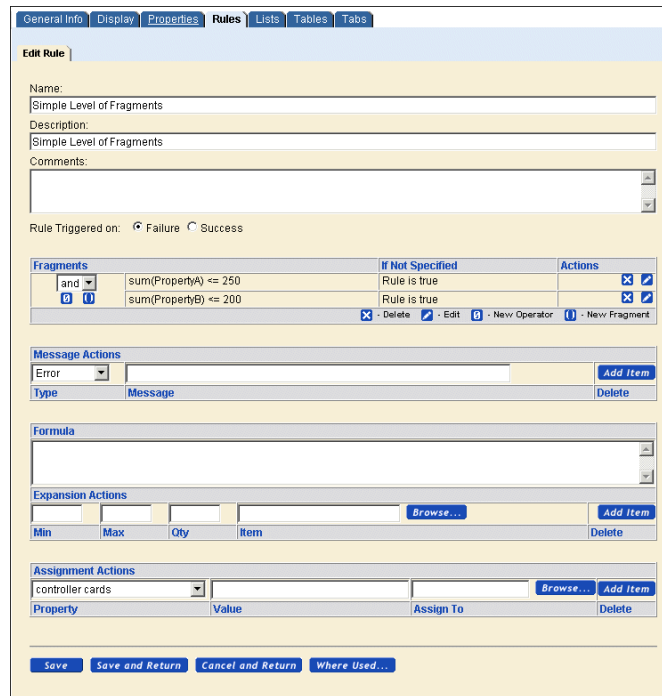


FIGURE 78. New Rule Tab with Two New Fragments

Example: To Create Nested Fragments

In this example, the modeler is creating the following rule with nested fragments:

(FragmentA AND FragmentB) OR (FragmentC AND FragmentD)

The screenshot shows the 'New Rule' tab in a software application. The tab is active and displays the following fields and options:

- Classification:** A dropdown menu.
- Default Priority:** A text box containing the value '50'.
- Name:** A text box containing the value 'new rule'.
- Description:** A text box.
- Comments:** A large text area.
- Rule Triggered on:** Radio buttons for 'Failure' (selected) and 'Success'.
- Firing Phases:** A group of checkboxes:
 - Order Processing
 - Pre-Order Validation
 - Post-Order Validation
 - Service Contract Creation
 - Service Contract Validation
 - Order Shipment
- Rule Editor:** A section at the bottom with a dropdown menu set to 'and' and two empty slots for fragments.

FIGURE 79. New Rule Tab

1. Click **New Operator** icon.

A nested level appears in the **New Rule** tab, as shown in Figure 80 on page 532. This level has its own drop-down boolean operators, as well as its own **New Fragment**, **New Operator**, and **Delete Operator** icons.

FIGURE 80. New Rule Tab with Nested Fragments

2. Create two fragments as described in "Example: To Create a Simple Level of Fragments" on page 527.

Use the nested drop-down list to select the boolean operator for these fragments. The default is AND.

Use the nested **New Fragment** icon to create the fragments at this nested level.

When the two fragments are completed, the **New Rule** tab appears as in Figure 81 on page 533.

You can nest as many fragments as you want by clicking the nested **New Operator** icon. Each time, a new nested operator will appear with a new set of nested icons. You use these nested icons to create the fragments for the nested level.

The screenshot shows the 'New Rule' configuration window. The 'Rules' tab is selected. The rule name is 'Nested Level of Fragments'. The 'Rule Triggered on' is set to 'Failure'. The 'Fragments' section contains two nested conditions: 'sum(PropertyA) <= 250' and 'sum(PropertyB) <= 200', both with 'Rule is true' as the 'If Not Specified' condition. The 'and' operator is selected between the two fragments. The 'Message Actions' section is set to 'Error' with a message type of 'Message'. The 'Expansion Actions' section is empty. The 'Assignment Actions' section is set to 'controller cards' with a value field and an 'Assign To' field. The 'Save' button is highlighted.

FIGURE 81. New Rule Tab with Nested Fragments

- Using the top-level list, select the boolean operator (in this example, OR) that will join the two nested levels.

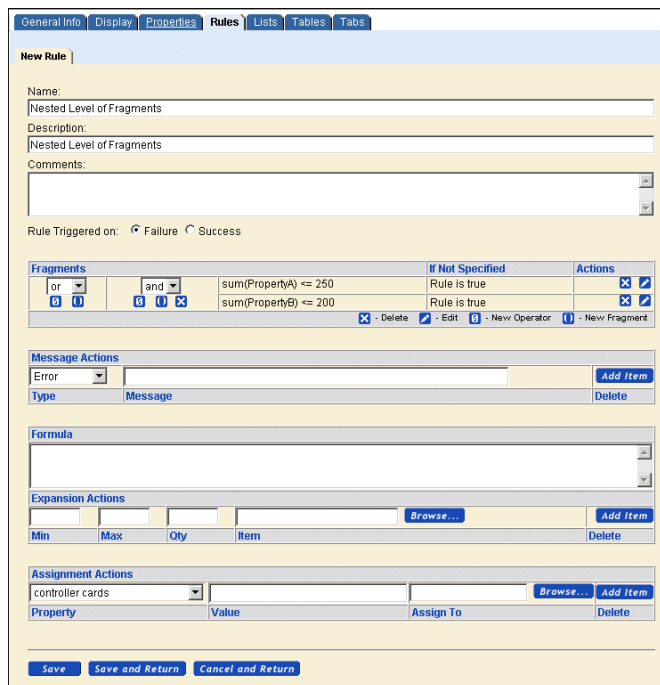


FIGURE 82. Nested Fragments with OR Boolean

4. Click the **New Operator** icon at the top level.

A new nested level appears in the fragments tab, as shown in Figure 83 on page 535. This level has its own drop-down boolean operators, as well as its own **New Fragment**, **New Operator**, and **Delete Operator** icons.

FIGURE 83. Nested Fragments

5. Create two fragments as described in "Example: To Create a Simple Level of Fragments" on page 527.

Use the nested drop-down list of boolean operators and the nested **New Fragment** icon for these fragments.

When the two fragments are completed, the **New Rule** tab appears as in Figure 84 on page 536.

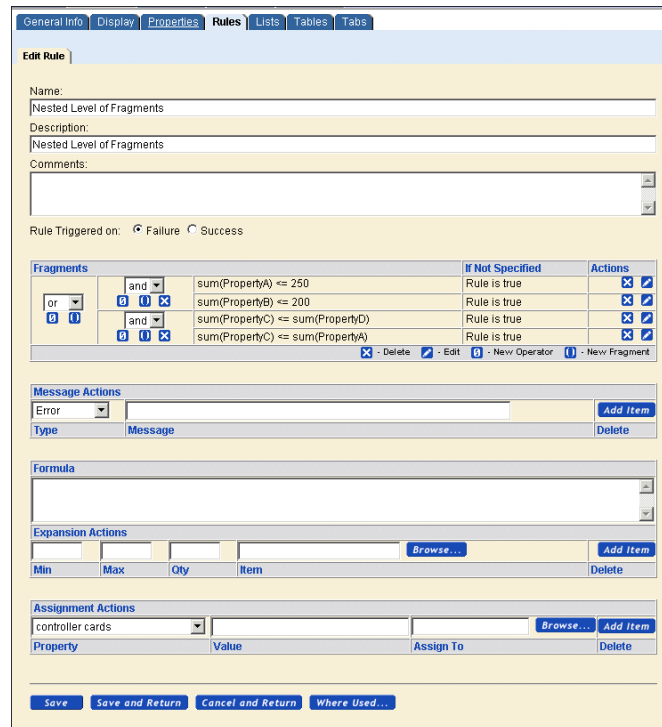


FIGURE 84. Two Nested Fragments Joined by OR

To Modify a Fragment

1. Find the fragment you want to modify, then click on the **Edit Fragment** icon in the Actions column.

This displays the **Edit Fragment** tab.

FIGURE 85. Edit Fragment Tab

2. Modify one or more elements of the fragment.
 - a. Check the **Not** check box if you want to define the fragment as a negative: “NOT (sum(PropertyA <= 250))”.
 - b. Select the first function from the Function1 drop-down list.
 - c. Select a property from the Property1 drop-down list.
 - d. Select the operator.
 - e. Select the second function from the Function2 drop-down list.
 - f. In the Property2 field, select a property from the drop-down list or enter a literal value in the field (if you selected “literal” as the function).
 - g. Select the value for the If Not Specified drop-down list.
3. Click **Save And Return**.

To Delete a Fragment

Find the fragment you want to delete in the Fragments table, then click the **Delete** icon in the Actions column on the same line as the fragment.

Working with Rule Actions

Perform these tasks when you want to include a rule action when you are creating (“To Define a Rule” on page 510) or modifying (“To Modify a Rule” on page 513) a rule.

You define rule actions in the lower part of the **New Rule** or **Edit Rule** tab. Rule actions comprise three types of actions:

- **Message Actions:** A message that is displayed when the rule is triggered.
- **Formula and Expansion Actions:** Defines an expansion action based on a rule expansion formula.
- **Assignment Actions:** Assigns the value calculated by the rule formula to one or more properties when the rule is triggered.

The screenshot shows the 'Edit Rule' tab with the following sections:

- Name:** Simple Level of Fragments
- Description:** Simple Level of Fragments
- Comments:** (Empty text area)
- Rule Triggered on:** Failure Success
- Fragments:**

Fragment	If Not Specified	Actions
and	Rule is true	[X] [✓]
sum(PropertyA) <= 250	Rule is true	[X] [✓]
sum(PropertyB) <= 200	Rule is true	[X] [✓]
- Message Actions:**

Error	Message	Actions
[Dropdown]	[Text]	[Add Item]
Type	Message	Delete
- Formula:** (Empty text area)
- Expansion Actions:**

Min	Max	Qty	Item	Actions
[Input]	[Input]	[Input]	[Input]	[Browse...] [Add Item]
				Delete
- Assignment Actions:**

Property	Value	Assign To	Actions
controller cards	[Input]	[Input]	[Browse...] [Add Item]
Property	Value	Assign To	Delete

Buttons at the bottom: Save, Save and Return, Cancel and Return, Where Used...

FIGURE 86. New Rule Tab Showing Action Area

To Create a Message Action

When you are creating ("To Define a Rule" on page 510) or modifying ("To Modify a Rule" on page 513) a rule, you perform this task in the Message Actions area of the **New Rule** or **Edit Rule** tab.

1. Select the type of message action from the drop-down list: Error, Warning, Suggestion.
2. Type the message.
3. Click **Add Item**.
4. Repeat these steps to enter additional messages.
5. Click **Save** to save the message action and continue editing. Click **Save and Return** to save the message and return to the **Define** tab.

The screenshot shows the 'Edit Rule' tab with the following sections:

- Name:** Simple Level of Fragments
- Description:** Simple Level of Fragments
- Comments:** (Empty text area)
- Rule Triggered on:** Failure Success
- Fragments:**

Fragments	If Not Specified	Actions
and	sum(PropertyA) <= 250	Rule is true
	sum(PropertyB) <= 200	Rule is true
- Message Actions:**

Type	Message	Actions
Error		Add Item
Error	We recommend you order more memory!	Delete
- Formula:** (Empty text area)
- Expansion Actions:**

Min	Max	Qty	Item	Actions
				Browse... Add Item
- Assignment Actions:**

Property	Value	Assign To	Actions
controller cards			Browse... Add Item

Buttons at the bottom: Save, Save and Return, Cancel and Return, Where Used...

FIGURE 87. Edit Rule Tab with Message Action

To Create an Expansion Action

When you are creating ("To Define a Rule" on page 510) or modifying ("To Modify a Rule" on page 513) a rule, you perform this task in the Expansion Actions area of the **New Rule** or **Edit Rule** tab.

1. Enter a formula.

The results of formula will be used to perform the expansion.

Edit Rule

Name: EXP_MX75_Automatic_Memory_Selection

Description: This rule adds and selects the minimum required memory for an end user

Comments:

Rule Triggered on: Failure Success

Fragments	If Not Specified	Actions
and value(MX75_Mem_Auto_Select) = Yes	Rule is false	<input type="checkbox"/> <input checked="" type="checkbox"/>

Message Actions

Type	Message	Actions
Suggestion	RAM modules are autoselected based on your software choi	<input checked="" type="checkbox"/>

Formula

sum(MX75_Mem_Required) - sum(MX75_Mem_Ordered)

Expansion Actions

Min	Max	Qty	Item	Actions
128	256	1	*AutoMemory.Auto Memory.256MB	<input checked="" type="checkbox"/>
0	64	1	*AutoMemory.Auto Memory.64MB	<input checked="" type="checkbox"/>
64	128	1	*AutoMemory.Auto Memory.128MB	<input checked="" type="checkbox"/>
256	512	2	*AutoMemory.Auto Memory.256MB	<input checked="" type="checkbox"/>

Assignment Actions

Property	Value	Assign To	Actions
isVisibleable	1		<input checked="" type="checkbox"/>

Save Save and Return Cancel and Return Where Used...

FIGURE 88. Expansion Action

2. Enter a minimum and a maximum amount of the formula result.

The minimum amount is the minimum value the rule formula result must be greater than. This value can be negative or greater than or equal to zero. The value must be less than the maximum value (Max). The maximum amount is the maximum value the rule formula result must match. This value must be greater than the minimum value (Min).

Note: Min and Max work slightly differently: for a fragment to evaluate to true, the rule formula must evaluate to greater than the Min value, but less than *or equal* to the Max value.

- Enter the quantity of the expansion items (must be greater than zero). You can use the supported functions to calculate the quantity and so you can specify the quantity as a function of a property. For example:

2*value(Memory Cards)

- Enter the item that will be expanded.

You must provide the full path to the expansion item within the current model. In the figure above, for example, the rule adds an option item called either 64MB, 128MB, or 256MB, located in option class AutoMemory in the current model.

Formula			
value(Expansion Cards)			
Expansion Actions			
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/> Browse...
Min	Max	Qty	Item
0	10	3*value (Expansion Cards)	MXWS-7550.Memory.64MB

FIGURE 89. Expansion Action With Example of Using Quantity Function

When a rule is used in multiple models, this fully qualified path could be difficult to specify since the current model name will very likely not be "MXWS-7650" for all the models where the rule is attached. To facilitate the use of expansion rules across multiple models, you can use special symbols as follows:

- You can begin the path with a period (.), which means “from the attachment point of the rule”. In other words, if you attach a rule to a model, then ".Memory.64MB" means "an option item called 64MB in an option class called Memory in the current model".
- You can begin the path with an asterisk (*), which means from the root of the model group hierarchy.
- If the name of a path component includes a quote character (' or "), then you must escape the quote character or wrap the whole expression in quotes. For example, to get the gauge property from the Tubing.3" pipe.threading option item, you can use

```
x = value(Tubing.3\"pipe.threading.gauge")  
OR  
x = value('Tubing.3\"pipe.threading.gauge')  
To retrieve Board.8'plank.thickness, use  
x = value(Board.8\"plank.thickness)  
OR  
x = value("Board.8'plank.thickness")
```

5. Repeat these steps to enter additional items.
6. Click **Save All Changes**.

The result of an expansion action picks a quantity selected on an option item. If the option item quantity is a drop-down list, ensure that the possible calculated values are consistent with the pickable values: otherwise, the drop-down list will not be able to display the calculated value.

To Create an Assignment Action

When you are creating ("To Define a Rule" on page 510) or modifying ("To Modify a Rule" on page 513) a rule, you perform this task in the Assignment Actions area at the bottom of the **New Rule** or **Edit Rule** tab.

The screenshot shows the 'New Rule' dialog box with the following sections:

- Name:** Assignment Rule
- Description:** Assignment Rule
- Comments:** (empty text area)
- Rule Triggered on:** Failure Success
- Fragments:** and (operator), If Not Specified (checkbox), Actions (checkbox)
- Message Actions:** Error (dropdown), Add Item (button), Type: Message, Delete (button)
- Formula:** (empty text area)
- Expansion Actions:** Min, Max, Qty, Item (input fields), Add Item (button), Delete (button)

FIGURE 90. Assignment Actions

1. Select a property from the drop-down list. The table below summarizes some of the special properties that can be assigned.
2. Enter a value for the property. You can use the supported functions to calculate the value and so you can specify the value as a function of a property. For example:

2*value(Memory Cards)

When you are assigning a value to a property whose type is String, you must use the following syntax to refer to properties:

$\${function(arg1, arg2, \dots, arg N)}$

For example, $\${expand("Color", "Black", 0)}$. See "Example Uses of Expand" on page 544 for other examples of the usage of the expand function.

3. Type the entity to which you want to assign the property and its value.
If you leave this field blank, the assignment defaults to the entity to which the rule is attached.
4. Click **Add Item**.

Assignment Actions		
Property	Value	Assign To
CONFIG: FIRST FIRE		Browse...
Expansion Slots	2*value(Expansion Cards)	MXWS-7550

FIGURE 91. Assignment Action With Example of Using Quantity Function

5. Repeat these steps to add additional items.
6. Click **Save All Changes**.

The following table summarizes some of the available properties for assignment. These properties may change in each release, so check with your Sterling Commerce representative for further information if required.

TABLE 28. Assignment Action Properties

Property	Action
_constraintMessage	String: a message on an item because it is constrained
_constraintType	Integer: type of constraint; 0 is suggest, 1 is warn, and 2 is error
_description	String: an items description

TABLE 28. Assignment Action Properties

Property	Action
_amEntitled	Integer: 0 false, 1 true
_isConstrained	Integer: 0 false, 1 true
_isSelected	Integer: 0 false, 1 true
_isVisibleable	Integer: 0 false, 1 true
_itemKey	Integer: database key of the item
_pickOverride	Integer: 0 false, 1 true; pick was overridden by a rule
_quantity	Integer: quantity; 0 quantities are not in the rule pool
_ratio	Numeric: ratio of this item to its children, computed if nested within another parent
_rawRatio	Numeric: raw ratio used in previous computation
_rulePick	Integer: 0 false, 1 true
_tabLevel	Integer: depth of this item

Example Uses of Expand

The syntax of the expand function is:

```
#{expand(property[,defaultValue[,format[,lookup]])}
```

For example, suppose that you want to display the name of the model as the name of the associated product together with the product description. At the model level, set the value of the UI: DISPLAY NAME property to: `#{expand("UI: PRODUCT NAME")}` or `#{expand("UI: PRODUCT DESCRIPTION","Description not available")}`.

Doing this ensures that if the product name or description changes and you recompile the model, the name or description displays with the new version when users next configure the product.

Here are some further examples of the expand function:

- String-valued property:
 - `#{expand("color")}`
 - `#{expand("color", "Black")}`
- Numeric-valued property:
 - `#{expand("weight")}`
 - `#{expand("weight", 0.0)}`

- `#{expand("weight", 0.0, #.00)}`

Option Constraints

Constraint tables enable you to limit a customer's choice of one or more option items based on the customer's choice of another option item. For example, the choice of an exterior color for a car might limit the choice of interior colors.

Working With Constraints

To Create a Constraint Table

You create an option constraint by creating a constraint table. You define constraint tables at the model level.

1. Navigate to the model where you want to create the constraint table.

See "To Access the Visual Modeler" on page 446 for information about how to navigate the model group hierarchy.

2. Click the **Tables** tab.

This displays two tabs: **General Info** and **Records**. The **General Info** tab displays general information about the table displayed in the Table Name field.

3. Click **New...**

This displays the **Create New Constraint Table** tab.

FIGURE 92. Create New Table Tab

4. Enter a Table Name, a Description, and a date range (Start Date/End Date) for the table. (You can click the **Calendar** icon to select dates from the calendar.)
5. Enter a message.

This message appears when the end-user chooses a selection which is incompatible with a constraint defined in the table.

- a. Select the message type: error, warning, or suggestion.
 - b. Enter the message in the Message field.
6. Click **Save Changes**.

This re-displays the **Tables** tab with the new table in the Table Name field. The next step is to create the option constraints that are a part of the table. You do this in the **Records** tab. See "To Define Option Constraints" on page 547.

To Modify a Constraint Table

1. Navigate to the model that contains the table you want to modify.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.
2. Click the **Tables** tab.
3. Select the table from the Table Names drop-down list.

4. Modify the table. You can:
 - Define option constraints (see "To Define Option Constraints" on page 547).
 - Modify option constraints (see "To Modify an Option Constraint" on page 551).
 - Delete option constraints (see "To Delete Option Constraints" on page 552).
 - Modify the name, description, or effectivity dates in the **General Info** tab.
 - Modify the error/warning/suggestion message in the **General Info** tab.

To Define Option Constraints

After you create a table and the option classes that will provide the constraints, you define the constraints. Each row in the table represents a constraint.

1. Navigate to the model that contains the table for which you want to define the constraint.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.
2. Click the **Tables** tab.

This displays two tabs: **General Info** and **Records**. The **General Info** tab displays general information about the table displayed in the Table Name field. The **Records** tab is where you will define the constraints.
3. Select the table from the Table Names drop-down list.
4. Click the **Records** tab.

This displays the currently defined option constraints.



FIGURE 93. Records Tab

5. Add columns to the constraint table.
 - a. Select an option class from the Table Column name drop-down list.

The drop-down list includes all the option classes belonging to the model including any option classes nested within option classes as well as option classes that are part of option class groups attached to the model. The drop-down list will display the path to the option class relative to the model.

For example, the following figure shows two selections in the drop-down list called **Monitor** and **Software**. Notice that the Navigation frame shows two option classes by these names directly below the model.

The drop-down list has another selection, **Software.Application**. Notice that the model has an option class called **Software** directly below the model, with a nested option class called **Application**. Notice how the drop-down list indicates the path relative to the model, **Software.Application**.

The drop-down list also includes a selection, **MX-7500 Service.Warranty**. This corresponds to the option class group, **MX-7500 Service**, directly below the model. **Warranty** is an option class within the group.

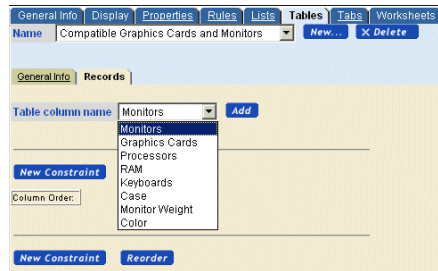


FIGURE 94. Records Tab with Drop-Down List

- b. Click **Add**.
The column name is added to the table.
- c. Repeat the last two steps for every column you want to add.

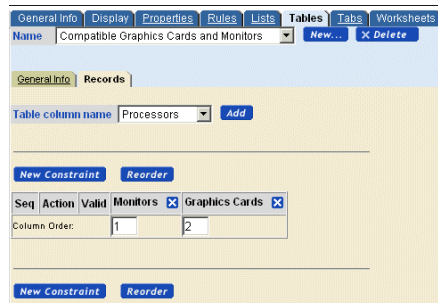


FIGURE 95. Records Tab with Columns

6. Define an option constraint.
 - a. Click **New Constraint** to add a new row to the table.

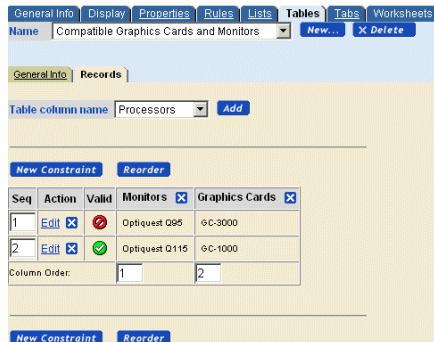


FIGURE 96. Constraint Table with New Constraint Added

- b. Click **Edit**.

This displays the option classes as table columns, along with their option items.

The option items that display include any option items belonging to an any option item group attached to the option class.

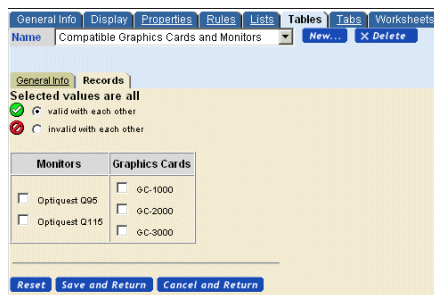


FIGURE 97. Defining Constraints

- c. Define compatibility ("Selected Values are all"). That is, will the selections you make in one column be valid or invalid with the selections in the other column(s)?
- d. Select one or more option items in each column.
- e. Click **Save**.

A new row appears in the table.

- Repeat the last step for each constraint you want to define.

To Modify an Option Constraint

- Navigate to the model that contains the table with the constraint you want to modify.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

- Select the table from the Table Names drop-down list.
- Click the **Records** tab.

This displays the currently defined option constraints.

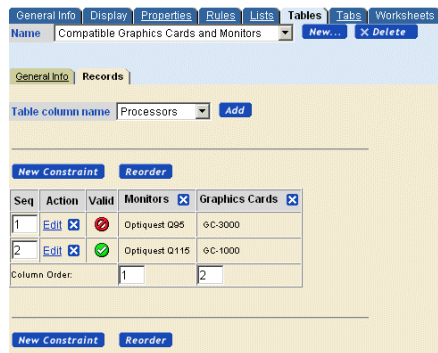


FIGURE 98. Records Tab

- Find the constraint row you want to modify and click **Edit**.

This displays the constraint information.

- Modify the constraint information.
 - Modify compatibility.

Will the selections you make in one column be valid/invalid with the selections in the other column(s)?

- Modify the option items in each column.
- Click **Save**.

The row is changed based on your modifications.

To Delete Option Constraints

1. Navigate to the model that contains the table with the constraint you want to delete.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. Click the **Tables** tab.
3. Select the table from the Table Names drop-down list.
4. Click the **Records** tab.

This displays the currently defined option constraints.

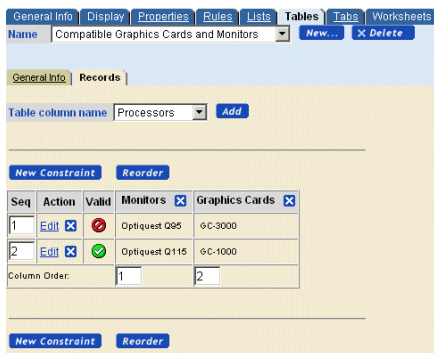


FIGURE 99. Records Tab

5. Find the constraint row you want to delete.
6. Click **Delete (X)**.

The constraint row is deleted.

To Delete a Constraint Table

1. Navigate to the model that contains the table with the constraint you want to delete.

See "To Access the Visual Modeler" on page 446 for information about how to navigate the model group hierarchy.

2. Click the **Tables** tab.
3. Select the table from the Table Names drop-down list.

4. Click the **Delete** button.

The constraint table is deleted.

Importing and Exporting Models

Importing Model Groups and Models

You can import model groups and models in the form of XML files. You can either import the entity relative to its original root model group, or you can designate a location into which to import. The model will appear in the Navigation frames, enabling you as modeler to add to or modify the imported model.

To Import Model Groups and Models

1. Access the Visual Model page.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. If you want to import to a selected point, then navigate to the model group within which you want to import the file.
3. Click **Import** in the toolbar.

This displays the Visual Modeler Import window (see Figure 100 on page 554).

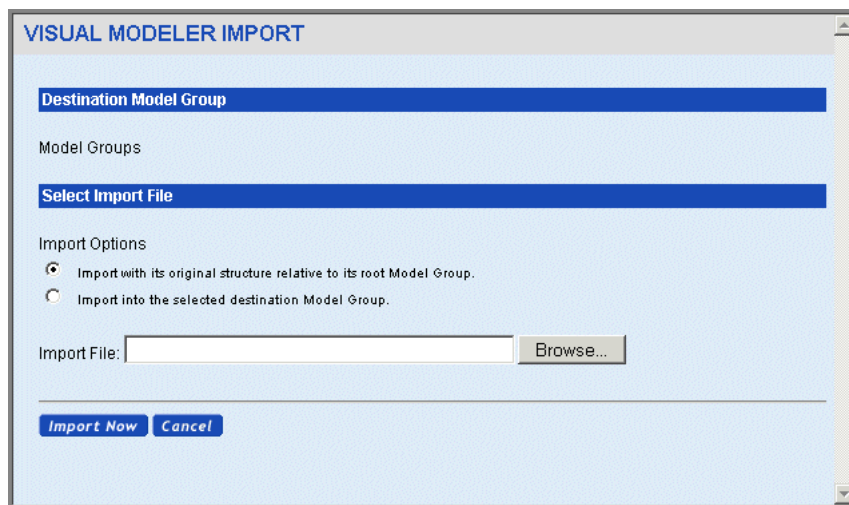


FIGURE 100. Import Window

4. Click **Browse...** to find the XML file you want to import.

When you select the file, the file will be displayed in the field along with the complete path to the file.

5. Select the import option.
 - **Import with its original structure relative to its root model group**
When you make this selection, the Import process will ignore any Destination Model Group indicated at the top of the window.
 - **Import into the selected destination model group**
6. Click **Import Now**.

The imported model group or model and its structure will be imported based on the import option you selected.

Exporting Model Groups and Models

You can export any model group or model as an XML file to a specified location on your machine.

To Export a Model Group or Model

1. Navigate to the model group or the model that you want to export.

See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.

2. Click **Export** in the toolbar.

You can either open the XML file at its current location, choosing a desired text processing tool, or you can save the file to a desired location.

Using Dynamic Instantiation

Dynamic instantiation provides a way to allow users to configure products on the fly while avoiding the need to create option items for each possible product configuration in your model. For example, consider a server rack. The user can decide on the number of slots they need and create dynamic instantiation controls for each type of component, such as servers and storage arrays that can fit into a slot, AC or DC power, and so on. As the modeler, you create the rack model, then create option classes for each of the rack's configurable features (such as servers and storage arrays) and set them as dynamic instantiation control classes. An end-user buying computer racks navigates to the rack product on your site and clicks the Configure button next to the servers and storage array choices. This causes a new option item to be added to the model for that configurable feature. The user can then configure each option item by clicking the Configure button that appears next to each added item. When the entire rack and all the configurable features have been added and configured, the user clicks the Add button located in the button bar at the top of the Configurator page to add the rack to their cart.

The following steps describe the process in more detail.

1. On the Model Group Navigation page, click New Model.

The New Model page displays.

2. Enter a name for the model, then click Save and Edit.

The Model Navigation page for the new model displays.

3. Click New Option Class.

The New Option Class page displays.

4. Enter a name for the new option class, then click Save and Edit.

5. Click the Display tab, then choose Dynamic Instantiation from the UI Control drop-down list, as shown in the following figure.
Set other Display properties as appropriate, then click Save All Changes.
6. Click the Properties tab, then set the following properties from the Unattached Properties drop-down list:
 - a. CONFIG: SUBMODEL NAME
Enter the name of an existing submodel for the property value, then click Attach.
 - b. CONFIG: SUBMODEL RETURN
Enter the name of an existing submodel to which the end-user should return after clicking the Add button, then click Attach.
7. Click Save All Changes.
8. Return to the new model's root node, then click the Compile and Test icon to test your dynamic instantiation option class.

Searching

You can search for entities that contain properties and property values that you specify as parameters. You can search across the entire hierarchy, or you can limit your search to model groups, models, option classes, option items, and rules, or you can limit your search even further to the currently selected model or group.

To Search for Entities

1. Access the Visual Modeler page.
See "To Access the Visual Modeler" on page 446 for information on how to navigate the model group hierarchy.
2. If you want to search within a specific model or group, then navigate to and select the model or group.
3. Click **Search** in the toolbar.
This displays the Search window (Figure 101 on page 557).

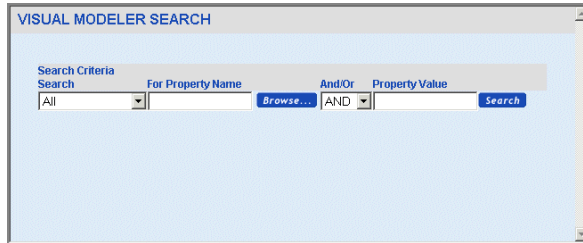


FIGURE 101. Search Window

4. Select the scope for the search from the Search drop-down list.

You can search all entities, you can limit the search to model groups, models, option classes, option items, or rules. If you are searching within a specific model or group (see Step 2), then you can limit your search to **Current Model** or **Current Group**.

5. Enter either a property name or a property value or both.

Click **Browse...** to display a browser window to select a property from a list of all the properties in the Visual Modeler.

Use the drop-down list to select AND or OR. Select AND to produce search results that include both the Property Name and Property Value parameters you select. Select OR to produce search results that include either parameter.

When you enter a property value, the search results will include property values that contain the property value you enter. For example, if you enter "75", then the search results will include any properties with the value "75" as well as property values such as "7550-1" or "MX-75-1".

6. Click **Search**.

The search results will display below the parameters. By default, the result is sorted in ascending order by property name. You can click on one of the following columns to sort:

- Property Name
- Value
- Location

When you click the column title the first time, the column is sorted in ascending order.

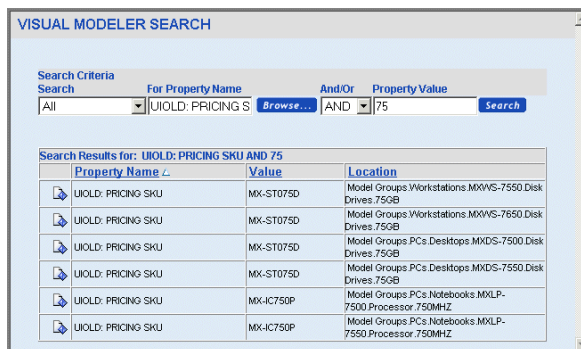


FIGURE 102. Search Window with Results

Reporting

You can run a report on a model that you specify. You can select the types of information you want in the report:

- Rule definitions
- List definitions
- Property definitions
- Display Settings
- Attached Properties
- Attached Rules
- Expand Groups

To Run A Report

1. Access the Visual Modeler page.
See "To Access the Visual Modeler" on page 446.
2. Click **Report** in the toolbar.
This displays the Report Entry window.

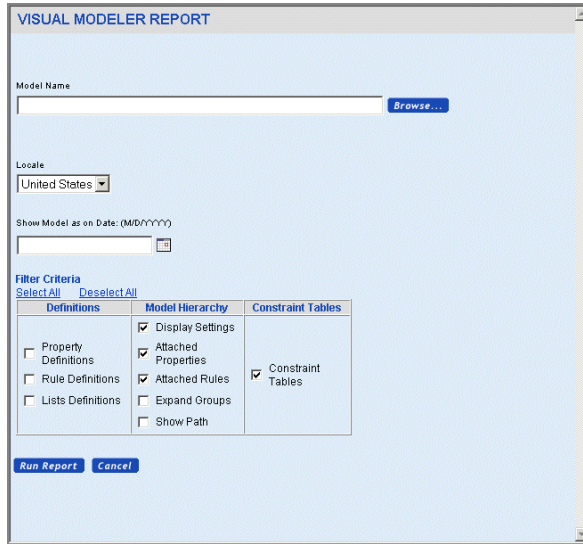


FIGURE 103. Report Entry Window

3. Enter the model you want to report on.

You can click **Browse...** to find and select the model in the model hierarchy.

4. Select the locale in which you want to run the report.
5. Select a date to report.

This produces a report for the models for whom the selected date falls within the range of their effectivity dates. The report does not display any models (or entities within the model) for whom the selected date falls outside their effectivity dates.

6. Select the information you want to include in the report.
7. Click **Run Report**.

A report is displayed based on the parameters you entered (Figure 104 on page 560).

VISUAL MODELER REPORT

[Back](#)

Model Report: MXDS-7550 Date: 6/23/2002

Model Hierarchy: MXDS-7550

▼ **MXDS-7550**

Description: Matrix 7550 Desktop
 Effectivity Dates: 9/12/2001 to 9/20/2002
 Product ID: MXDS-7550

Display Parameters

Name	Value
LI_ICON_GRAPHIC	..Images/Desktop.gif
LI_JSP_FILENAME	Configurator_Tabbed.jsp

Attached Properties

Name	Value
LITENPrum_cols	2
MX75_Mem_Ordered	0
MX75_Mem_Required	0
MX75_Card_Slot_Available	4
MX75_Bays_Available	2

Attached Rules

Name	Start Date	End Date
MSG_E_Available_Card_Slots_Check	10/8/2001	10/8/2101
EXP_MX75_Automatic_Memory_Selection	10/8/2001	10/8/2101
MSG_E_Available_HDD_Slots	10/8/2001	10/8/2101
MSG_E_MX75_Memory_Software_Check	10/8/2001	10/8/2101

▼ **Microprocessor**

Description: CPU
 Effectivity Dates: 9/12/2001 to 9/20/2002
 Product ID: Not Assigned

Display Parameters

Name	Value
LI_IGNORE_IN_QUOTE	yes

FIGURE 104. Report Results Window

This chapter describes the user interface (UI) controls and how they can be used to help your customers configure your products. It covers:

- "UI Properties" on page 561
 - "Visual Modeler UI Properties" on page 562
 - "Working with Display Properties" on page 562
- "Display Properties" on page 571
- "Tabular Display of Properties" on page 576
- "Image Properties" on page 579
- "User-Entered Values" on page 580

The basic concepts and tasks of modeling are covered in CHAPTER 36, "Using the Visual Modeler".

UI Properties

A property is an attribute of a model, option class, or option item. UI properties are used to determine the look-and-feel of a product as it is configured. You can use UI properties to control how option classes are displayed, how to display properties of option items, as well as basic guiding text and pictures.

The Visual Modeler provides a set of built-in UI properties which are understood by the Sterling Configurator engine. These control the behavior of the engine and the presentation of the model to the end-user. These properties are summarized in "To Define Display Property Values" on page 571.

Working with Display Properties

The Visual Modeler provides certain *display properties* that come pre-defined with the Visual Modeler. These display properties enhance the customer experience by enabling you to provide values that define various aspects of the model or its elements. They can all be specified using the Display tab of a model, option class, or option item, or as UI properties in the Properties tab. For example, you can define a "Pre-Pick Guiding Text" for an option class either by defining it on the Display tab or by specifying the value of the UI: PRE-PICK GUIDING TEXT property on the Properties tab.

Display properties also allow you to create fields and options that end-users may use to enter their own values rather than values specified by you. See "User-Entered Values" on page 580. Note that every property displayed on the Display tab corresponds to a UI property. This means that display properties can also be set using the Properties tab provided that you know which UI property matches the display property. See "Display Properties" on page 571 for more details.

Visual Modeler UI Properties

The following table summarizes the UI properties that are built in to the Visual Modeler.

TABLE 29. Visual Modeler UI Properties

Property	Type	Comments
UI: ADDITIONAL DESCRIPTION	string	You can use this property to add additional descriptive text to an option class. use this property in conjunction with the UI: DISPLAY RESULTS property.
UI: ALIGNMENT	string	"Horizontal" or "Vertical" controls layout of radio buttons and check box controls.

TABLE 29. Visual Modeler UI Properties (Continued)

Property	Type	Comments
UI: AUTOMATIC POST	string	<p>"yes" or "true" turns on automatic posting for an option class.</p> <p>After a customer makes a pick of an option item, then you usually want the server to re-display the page so that rules can be fired and any changes to the available option classes displayed. However, if you do not want picks in an option class to cause a re-display, then set this property to "no" or "false". This is equivalent to selecting On User Request from the Submit to Server Display property drop-down list.</p> <p>The option class is displayed with Update button: after making a pick in this option class, a user can click the Update button to request a re-display of the page from the server.</p>
UI: CLASS DISPLAY NAME	string	<p>Use this property at the model level to determine what is displayed as the displayed name of option classes. By default, this property takes the value <code>\${expand("_description")}</code> which means that the value of the option class's Description field is displayed.</p> <p>For example, if you want to display option class names instead of descriptions, then set this property to <code>\${expand("_name")}</code>. You can overwrite this value at a single class by using the UI: DISPLAY NAME property.</p>
UI: COLUMN ALIGNMENT	string	<p>Used in the tabular display control to specify the alignment of the values in the column. The tabular display control uses the ";" character to separate entries from each other, so the format of this column is something like: "left;left:center:right".</p>
UI: COLUMN HEADINGS	string	<p>Used in the tabular display control to specify the titles of columns. Each title is separated from each other with the ";" character. For example: "Speed;Pins;Manufacturer".</p> <p>See "Tabular Display of Properties" on page 576 for an example of using this property.</p>
UI: COLUMN PROPERTIES	string	<p>A semi-colon-separated list of property names used in the tabular display of properties. For example: "SPEED;NOPINS;SUPPLIER", where SPEED, NOPINS, and SUPPLIER are properties defined on option items in an option class.</p> <p>See "Tabular Display of Properties" on page 576 for an example of using this property.</p>

TABLE 29. Visual Modeler UI Properties (Continued)

Property	Type	Comments
UI: COLUMN SPAN	numeric	Controls how many columns an option class requires for its display in the customer-facing display of the model. This is the same as entering a number for the Number of Columns field on the Display tab. See also UI: SKIP COLUMNS.
UI: CONFIG CELL HTML CLASS	string	Sets the CSS class attribute in the HTML. Use this property to control the look-and-feel of cells. Note that the Visual Modeler uses the internal.css CSS file when you test models.
UI: CONSTANT GUIDING TEXT	string	Defines the guiding text that will always be shown for an option class. This is the same as entering text for the Constant Guiding Text field on the Display tab. See also UI: POST PICK GUIDING TEXT and UI: PRE PICK GUIDING TEXT.
UI: CONTROL	string	The name of the JSP fragment used to render an option class. Do not use UI: JSP FILENAME at the option class level.
UI: DEFAULT SELECTION	string	"true" or "yes" on an item makes the item a default selection within its parent option class.
UI: DISPLAY ADDITIONAL INFO	string	Use this property to provide a description specific to a particular instance of a sub-model. If you attach this property to the root node of a submodel and pass it as an output property to the parent model, the parent model displays the description next to the item in the parent model. This allows you to give feedback to the end-user about how the sub-model is configured. This is particularly useful for dynamic instantiation, where there can be multiple instances of a sub-model, each configured differently, and you want to provide an appropriate description for each instance of the submodel.
UI: DISPLAY NAME	string	Use this property to determine what is displayed as the displayed name of the option class. By default, this property takes the value <code>\${expand("_description")}</code> which means that the value of the option class's Description field is displayed.

TABLE 29. Visual Modeler UI Properties (Continued)

Property	Type	Comments
UI: DISPLAY RESULTS	string	<p>This property is deprecated. A property that is displayed along with the description of items. This special property also allows the usage of text expansion macros. Currently we support:</p> <pre> <code> $\\${expand(propname[,defaultValue[,picture-String]])}$</code> </pre> <p>but the name of this "function", expand in this case, is accessed via the object manager.^a</p> <p>An example usage is to set a description string in the UI: ADDITIONAL DESCRIPTION property, and then set the value of this property to <code>$\\${expand("UI: ADDITIONAL DESCRIPTION")}$</code>.</p>
UI: HELP URL	string	<p>A URL that is used to turn an option class description into a hyperlink, typically used to provide additional information about what that option class is for, but could also be a datasheet or any other hyperlink. Clicking on the hyperlink will bring up the page in a new window. This is the same as entering text for the Help URL field on the Display tab.</p>
UI: ICON GRAPHIC	string	<p>Used with an option class to display a picture along with the description of the option class. This is the same as entering text for the Image field on the Display tab: see "Image Properties" on page 579 for information on how values in this field are resolved to URLs.</p>
UI: IGNORE IN QUOTE	string	<p>When set to "yes" or "true" will cause whatever item this property is attached to, to be filtered out of the summary page, and flagged as not visible in the BOM transfer to the shopping cart. This is the same as checking Ignore in Quote on the Display tab.</p> <p>Typically, this field is used to ensure that only selected option items are displayed in shopping carts and to suppress option classes in the list of items in a shopping cart.</p>
UI: JSP FILENAME	string	<p>The name of the JSP page that will render the model: Configurator_Tabbed.jsp or configurator.jsp. This property is added to support easier customization and eventually to allow different presentations per model. Using built-in customization elements of Sterling Configurator, it is possible to dynamically change pages as well.</p>
UI: LEAD TIME	numeric	<p>Attached to items in the model. It is used to build a maximum lead time for the entire model by finding the largest lead time of all items currently selected.</p>

TABLE 29. Visual Modeler UI Properties (Continued)

Property	Type	Comments
UI: NUMBER OF COLUMNS	numeric	<p>Number of columns to divide the end-user configurator presentation. This property is defined at the model level to manage how many columns are used to display the option classes for a model.</p> <p>This property in conjunction with UI: COLUMN SPAN, UI: ROW SPAN, and UI: SKIP COLUMNS controls how option classes are arranged on the page. This property is the same as setting the Number of Columns property in the Display tab.</p>
UI: OPTION CLASS REQUIRED	string	"yes" or "true" causes Sterling Configurator to require that a selection be made for an option class. For radio buttons this causes the None selection to be removed.
UI: OPTION CLASS SELECT	string	<p>This property is used to specify what UI control should be used when no specific UI: CONTROL value is specified. Its use is primarily to support importing models from external configuration systems or from earlier releases of the Visual Modeler.</p> <p>It takes "single" or "multiple" as values, and is only used in the absence of a UI: CONTROL property to determine if a radio button or checkbox control should be shown for an option class.</p>
UI: OPTION CLASS TYPE	string	Obsolete: do not use.
UI: OPTION CLASS VIEW	string	<p>"POPUP", "POPUP-QTY", or "INVISIBLE". This controls the display behavior of an option class. If POPUP, a standard option class is shown; if POPUP-QTY is selected, then a quantity box will be shown for each selected item within that control. Finally, INVISIBLE is used to prevent the display of the control entirely.</p> <p>INVISIBLE is often used to hide option classes until other picks made by the customer requires the class to be displayed.</p>
UI: POPUP-QTY ALLOWED VALUES	string	<p>This controls what values are available for a selection in a popup drop-down list. Use this at the option class level, in conjunction with setting UI: OPTION CLASS VIEW to POPUP-QTY.</p> <p>A "," separated list of allowed values. Ranges can be specified with "-", so 1-4,7-9 is the same as 1,2,3,4,7,8,9. If you leave this field blank, then a text field is displayed with the current value; otherwise a drop-down list with the allowed values is displayed.</p>

TABLE 29. Visual Modeler UI Properties (Continued)

Property	Type	Comments
UI: POST PICK GUIDING TEXT	string	<p>A guiding text message displayed with an option class description if the user has made at least one pick from within the option class. This is the same as entering text for the Pre-Pick Guiding Text field on the Display tab.</p> <p>This property is not displayed until a customer makes a pick. See also UI: CONSTANT GUIDING TEXT and UI: PRE PICK GUIDING TEXT.</p>
UI: PRE PICK GUIDING TEXT	string	<p>A guiding text message displayed with an option class description if the user has not made a pick from within the option class. This is the same as entering text for the Post-Pick Guiding Text field on the Display tab.</p> <p>Once a pick has been made, then this property is no longer displayed. See also UI: CONSTANT GUIDING TEXT and UI: POST PICK GUIDING TEXT.</p>
UI: PREVENT SELECTION	string	<p>"yes" or "true" causes the Sterling Configurator to prevent the user from selecting items that would violate a constraint table rule. If the Constraint Selections display property is set to "Hide constrained items", then this property is set to "yes".</p>
UI: PRICE	numeric	<p>The price for an item that will be used if <code>STATIC_PRICING</code> or <code>OVERRIDE_PRICINC</code> is set in the business rules. In the case of <code>OVERRIDE_PRICING</code>, this value will be used if a price cannot be found for the item in the price list.</p>
UI: PRICING SKU	string	<p>The SKU to use when looking up the item in the price list. Note that if you set a product ID value for this property, then it overrides the value of the Assigned Product ID in determining prices.</p>
UI: PRICING STYLE	string	<p>Usually, you use this property at the option class level. It controls how prices of option items are displayed to the end user as follows:</p> <p>NONE: Do not display prices as user configures product.</p> <p>ABSOLUTE: Display prices next to option items as absolute prices.</p> <p>DELTA: Display prices next to option items as their effect on the price of the whole configured product.</p> <p>This property is the same as setting Pricing Style in the Display tab.</p>

TABLE 29. Visual Modeler UI Properties (Continued)

Property	Type	Comments
UI: PRODUCT ID	string	<p>If a product has been associated with a node of a model, then this property can be used to retrieve the product ID of the associated product.</p> <p>The value of this property is resolved at compile time, so if the product ID is changed, then you must re-compile the model for the change to take effect.</p>
UI: PRODUCT NAME	string	<p>If a product has been associated with a node of a model, then this property can be used to retrieve the product name of the associated product.</p> <p>The value of this property is resolved at compile time, so if the product name is changed, then you must re-compile the model for the change to take effect.</p>
UI: PRODUCT DESCRIPTION	string	<p>If a product has been associated with a node of a model, then this property can be used to retrieve the description of the associated product.</p> <p>The value of this property is resolved at compile time, so if the product description is changed, then you must re-compile the model for the change to take effect.</p>
UI: QUANTITY AVAILABLE	numeric	<p>Do not use in this release.</p> <p>Used in the quantity matrix, this can optionally be attached to the items for the matrix. If so it will set the quantity available of each item. If the control is set to show quantity available this property value will be displayed in a secondary row for each item.</p>
UI: REQUIRED	string	<p>Obsolete: do not use.</p>
UI: ROW SPAN	numeric	<p>Controls how many rows an option class requires for its display in the end-user presentation of the page. In conjunction with UI: NUMBER OF COLUMNS and UI: COLUMN SPAN, this property controls the layout of the page viewed by end-users. This is the same as entering a number for the Number of Rows field on the Display tab.</p> <p>See also UI: SKIP COLUMNS.</p>
UI: SHOW ITEM IMAGES	string	<p>"yes" or "true" controls whether item images are shown.</p>

TABLE 29. Visual Modeler UI Properties (Continued)

Property	Type	Comments
UI: SKIP COLUMNS	numeric	<p>Number of columns to skip after this class. It is used to add to the count variable that is tracking how many cells are being used to lay out the option classes. This is the same as entering a number for the Number of Columns to Skip field on the Display tab.</p> <p>If you have used the UI: COLUMN SPAN property or UI: ROW SPAN for another option class, then use this property to account for table cells in the layout that the multiple span class uses.</p>
UI: SUPPRESS NAME DISPLAY	string	"yes" or "true" causes Sterling Configurator to not display the names of option classes.
UI: SUPPRESS NONE SELECTION	string	"yes" or "true" suppresses the NONE selection value for radio buttons.
UI: SUPPRESS UEV NONE VALUE	string	<p>"yes" or "true" suppresses the NONE selection for UEV combo boxes. Use this in conjunction with UI: UEV ALLOWED VALUES property.</p> <p>For example, if you have specified that a user-entered value field can only take the values Red, Green, Blue, then if the value of this property is set to "yes", then None will not appear in the drop-down list of selectable values. If you set the value of this property to "no", or do not attach this property, then None will be a selectable value.</p>
UI: UEV ALLOWED VALUES	string	<p>Comma-separated list of values for a combobox UEV control.</p> <p>Suppose that you want to allow customers to enter only one color from a small list of colors. Then enter the list like this:</p> <p>Black,Blue,Green,Red,White</p> <p>When this property is set, then the user-entered value option item is displayed as a drop-down list of these values. None is also displayed as a selectable option, unless you set the UI: SUPPRESS UEV NONE VALUE property to "yes".</p> <p>This property is the same as setting values in the Allowed Values display property.</p>

TABLE 29. Visual Modeler UI Properties (Continued)

Property	Type	Comments
UI: UEV ASSIGNMENT PROPERTY	string	<p>The name of a property where a UEV will store its value. This property should be of the correct type to contain the UEV. Note: numeric properties can be used to hold INTEGER UEVs as well as NUMERIC UEVs.</p> <ul style="list-style-type: none"> • If the value of this property is just a property name, then the property will be set on the current item. • If the value contains a path to a property as well as the property name, then the property will be set on the item referenced by the path if it exists. <p>Once a user makes their pick in the user-entered value field, then the assigned property can be used by rules or in the display of the model, just like any other property.</p> <p>This property is the same as setting a value in the Assign Value to Property display property.</p>
UI: UEV INTEGER VALUE	integer	<p>Filled in by the engine when an integer UEV has a value in it. This provides you with a way to reference the value of the field without assigning it to another property: see UI: UEV ASSIGNMENT PROPERTY to use another property.</p>
UI: UEV LIST VALUE	list	<p>Filled in by the engine when a list UEV has a value in it (not currently used). This provides you with a way to reference the value of the field without assigning it to another property: see UI: UEV ASSIGNMENT PROPERTY to use another property.</p>
UI: UEV NUMERIC VALUE	numeric	<p>Filled in by the engine when a numeric UEV has a value in it. This provides you with a way to reference the value of the field without assigning it to another property: see UI: UEV ASSIGNMENT PROPERTY to use another property.</p>
UI: UEV POSTFIX	string	<p>A string of text displayed after the UEV entry field.</p> <p>This property is the same as setting a value in the Text After Entry Field display property.</p>
UI: UEV PREFIX	string	<p>A string of text displayed before a UEV entry field.</p> <p>This property is the same as setting a value in the Text Before Entry Field display property.</p>
UI: UEV SELECTION	varies	<p>Obsolete: do not use.</p>

TABLE 29. Visual Modeler UI Properties (Continued)

Property	Type	Comments
UI: UEV SPECIAL	string	Used by the user entered value control to enable a file list or notes control. This will be phased out and replaced by a new file attachment control and notes control in future releases: do not use.
UI: UEV STRING VALUE	string	Filled in by the engine when a string UEV has a value in it. This provides you with a way to reference the value of the field without assigning it to another property: see UI: UEV ASSIGNMENT PROPERTY to use another property.
UI: UEV TYPE	string	"string", "integer", or "numeric"; the type of UEV control.

- a. To add additional macros, define a new class that implements the IExpansionHandler interface, and put a reference to it into the object manager.

Display Properties

To Define Display Property Values

1. Navigate to and display the detail page for the model, option class, or option item.

See "To Access the Visual Modeler" on page 446 for information about how to navigate the model group hierarchy.

2. Click the **Display** tab.

This displays the display properties appropriate to the level.

3. Edit the desired fields.

See Table 30 on page 572 for an alphabetical list of the properties, where they can be assigned, and what they mean. Because each display property corresponds to a UI property, the table also provides the name of the corresponding UI property, and further information about each UI property is provided in "Visual Modeler UI Properties" on page 562.

4. Click **Save All Changes**.

TABLE 30. Display Properties

Field Name/Property Name	Where Used	Description
Automatic Post/UI: AUTOMATIC POST	Model Option Class	Depending on the value you choose, this property specifies how posting is done: none: No update is performed when the customer selects an option item. update: An incremental update occurs when the customer selects an option item. final (default): A final update occurs when the customer selects an option item.
Constant Guiding Text/UI: CONSTANT GUIDING TEXT	Model Option Class	Used to add extra text to the displayed HTML page. This text is "constant", that is, it appears all the time, even after a selection is made. For example, guiding text for a configurable camcorder may state "Only lithium batteries type XYZ are compatible with this model."
Control/UI: CONTROL	Option Class	Enables you to determine how the option items are displayed: Radio button: Items appear as radio buttons. Customer can only select one. Checkbox: Option items will appear with check boxes; multiple selection allowed. Drop down list: Items appear in a drop-down list. Combobox: Items appear in drop-down list, but end-users can also type in a selection. Multiple Selection listbox: Items appear in a scrollable list from which the customer can make multiple selections. Display All Children: When you have nested option classes, nested classes appear with their option items visible (as opposed to option items only appearing when nested option classes are "picked"). User EnteredValue: Items appear as user-entered fields. Tabular Display: Items appear as rows in a table.

TABLE 30. Display Properties (Continued)

Field Name/Property Name	Where Used	Description
Default Selection/UI: DEFAULT SELECTION	Model Option Class Option Item	This property specifies that, if the user does not choose an entity in the option class, this entity (embedded model, nested option class, or option item) is automatically selected. You can use this property in conjunction with the Option Class Required special property. You can only assign it to one option item in an option class.
Display Template	Model	Select the type of user interface from the drop-down list: Tabbed UI or Non-tabbed UI. See "Working with a Tabbed User Interface" on page 488.
Help URL/UI: HELP URL	Model Option Class	Enables you to display a link (URL) to a page that has additional information about the model, option class, or option item.
Icon Graphic/UI: ICON GRAPHIC	Model Option Class	Provides the location (fully qualified path) of a GIF format file to be displayed next to this model, option class, or option item.
Ignore In Quote/UI: IGNORE IN QUOTE	Model Option Class Option Item	This special property is attached to option classes and option items that will not be transferred into the summary page when these option items are selected by the customer or through an expansion rule.
Lead Time/ UI: LEAD TIME	Model Option Class Option Item	Enables you to specify a lead time between when a customer orders a product that includes this item and when that product can be expected to ship.
Option Class Required/UI: OPTION CLASS REQUIRED	Option Class	Enables you to specify whether or not a customer must make a selection in that option class to complete the configuration. Customer must select one of the option items to complete the configuration.

TABLE 30. Display Properties (Continued)

Field Name/Property Name	Where Used	Description
Option Class View/UI: OPTION CLASS VIEW	Option Class	<p>Determines (1) if the items in this option class are displayed, and (2) if the pop-up quantity is displayed next to the option item.</p> <p>Popup: When the customer clicks the drop-down arrow, the line is expanded to display all items.</p> <p>Popup-qty: Customer types in a number in the quantity field. The entered value influences the quantity of option items that are ordered for this option class.</p> <p>Invisible: Option class and its items are not displayed to the customer. This is typically used for an automatic expansion when the customer does not need to know about the added option items that are part of the configuration. For example, if a customer orders a special wheel, then invisible option items may include nuts and bolts that are included with the special wheel.</p>
Popup-Qty Values/ UI: POPUP-QTY ALLOWED VALUES	Option Class	<p>Enables you to set quantity for that item. The quantities specified appear as possible selections in a quantity box next to the item.</p>
Post-Pick Guiding Text/UI: POST PICK GUIDING TEXT	Model Option Class	<p>Used to add extra text displayed on the HTML page after the customer has made a selection. For example, a model of a computer has an option class called "Operating System", and that option class has an option item called "Windows 2000". Post-pick guiding text for that option item might be "Windows 2000 requires a minimum of 256MB of RAM; make sure the amount of RAM you select is at least 256MB".</p>
Pre-Pick Guiding Text/UI: PRE PICK GUIDING TEXT	Model Option Class	<p>This special property is assigned at the option class level, and is used to add extra text that is displayed on the HTML page. The text disappears once a selection is made. For example, pre-pick guiding text for a CPU option class of a configurable computer may state "Choose a Processor". Once a processor is chosen, the text disappears.</p>
Prevent Selection of Items Resulting in Constraint Errors/ UI: PREVENT SELECTION	Option Class	<p>Enables you to prevent a customer from selecting items in this class that are incompatible with items in another class (based on an option constraint table).</p> <p>If the Constraint Selections display property is set to "Hide constrained items", then this property is set to "yes".</p> <p>If Option Class Required is selected, then you cannot check this box.</p>

TABLE 30. Display Properties (Continued)

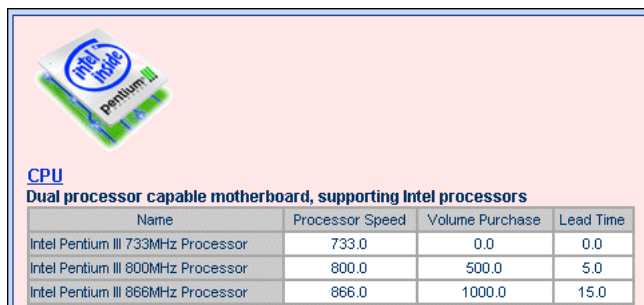
Field Name/Property Name	Where Used	Description
Pricing Style/UI: PRICING STYLE	Model Option Class	<p>This special property enables you to specify how option class items will display price information. There are three possible values:</p> <p>none: If you assign this property with the value <i>none</i>, then the option class' items are displayed without any pricing information.</p> <p>delta: If you assign this property with the value delta, then the option items display pricing information in relation to the total base price of the configurable product.</p> <p>When end-users first see the option class, they see the option items with prices as "Add \$xxx.xx", meaning "selecting this item adds this amount to the current configuration price of the model." Once the end-user selects an option item, the other option items will show either "Add \$xxx.xx" or "Subtract \$xxx.xx", depending on how choosing those option items will affect the price.</p> <p>absolute: If you assign this property with the value <i>absolute</i>, then the option items display pricing information as the total cost of that item. This kind of pricing information is not relative to any base price. It is simply the cost of that item.</p> <p>Note: Only the absolute property set at Option Class level is considered while displaying the price information.</p>
Price/UI: PRICE	Model Option Class Option Item	<p>This special property enables you to assign a specific price to the item. This property is used to attach a price to a model if your model, option class, or option item is not associated with a product ID (see "To Associate a Product with a Model, Option Class, or Option Item" on page 456).</p> <p>Note that prices assigned to option items in this way are not preserved when the configured product is returned to a cart.</p>
Return From Submodel/ CONFIG: SUBMODEL RETURN	Option Item	<p>Setting this property to "no" allows end users to transition from one model to the next. If a user returns to a model, all selections and derived properties are reset.</p>
User Entered Value Type/UI: UEV TYPE	Option Item	<p>This property is displayed only if you selected "User entered value" for the Control display property of the option class to which the item belongs. This property enables you to define the type: string, integer, or numeric.</p>

TABLE 30. Display Properties (Continued)

Field Name/Property Name	Where Used	Description
User Entered Value Prefix/UI: UEV PREFIX	Option Item	This property is displayed only if you selected "User entered value" for the Control display property of the option class to which the item belongs. This enables you to provide a text string that precedes a user-entered value (For example, "\$").
User Entered Value Postfix/UI: UEV POSTFIX	Option Item	This property is displayed only if you selected "User entered value" for the Control display property of the option class to which the item belongs. This enables you to provide a text string that follows any user-entered values (for example, "inches", "feet", and so on).
User Entered Value Allowed Values/UI: UEV ALLOWED VALUES	Option Item	This property is displayed only if you selected "User entered value" for the Control display property of the option class to which the item belongs. This property enables you to define a comma-delimited list of values for numbers (1-3, 5, 9, 10-12, and so on). For strings, you can enter the name of a list property.
Validate Submodel/CONFIG: VALIDATE SUBMODEL	Option Item	This setting ensures that the submodel is correctly configured in nested configuration scenarios. Use Validate Submodel in conjunction with the Submodel Return property. The default behavior is not to validate a submodel configuration after returning to a parent model. When you set this property to "yes" and the Return From Submodel property to "yes", the submodel configuration will be validated after the user returns to the parent model and is configuring the parent or sibling. Consider using this setting carefully as there can be performance issues.

Tabular Display of Properties

To help users choose between two or more option items in an option class, it is often helpful to display one or more properties for each option item in the form of a table. For example:



CPU
Dual processor capable motherboard, supporting Intel processors

Name	Processor Speed	Volume Purchase	Lead Time
Intel Pentium III 733MHz Processor	733.0	0.0	0.0
Intel Pentium III 800MHz Processor	800.0	500.0	5.0
Intel Pentium III 866MHz Processor	866.0	1000.0	15.0

FIGURE 105. Example Tabular Display of an Option Class

You cannot use the tabular display for pickable option items. Use tabular displays with another option class that allows users to make a selection.

To Display Properties in a Tabular Form

1. Navigate to the option class whose option items you want to display in a tabular form.
2. Either:
 - a. Click the **Display** tab.
 - b. Select Tabular Display from the Control drop-down list.
 - c. Click **Save All Changes**.Or:
 - a. Click the **Properties** tab.
 - b. Select UI: CONTROL from the Unattached Properties drop-down list and enter "controls/displayProps.jsp" as its value.
 - c. Click **Attach**.
3. Select UI: COLUMN HEADINGS from the Unattached Properties drop-down list and enter a semi-colon delimited list of headings as its value.
For example, "Size;Weight;Color".
4. Click **Attach**.

5. Select UI: COLUMN PROPERTIES from the Unattached Properties drop-down list and enter a semi-colon delimited list of the property names as its value.

For example, "Monitor Size;Monitor Weight;Monitor Color".

You can use property values as described in "Properties as Variables" on page 504 to help you display the values of properties exactly as you need.

To Display Properties in a Tabular Form

1. Navigate to the option class whose option items you want to display in a tabular form.

2. Either:

- a. Click the **Display** tab.
- b. Select Tabular Display from the Control drop-down list.
- c. Click **Save All Changes**.

Or:

- a. Click the **Properties** tab.
- b. Select UI: CONTROL from the Unattached Properties drop-down list and enter "controls/displayProps.jsp" as its value.
- c. Click **Attach**.

3. Select UI: COLUMN HEADINGS from the Unattached Properties drop-down list and enter a semi-colon delimited list of headings as its value.

For example, "Size;Weight;Color".

4. Click **Attach**.

5. Select UI: COLUMN PROPERTIES from the Unattached Properties drop-down list and enter a semi-colon delimited list of the property names as its value.

For example, "Monitor Size;Monitor Weight;Monitor Color".

You can use property values as described in "Properties as Variables" on page 504 to help you display the values of properties exactly as you need.

6. Click **Attach**.

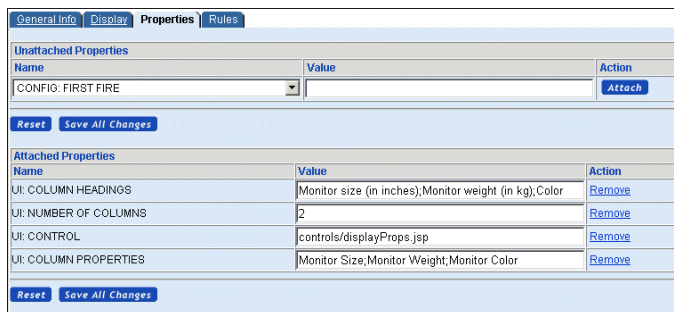


FIGURE 106. Defining Tabular Display Properties

Note that the number of columns in the table is inferred from the number of properties you define in the UI: COLUMN PROPERTIES property.

7. Click **Save All Changes**.
8. If you now click **Test**, then you can verify that the option class is now presented as a table with one row for each option item and one column for each property specified.

Image Properties

You can associate images with models, option classes, and option items as described in this section.

Models and Option Classes

Use the Icon Graphic field on the **Display** tab for models and option classes. This corresponds to the UI: ICON GRAPHIC property.

Option Items

You can attach images to option items and display them to end-users using the UI: ITEM IMAGE NAME property to specify an image for each option item. You must set the UI: SHOW ITEM IMAGES property to be “true” at the option class level.

The value of the UI: ITEM IMAGE NAME can be interpreted as a relative URL or as an absolute URL:

- If you enter "2of4stars.gif" or "../images/2of4stars.gif", then the image will be displayed by resolving the image location to:

`http://server:port/Sterling/en/US/images/2of4stars.gif`

- You can use absolute URLs to point to different locations anywhere on the Web. This is particularly useful if you use a different Web server to serve up static content for your Web site. For example:

`http://imageserver:port/configurator/images/2of4stars.gif`

User-Entered Values

You can allow your customers to type in values for a configurable product's options. For example, you may want to let customers enter a color that is not one of the pre-defined colors in a model, or you may want to let them enter a product ID for a product that is not in your product catalog, but which you can fulfil by special order.

The User Entered Value properties, described in Table 30 on page 572, enable customers to type in values. For example, suppose that you have a configurable product and you want to let the user specify their own choice of color. Do the following:

1. Navigate to the model and click **Edit**.
2. Click **New Option Class**.
3. In the Name field, enter "Custom Color Class".
4. In the Description field, enter "Enter your preferred color".
5. Click **Save**.
6. Click the **Display** tab.
7. Set the Control display property to "User Entered Values".
8. Check the **Ignore in Quote** check box.
9. Click **Save All Changes**.
10. Click the **General Info** tab.
11. Click **New Option Item**.
12. In the Name field, enter "Custom Color Item".
13. In the Description field, enter "We will provide a color match before shipping."
14. In the Navigation panel, navigate to the Custom Color Item option item.

15. Click the **Display** tab.
16. Select String, Integer, or Numeric from the **User Entered Value Type** drop-down list.
17. Click **Save All Changes**.
18. Click **Compile**.

You can use user-entered values in rules by referring to the appropriate UEV property: UEV: NUMERIC VALUE (for Integer or Numeric values) or UEV: STRING VALUE (for String values).

UI Control Reset Behavior

Some UI controls allow the user to reset (clear) a selection and start over. A Clear button displays in the configuration UI by default to enable this reset behavior. The following table summarizes the default behavior of the Clear button in UI controls.

TABLE 31. Behavior of Clear Button in UI Controls

UI Control	Default View	Default Selection	Allowed User Action
Checkbox	Displays all values.	The model's default selections are checked. If there is no default value, nothing is checked.	Check or uncheck values. Clicking Clear checks the default value. If there is no default value, clicking Clear clears all values.
Radio Button	Displays all values.	The model's default selection is selected. If there is no default value, nothing is selected.	Check or uncheck values. Clicking Clear selects the default value. If there is no default value, clicking Clear clears all values.
List Box	Displays all values.	The model's default selection is selected. If there is no default value, nothing is selected.	Select any value in the list box. Clicking Clear selects the default value. If there is no default value, clicking Clear clears all values.
Multiple Selection List Box	Displays all values.	The model's default selections are selected. If there is no default value, nothing is selected.	Select or unselect any value. Clicking Clear selects the default value. If there is no default value, clicking Clear clears all values.
Display All Children	Displays all values.	No default selection.	User cannot take any action.

TABLE 31. (Continued) Behavior of Clear Button in UI Controls

UI Control	Default View	Default Selection	Allowed User Action
Drop-down List	Displays all the values in the drop-down.	The model's default selection is selected. If there is no default value, nothing is selected.	Select any value in the drop-down list. Clicking Clear selects the default value. If there is no default value, clicking Clear clears all values.
Dynamic Instantiation	Nothing displays.	No default selection.	User cannot take any action.
Tabular Display	Nothing displays.	No default selection.	User cannot take any action.
Single-Select Tabular Display	Displays all values and a Reset button.	The model's default selection is selected. If there is no default value, nothing is selected.	Select any value. Clicking Clear selects the default value. If there is no default value, clicking Clear clears all values. Clicking Reset clears all values.
Multi-Select Tabular Display	Displays all values.	The model's default selection is selected. If there is no default value, nothing is selected.	Select or unselect value(s). Clicking Clear selects the default value. If there is no default value, clicking Clear clears all values.
Tabular Display with Quantity Box Selection	Displays all values with quantity boxes.	The model's default selection is selected. If there is no default value, nothing is selected.	The user cannot take any action.
User Entered Values	Displays all with text boxes.	No default selection.	Enter values. Clicking Clear clears all values.

This chapter covers the tasks associated with enterprise system administration for the Visual Modeler. Enterprise employees are responsible for maintaining their enterprise installation. See "Configuring the Visual Modeler" on page 415 for an overview of enterprise system administration.

Note that some site administration tasks are performed by site system administrators: see CHAPTER 42, "Site System Administration" for more information.

System Administration Tasks

You perform the System Administration tasks through the System Administration link on the Visual Modeler Administration page. This link is accessible only to authorized personnel.

You can modify system configuration settings only if you have the appropriate access function. In the reference implementation provided with the Visual Modeler, only users with the Program Management function (defined in the **Entitlements.xml** configuration file as EnterpriseProgramManagement) may access the System Administration pages.

To Modify System Settings

1. Click **System Services** in the System Administration panel on the Visual Modeler home page.

The system configuration properties are organized into logically-related groups.

2. Access each group by clicking the corresponding link on the System Administration page.

Each link takes you to a new page that displays the current values for each property.

3. Make the appropriate changes as necessary.

See "Configuration Properties" on page 584 for a description of each set of properties.

4. Click **Save All and return to List**.

5. By default, changes to the value of a system property take effect immediately, and are persisted to the file system. A server restart is not necessary, but if you do restart the server, the new value of the property remains in effect.

Configuration Properties

Use the steps described in "To Modify System Settings" on page 584 to access the property you want to modify. With the exception of the following, the properties each contain a detailed description within the user interface.

Locale Settings

The locale names supported by your installation combine the ISO-639 language codes and ISO-3166 country codes. You can define display names that will appear for these locale names in the Visual Modeler. You can define a display name for each supported locale, that is, how each locale name will appear for each supported locale. For example, you can decide that, in the en_us locale, "en_us" will be displayed as "United States", while in the de_de locale (Germany), "en_us" will be displayed as "Vereinigte Staaten".

Note:	If a display name is not defined for a locale name for the locale effective during a session, then the fields in which that locale name should appear will be blank.
--------------	--

Changes that you make to locale names become active when you restart the Visual Modeler.

To Define the Display Names

1. Select an effective locale from the Effective Locale drop-down list.
The locale names for the supported locales appear in the Locale Name column. The current display name, if any, for each locale appears in the next field in the language of the effective locale.
2. In the text field next to each locale name, type the display name you want to appear for each locale name.
3. Repeat the last two steps for each supported locale.
4. Click **Save** to save the changes and remain at the Locale Settings page; click **Save All and return to List** to save the changes and redisplay the System Administration page.

If you click **Save**, then the Effective Locale field re-displays the default system locale, as defined in the Internationalization properties.

Repeat these steps for each locale in the Effective Locale drop-down list.

Attention: If you change the effective locale without clicking Save , then any unsaved changes to Display Names will be lost.

Job Scheduler Settings

The Visual Modeler supports the ability to schedule tasks that must be performed at regular intervals as cron jobs. See CHAPTER 41, "Job Scheduling Administration" for more information on this scheduling feature.

There are two types of cron jobs: system and application.

- System cron jobs run without session information and without an associated Visual Modeler user. Typically, they are used for low-level background tasks such as garbage collection. System cron jobs do not save their last execution time or execution status to the Knowledgebase because the same job may be run on several servers in a cluster.
- Application cron jobs are used when session information (such as a username or locale) is required to run the job or if audit information might be needed to determine how changes to data objects were made.

Application cron jobs are initiated by posting an XML message to the Visual Modeler using the message URL for cron jobs: consequently, to enable application cron jobs, you must take care to set this URL correctly.

For example, if the main URL used to access the Visual Modeler is:
`http://server:port/Sterling/en/US/enterpriseMgr/matrix`

then set the cron job message URL to:
`http://server:port/Sterling/msg/matrix`

Similarly, if the main URL used to access the Visual Modeler is:
`http://server:port/store/en/US/enterpriseMgr/anderel`

then set the cron job message URL to:
`http://server:port/store/msg/anderel`

You can choose whether or not to allow either type of cron job to run on your implementation.

<p>Attention: Some cron jobs such as the search index builder must be run as an application cron jobs. To support advanced search, you must enable application cron jobs.</p>
--

Application cron jobs are created specifying a username and password of a Visual Modeler user. You must ensure that the Password data field of the CronConfig data object is not set to store one-way encrypted values.

In a clustered installation of the Visual Modeler, if you want a job to run on all servers in the cluster, then make it a system cron job. If you want the job to be run on only one server in the cluster, then you must make it an application cron job.

Frequently Used System Administration Settings

This section describes some of the most commonly used system administration settings: it does not cover all the possible settings.

Commerce Manager

Are comergent applications rendered as part of a frameset?

Set this property to TRUE if the Visual Modeler end-user pages are displayed within a frame set. For example, suppose that you have a frame set defined as:

```
<html><frameset rows="120,*"><frame src="http://server:port/Navigation.html"><frame src="http://server:port/Sterling/en/US/adirect/matrix?cmd=OnlineOrderingPageDisplay"></frameset></html>
```

When this page is displayed, you want the Visual Modeler pages to be displayed without their built-in banner heading. By setting this system property to “True”, you suppress the display of the built-in banner heading: end-users only see the navigation links provided in your **Navigation.html** page.

Availability Data Access Method

This property controls how inventory availability is obtained and estimated delivery date is calculated.

- If you select Static: inventory availability information is obtained from the static database tables, and estimated delivery date is not calculated.
- If you select System Initiated Real-Time: inventory availability information is obtained using a real-time availability check call to the Visual Modeler (when the supplier is the current storefront) and/or static database tables (when the supplier is not the current storefront), and estimated delivery date is calculated automatically.
- If you select On Demand Real-Time: a Check Availability button is displayed on the Catalog Detail or Commerce page. Upon clicking the button, inventory availability information is obtained using a real-time availability check call to the Visual Modeler (when the supplier is the current storefront) and/or static database tables (when the supplier is not the current storefront). While estimated delivery date is calculated on the Checkout page if the user clicks the Check Availability button, it is calculated automatically when the user places the order regardless of whether the user clicks the Check Availability button or not.

<p>Note: Before you choose System Initiated Real-Time or On-Demand Real-Time, you must make sure that the Visual Modeler is integrated.</p>
--

SMTP Host Machine

Set this property to the appropriate name of the SMTP host machine which is used to send email from the Visual Modeler.

Application Settings

Allowed Decimal Places for displaying extended prices

The value of this property determines how many decimal places are used in displaying calculated extended prices to users.

Allowed Decimal Places for displaying list prices

The value of this property determines how many decimal places are used in displaying list prices to users.

Lines Per Page in List Displays

This property controls the pagination behavior of the Visual Modeler. It specifies how many items appear on each page of a paginated list.

During implementation, you can configure business rules by editing the property files provided with the Visual Modeler. After implementation, you can manage the business rules from the Business Rules Manager page. See "Configuring the Visual Modeler" on page 415 for an overview of business rules administration.

Business Rules Administration Tasks

To manage the business rules in your Visual Modeler, you must be an enterprise user that has been assigned the appropriate function: typically, this is the Program Management function.

To Manage Business Rules

1. Click **Business Rules** in the System Administration panel on the Visual Modeler home page.

The Business Rules Manager page displays.

The Business Rules link appears on the Visual Modeler home page only if you are assigned the appropriate function to perform this task.

2. Click a link to modify the desired set of business rules.

Each business rule contains help text describing the rule.

3. Click **Save**.
4. Unless otherwise directed, the changes to the value of a business rule take effect immediately, and are persisted to the file system. This means that a server restart is not necessary, but if the server is restarted, the new value of the business rule continues to be used.

This chapter covers the tasks associated with Job Scheduler. For an overview of job scheduling, see "Job Scheduling" on page 416. The latter section also includes important information about setting properties related to job scheduling.

Note: Two settings define how jobs are run on your Visual Modeler. See "Job Scheduler Settings" on page 585.

Enterprise and Storefront Cron Jobs

Some cron jobs affect the entire e-commerce site and are managed only by enterprise administrators. For example, the cron job that maintains the catalog index, the cron job that controls when segment memberships are reprocessed, and the cron job that controls when the nightly segment build process starts, are all enterprise-level cron jobs. There can be only one instance of these cron jobs on a given e-commerce site, and only the enterprise administrator can see and manage them.

It is possible for storefront admins to create these "one instance" cron jobs as well: they also have access to the cron facility. Enterprise administrators must enforce the one-instance rule and ensure that storefront administrators first check with them before creating cron jobs. The following is the list of out-of-the-box cron jobs for which only one instance can exist on an e-commerce site.

- Product Sync
- User Sync
- Nightly Segments Build
- Reprocess Segments
- Maintain Indexsets

Storefront administrators can create and manage their own storefront-specific cron jobs. As a storefront administrator, you can see only your own storefront cron jobs.

Note that in a cluster environment, application cron jobs must be initiated by only one member of the cluster. In a cluster environment, you control which server starts the cron job, although you do not control which member of the cluster runs the job. System cron jobs should generally be run on all members of the cluster.

Job Scheduling Tasks

You can perform the following tasks:

- "To Display a Scheduled Job" on page 592
- "To Create a Job" on page 593
- "To Modify a Job" on page 594
- "To Run a Cron Job Immediately" on page 594
- "To Delete a Job" on page 595
- "To View the History of a Cron Job" on page 595

To Display a Scheduled Job

1. Click **Job Scheduler** in the System Administration panel on the Visual Modeler home page.
2. Click the name of a cron job to display the details of the selected job.

To Create a Job

Attention: If you are running multiple instances of the Visual Modeler, then creating or modifying a cron job will affect any of these instances running off the same Knowledgebase instance.

1. Click **Job Scheduler** in the System Administration panel on the Visual Modeler home page.
2. Click **Create New Job**.
The Cron Job Configuration page displays.
3. Enter the information about the job.

TABLE 32. Cron Job Configuration Page

Field	Description
Job Name	The name of the cron job
Program	The java implementation class that executes the job
Command Line Arguments	<p>The command line parameters that provide information about the job.</p> <p>For example, you can specify that a cron job should time out after 300 seconds (5 minutes) by setting the RequestTimeout parameter as follows:</p> <p>RequestTimeout=300</p>
Cron Job Type	<p>The type of the cron job: a system level cron job (such as cache cleaning) or an application level cron job (such as importing/exporting)</p> <p>If you select Application, then you must enter the username and password required for access to the particular data. For example, if the application-level cron job involves product manager, then you must enter a username and password with privileges to access Sterling Product Manager.</p>

TABLE 32. Cron Job Configuration Page (Continued)

Field	Description
Frequency	How often will the job be run? Every three days? Every week? Every five minutes? and so on.
Start date and time/ End date and time	The effective start and end period between which dates and times the cron job will run. This, along with Frequency, determines when the job will run. For example, if you entered a frequency of three days, then the job will run every three days from the task start date and time until the task end date and time is reached. You can enter the same dates and times for both start date and time, in which case the job will be run only once, at a specific time.

4. Check the box next to **Active** to make the job available to be run.
5. Click **Save All Changes**.

To Modify a Job

<p>Attention: If you are running multiple instances of the Visual Modeler, creating or modifying a cron job will affect any of these instances running off the same Knowledgebase instance as the cron job.</p>
--

1. Click **Job Scheduler** in the System Administration panel on the Visual Modeler home page.
2. Click the name of a cron job to display the details of the selected job.
The details are displayed on the Cron Job Configuration page.
3. Enter the information about the job.
See Table 32, "Cron Job Configuration Page", on page 593 for a description of the fields.
4. Check the box next to **Active** to make the job available to be run.
5. Click **Save All Changes**.

To Run a Cron Job Immediately

You may need to sometimes run a cron job immediately.

1. Click **Job Scheduler** in the System Administration panel on the Visual Modeler home page.
2. In the list of cron jobs, identify the job that you want to run immediately.

3. Click **Run Now**.

The cron job will be immediately scheduled to run, but if jobs are ahead of it in the cron job queue, then it will not run until those jobs have completed.

To Delete a Job

1. Click **Job Scheduler** in the System Administration panel on the Visual Modeler home page.
2. Check the box next to the job(s) you want to delete.
3. Click the Delete icon (**X**) in the Actions column.

To View the History of a Cron Job

You may need to review how a cron job has run in the past. To do this:

1. Click **Job Scheduler** in the System Administration panel on the Visual Modeler home page.
2. In the list of cron jobs, identify the job whose history you want to view.
3. Click **Show History**.

Cron Jobs

If you install the Visual Modeler with the reference data, the installation includes the pre-defined cron jobs described in this section.

If you installed the Visual Modeler with minimal data, only the Cache Cleanup job is included. If you want to implement the other jobs, you must create them by following the steps in "To Create a Job" on page 593. The following sections contain the information needed to create these jobs.

Note that all cron job timeout values are specified in seconds.

Specifying -1 as a timeout value means that the cron job never times out.

Cache Cleanup

This group of properties determine the frequency and class of the cron job used to clean the cache.

Maintain Configuration

This cron job deletes saved configurations of the specified configuration type that are older than the specified age. The default age for deletion is 10 days.

TABLE 33. Maintain Configuration Cron Job

Cron Job Field	Entry
Program	com.comergent.apps.configurator.main.ConfigMaintenanceCron
Command Line Arguments	ConfigType=Config&AgeInDays=10
Cron Job Type	Application

This chapter describes the tasks associated with managing the Visual Modeler site. Site system administrators are responsible for:

- managing system users: see "System User Administration" on page 598
- managing the system profile: see "System Profile Administration" on page 599
- managing system properties: see "System Property Administration" on page 599
- managing system cron jobs: see "System Cron Jobs" on page 599
- reviewing the system status: see "System Status" on page 600

Overview

There is a distinction between *system administration* and *enterprise administration*:

- System administration is the responsibility of *system administrators*: they manage the basic system properties of the Visual Modeler and the system cron jobs.
- Enterprise administration is the responsibility of enterprise users: these users manage the building blocks of the enterprise e-commerce system: partners, users, products, price lists, storefronts, and so on.

A system administrator can manage:

- System users: see "System User Administration" on page 598
- System profile: see "System Profile Administration" on page 599
- System properties: see "System Property Administration" on page 599
- System cron jobs: see "System Cron Jobs" on page 599
- System status: see "System Status" on page 600

All these tasks are performed from the system administration home page.

To Access the System Administration Home Page

1. Point your browser to the System Administration URL. By default, this is:

`http://server:port/Sterling/en/US/enterpriseMgr/admin`

Check your site documentation to identify this URL.

2. Log in as a system administrator. When the Visual Modeler is first installed, the default username/password combination is admin/admin. If other system administrator users have been created, then you can log in using one of these userids.

<p>Attention: You must change at least the password of the admin user to protect the system from unauthorized access. We suggest that you create a different system administrator user, and then delete the admin user.</p>
--

3. From this page, you can perform the task described in the following sections.

System User Administration

To Create a System Administrator User

1. Log in as a system administrator.
2. Click **System Users**.
3. Click **Create User**.
4. Enter information for the new user as appropriate.
5. Click **Save**.

The new user information is saved.

6. You can verify that the new user has been successfully created, by logging out and logging back in as the newly-created user.

System Profile Administration

To Manage the System Administrator Profile

1. Log in as a system administrator.
2. Click **View Your Organization Profile**.
3. Modify the profile details as appropriate.
4. Click **Save**.

System Property Administration

System administrators can manage the system-level properties for the Visual Modeler, including configuring logging settings, job scheduler categories, Configurator settings (number of models to cache, default template directory and page template), whether or not to use session-based caching, and so forth.

To Update a System Property

1. Log in as a system administrator.
2. Click **System Services**.
3. Click the link to the set of properties that you wish to update.
4. Modify the values of properties as required.
5. Click **Save All and Return to List**.

System Cron Jobs

System administrators can manage system cron jobs. Enterprise administrators manage application cron jobs.

To Create a System Cron Job

1. Log in as a system administrator.
2. Click **Job Scheduler**.
3. Click **Create New Cron Job**.

4. Enter details for the new system cron job as appropriate.
5. Click **Save All Changes**.

System Status

To View the System Status

1. Log in as a system administrator.
2. Click **System Status**.
3. Review the system status details as appropriate.

Part 5: Tutorial

The chapters in this section of the guide provide step-by-step lessons on administering a storefront and creating product models.

In this lesson, you learn how to create a child storefront of the tenant storefront, Matrix, and how to administer that child storefront.

Storefront administration involves the following tasks:

- Creating a Storefront
- Creating a Storefront Administrator
- Creating Additional Storefront Administrators
- Setting Default Storefront Preferences
- Setting Storefront Business Rules
- Creating a Storefront Partner
- Creating a Storefront Partner Administrator
- Creating a Storefront Partner User

Creating a Storefront

1. Click the **Admins Login Here** link on the Matrix home page, then log in to the Visual Modeler as the **ajones** user (password: **ajones**).

The administration home page.

2. Click **Go** in the Search for Organization by Name panel.
The Profile List page displays.
3. Click **Create Storefront**.
4. Enter information for the new profile as shown in the following table.

TABLE 34. New Storefront Information

Field	Value
Profile name	AllNet Corp
Organization ID	M00212G
Address Line 1	4945, Central Ave.
Address Line 2	First Floor
City	Mill Valley
State/Province and Postal code	CA/94941
Country	USA
Skin Url (the name used in the URL to access this skin)	allnet

5. Click **Save**.
6. The Profile Detail page displays with new tabs.
7. Click the **Commerce** tab.
8. Under the **Payment Options** panel, check Account.
9. Under the **Shipping Options** panel, check Standard Shipping and Premium 2-Day.
10. Click **Edit** next to the skin ID.
The Edit Skin page displays. Use this page to upload the logo and CSS file for the AllNet skin. A skin is a combination of logo, CSS, and URL.
11. In the **Upload Logo Image to Server** field, specify the path to a folder that contains the storefront logo. For example, if you have loaded the Matrix storefront data, you can find the logos in your servlet container's **Sterling/hdocs/partnerlogos** directory. Alternatively, you can browse to the location where the image is kept by clicking **Browse...**

12. Click **Upload**.

The logo pathname displays in the Logo URL or Uploaded File Path field.

13. Upload a stylesheet file by specifying the path to a folder that contains the CSS file. If you do not specify a CSS file, the Matrix storefront's CSS is used.

14. Click **Save**.

15. Click **Return to List**.

Creating a Storefront Administrator

Now that you have created a storefront, you will create an administrator for this storefront. A storefront administrator is responsible for maintaining the profile of the storefront as well as its users.

1. Click the **AllNet Corp** link.

The Profile Detail page displays.

2. Click **View Users**.

The User List page displays.

3. Click **Create User**.

4. Enter the following information about the user:

- Username: bbunson
- Password: bbunson
- Confirm Password: bbunson
- Title: Mr
- First Name: Brad
- Last Name: Bunson
- Email: bbunson@farcom.com

5. Under the **User functions** panel, select the Manager, Program Management, Profile Administration, and Marketing Manager - Segmentation checkboxes.

6. Click **Save**.

The User Detail page displays with new tabs.

7. Click the **Addresses** tab.

8. Enter the following address information:
 - Address Line 1: 4945 Central Ave.
 - Address Line 2: First Floor
 - City: Mill Valley
 - State/Province and Postal Code: CA/94941
 - Country: USA
 - Select the Use as Ship To Address checkbox. A new checkbox appears for Set As Default Ship To Address. Select this checkbox as well.
9. Click **Save**.
10. Click **Logout**.
11. Notify the new storefront administrator of the storefront URL, the administrator's username and password.

Creating Additional Storefront Administrators

Now that you have created a storefront administrator (Brad Bunson), you can log in as this storefront administrator and create more users for the storefront. These users are also referred to as storefront administrators.

1. Log in to the storefront administration home page by pointing your browser to the appropriate URL. The URL for the AllNet storefront looks similar to the following:

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

The AllNet home page displays.

2. Click the **Admins Login Here** link and log in as Brad Bunson (bbunson/bbunson).

The AllNet administration home page displays.

Notice that except for the absence of the **Configuration Models** link and the **Advisor Flows and Questionnaires** link, the AllNet administration home page is similar to the Matrix administration home page.

3. Click **System Users** under the System Administration panel.

The **User List** page displays.

4. Click **Create User**.

The **Create New User** page displays.

5. Enter the following information about the user:

- Username: mhailey
- Password: mhailey
- Confirm Password: mhailey
- Title: Ms
- First Name: May
- Last Name: Hailey
- Email: mhailey@farcom.com

6. Under the **User functions** panel, select the Commerce, Sales Executive, and Profile Administration checkboxes.

7. Click **Save**.

8. Click **My Home** to return to the home page.

Setting Default Storefront Preferences

When you create the AllNet storefront, the storefront is automatically assigned the following default preferences, which are specified in the CMGT_USER_PROPERTY, CMGT_USER_PROPERTY_LOCALE, and CMGT_USER_PROPERTY_UI_HINT tables.

- User Cart Mode: **Multiple Carts**
- Shopping Cart Display: **Simple View**
- Checkout Type: **Single Step**
- Home Page View: **Portal View**
- Sold-To Address: **Yes**
- Availability: **Yes**
- Delivery Date: **Yes**
- Taxable: **Yes**
- Availability Text Display: **Display Number Available**

However, you can override these default preferences for the storefront.

For the purpose of this tutorial, you will modify the values of the following user preferences:

- User Cart Mode: **Single Cart**
- Home Page View: **Catalog View**

To modify the preferences for the AllNet storefront, perform the following steps:

- For Oracle:
 - a. Find the CMGT_USER_PROPERTY table in the database.
 - b. Under the VALUE column of the userCartMode property, enter Single_Cart. Similarly, enter Catalog_View under the VALUE column of the homePageView property.
 - c. Restart your server.
- For SQL Server 2005:
 - a. To modify the value of the User Cart Mode user preference to Single Cart, execute the following SQL statement against the database:

```
update CMGT_USER_PROPERTIES set value='Single_Cart'  
where name='userCartMode'
```

- b. To modify the value of the Home Page View user preference to Catalog View, execute the following SQL statement against the database:

```
update CMGT_USER_PROPERTIES set value='Catalog_View'  
where name='homePageView'
```

- c. Click **Logout** on the AllNet administration home page.

Log in to the AllNet administration home page.

Setting Storefront Business Rules

In this section, you set business rules that are specific to the AllNet storefront. When you create the AllNet storefront, you get by default the business rules settings of the tenant (Matrix) storefront. You can override these business rules and set them afresh.

For the purpose of this tutorial, you will set the values of the following business rules for the Saved List property:

- Enable Alternative items link in: Templates, Wish Lists, Registries, Cart
 - Enable Complementary items in: Templates, Wish Lists, Registries
1. Click the **Business Rules** link under the System Administration panel.
The **Business Rules** page displays.
 2. Click the **Saved List** link.
The Business Rules page redisplay with the business rules and their default values on the right panel.
 3. Select the following checkboxes for the **Enable Alternative items link in** business rule:
 - **Templates**
 - **Wish Lists**
 - **Registries**
 - **Cart**
 4. Select the following checkboxes for the **Enable Complementary items link in** business rule:
 - **Templates**
 - **Wish Lists**
 - **Registries**
 5. Click **Save**.
 6. Click **My Home** to return to the home page.

Exercise

Create another product with the following information:

- Product ID: ANLP-7510
- Product Name: AllNet 7510 Notebook
- Description: A highly power-packed solution
- Component Type: Normal
- Status: Released
- Start Date: Enter the current date

- End Date: Enter a date 100 years from the current date.

Creating a Storefront Partner

1. Click **Go** in the Search for Organization by Name panel.
The Profile List page displays.
2. Click **Create Profile**.
3. Enter information as shown in the following table.

TABLE 35. New Profile Entries

Field	Value
Profile name	ITech Solutions
Profile type	Retailer
Profile level	Gold
Organization ID	IT-23512
Address Line 1	123, Park Street
City	Richmond
State/Province and Postal code	VA/94941
Country	USA

4. Click **Save**.
The Profile Detail page for ITech Solutions displays with new tabs.
5. Click the **Business** tab.
6. From the Territories drop-down list, select North America.
7. From the Customer Types drop-down list, select the following:
 - General
 - Government
 - Commercial
 - Pharmaceutical
 - Telecommunications
 - High Technology

- Education
- 8. Under the Contracts panel, enter the following information about business agreements between AllNet Corp and ITech Solutions:
 - Name: ABC-245ST
 - Type code: XYZ-7890
- 9. Click **Save**.
- 10. Click the **Pricelists** tab.
- 11. Select the checkbox next to AllNet Price List for All, then click **Save**.

Creating a Storefront Partner Administrator

In this section, you create an administrator for the partner you just created. This partner administrator is responsible for profile maintenance of the partner and its users.

1. Click **View Users**.

The User List page displays.
2. Click **Create User**.
3. Enter information as shown in the following table:

TABLE 36. Entries for New User

Field	Value
Username	rjones
Password	rjones
Confirm Password	rjones
Title	Ms
First Name	Rosy
Last Name	Jones
Email	rjones@itech.com

4. Select the **Manager** checkbox.
5. Select the **Profile Administration** function.
6. Click **Save**.

7. Click **Logout**.

Creating a Storefront Partner User

In this section, you learn how to create a user for Itech Solutions.

1. From the **Login** panel, log in as rjones.
The Partner User Home page displays.
2. Click **Update User Accounts**.
3. Click **Create User**.
The User Detail page displays.
4. Enter information as shown in the following table.

TABLE 37. User Detail Header Information

Field	Value
Username	adesai
Password	adesai
Confirm Password	adesai
Title	Ms
First Name	Anita
Last Name	Desai

5. Check the **Commerce** function.
6. Click **Save**.
7. Click **Logout**.

The Visual Modeler application enables you to create configurable product models. Customers can customize a product by selecting choices that the product offers to ensure that the product precisely meets their needs.

Before creating a configurable product model, you must ensure that all the products to be used in the model are already created using the Business Center application. For more information about creating a configurable product, refer to the *Business Center: Item Administration Guide*.

In this lesson, we create a configurable product model. This comprises the following steps:

1. Create the product model
2. Test the product configuration experience

After a configurable product model is created, it needs to be associated to a product. For more information about associating a model to a bundle product, refer to the *Business Center: Item Administration Guide*.

Once we have created and tested the basic model, the lesson continues by demonstrating the use of more advanced configuration options and UI controls that help you manage the customer's experience as they configure the product. These topics are:

- "Properties" on page 617

- "Rules" on page 620
 - "Creating a Rule" on page 623
 - "Using Rules to Control Display of Option Items" on page 625
- "UI Controls" on page 631
 - "Display Properties" on page 631
 - "Tabular Displays" on page 632
 - "Calculated Property Values" on page 633
 - "User-Entered Values" on page 636
 - "Images" on page 639

Create the Model

1. In the Product Configuration Administration panel, click **Configuration Models**.

The Visual Modeler administration page displays.

2. Click **New Model Group**.
3. Enter Computers in the Name and Description fields and click **Save**.
4. In the model group hierarchy tree, select the Computers node.
5. Click **New Model Group**.
6. Enter Workstations in the Name and Description fields and click **Save**.

Note that we are creating a hierarchy of model groups that mirror the hierarchy of product categories. This is not necessary, but can often help to maintain the organization of models and the products to which they correspond.

7. In the model group hierarchy tree, select the Workstations node.
8. Click **New Model**.
9. Click **Browse....**
10. In the product picker window that opens up, navigate to the Computers -> Workstations product category and select the MXWS-7700 product.

11. Click **Done**.

Notice that the Assigned Product ID field now reads MXWS-7700 and the Name and Description fields are populated with the name and description of the MXWS-7700 product.

12. Click **Save**.

The new model, MXWS-7700, is displayed in the Models and Groups panel.

13. Select the MXWS-7700 model and click the **Edit** icon on the Visual Modeler toolbar.

14. The Model Detail page is now displayed.

To begin our modeling, we will create an option class and two option items in the option class: this will enable customers to choose between two different monitors to go with their workstation.

15. Click **New Option Class**.

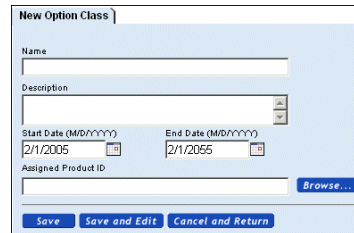


FIGURE 107. Visual Modeler Model Detail Page: New Option Class Tab

16. Enter Monitors in the Name field and enter “Please select a monitor” in the Description field.

17. Click **Save**.

The option class is created and added to the model tree in the Model navigation panel. Now we will create two option items in this option class: these will represent the selectable items that user may select.

18. Select the Monitors option class in the model tree.

19. Click New Option Item.

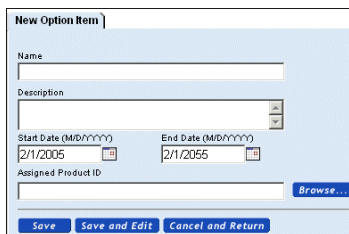


FIGURE 108. Visual Modeler Model Detail Page: New Option Item Tab

20. Enter Optiquest Q95 in the Name and Description fields.
21. Click **Save**.
22. The New Option Item tab is re-displayed. This time, enter Optiquest Q115 in the Name and Description field.
23. Click **Save**.
24. If you examine the Model navigation tree, you see that the model is now displayed with one option class and two option items.

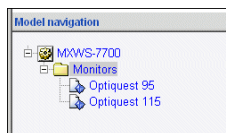


FIGURE 109. Model Navigation Tree

Now we are ready to compile and test this simple model.

25. Click the **Compile and Test** icon on the Visual Modeler toolbar.

Behind the scenes, an XML file is generated that holds the structure of this model. Each time we make changes to the model, we must re-compile the model so that our changes will become part of the customer's configuration experience.

A second browser window is displayed: this is the configuration experience that customers will have when they configure the MXWS-7700 product.

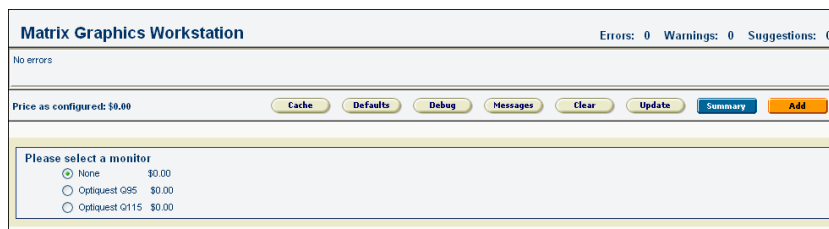


FIGURE 110. Visual Modeler Test Model Page

As you can see, this page provides a UI in which customers can select either of the two monitors (or neither), and the text displayed on the page is the descriptions that we provided for the option class and option items.

26. Close this window.

This completes the creation of the basic model. Next we must hook it up to the product.

27. Click **My Home** to return to the Enterprise Home page.

28. In the Product manager, navigate to the Product Detail page for the MXWS-7700 product.

29. In the Component Type drop-down list, select Configurable.

30. Notice that in the Model drop-down list, "Computers/Workstations/MXWS-7700" is selected automatically.

31. Click **Save Changes**.

You have completed the basic setup of this configurable product. If a customer selects the MXWS-7700 product and adds it to their shopping cart, then they will be able to configure it by choosing between the two monitor models provided.

The following sections will provide an in-depth tutorial of the various advanced features of the Visual Modeler. They cover:

- "Properties" on page 617
- "UI Controls" on page 631

Properties

Most of the customer's experience of configuring a product is determined by properties: these are attributes of the model, its option classes, and option items,

and they are used to determine how rules fire, the behavior of the UI, and information that can be displayed to the user. The Visual Modeler provides a set of built-in properties that control the behavior of the Sterling Configurator and how the model is presented to the end-user. You can also define properties that the Configurator will use to ensure that the user's product configuration choices make sense.

This section describes how to define and attach properties to a model. You define a property at the model group or model level, and you attach a property to a model, option class, or option item. The basic steps are:

1. Determine at what level to define and attach a property. Where you attach the property determines where you can use it: properties are available for use in the model group and model hierarchy beneath the point at which they are defined. For example, you may want to define a property, `MonitorSize`, at the `Monitors` option class level so that it is available for use with each of the option items (types of monitors) in that option class.
2. Once you define the property, then you attach it at the appropriate level of the model hierarchy and, if appropriate, assign a value to it. For example, once you have defined the `MonitorSize` property, then you attach it to the option item defining a specific monitor and set the value representing the size of the monitor.

We will define properties at the model level for the monitors available with the MXWS-7700 Workstation product. Once the properties are defined, we will assign property values and attach them at the option item level.

Defining Properties for the MXWS-7700 Model

1. In the Visual Modeler, navigate to the Workstations level of the Model Group hierarchy.

MXWS-7700 displays in the Models and Groups panel.

2. Select MXWS-7700, then click the Edit Model icon in the Visual Modeler toolbar.

MXWS-7700 displays at the top of the Model Navigation panel, and the **General Info** tab for the model displays.

3. Click the **Properties** tab. The Properties page displays as shown in the following figure.
4. Click the **Define** tab.

The Define Properties page displays, as shown in the following figure.

5. Define the MXWS-7700 properties:
 - a. Enter the property name: MonitorSize.
 - b. Choose Number from the Type drop-down list.
 - c. Click Add.

The MonitorSize property displays in the Defined Properties list.

6. Add the rest of the MXWS-7700 model properties using the property names and types shown in the following table.

TABLE 38. MXWS-7700 Properties

Name	Type
MaximumResolution	String
MonitorWeight	Number
MaximumWeight	Number
RequiresXVGA	Number

When you have finished defining and adding the properties, the Define Properties page displays as shown in the following figure.

7. Click Save All Changes.

The properties are now part of the MXWS-7700 model.

Attaching Properties

This section assumes that you are still editing the MXWS-7700 model in the Visual Modeler.

First, we will attach a property that applies to the entire model.

1. In the Model Navigation panel, click the MXWS-7700 link.

The General Info page displays.

2. Click the **Properties** tab.

The Properties page displays as shown in the following figure.

Notice the two lists: Unattached Properties and Attached Properties.

3. Choose MaximumWeight from the Unattached Properties drop-down list.
4. Enter 0 for the property value.

5. Click Attach.

The MaximumWeight property displays in the Attached Properties list.

6. Click Save All Changes.

Next, we will attach the properties that apply to the option items for the Monitors option class.

1. In the Model Navigation panel, click the Optquest Q95 link.

The **Properties** tab displays as shown in the following figure:

Notice the two lists: Unattached Properties and Attached Properties.

2. Choose MonitorSize from the Unattached Properties drop-down list.
3. Enter 19 in the Value field, then click Attach.

The MonitorSize property displays in the Attached Properties list.

Continue attaching properties to the option items as follows:

TABLE 39. Option Item Property Values

Option Item	Property	Value
Optquest Q95	MonitorSize	19
	MaximumResolution	1024x768
	MonitorWeight	7.2
Optquest Q115	MonitorSize	21
	MaximumResolution	1280x1024
	MonitorWeight	8.3

When you have finished attaching properties to each option item, click **Save All Changes**.

Rules

The Visual Modeler provides you with the ability to precisely control the customer's selections so that the selections that they make are compatible with each other and that as users make selections, they see selections that are related.

For example, suppose that you want to ensure that customers pick a good graphics card to go with their monitor. You can create a rule based on their monitor selection that displays a graphics card option class that displays only compatible graphics

cards. Alternatively, you can create a constraint table that specifies which monitor can be selected with which graphics card.

Before continuing this section, create a new option class and three option items as follows:

TABLE 40. Graphics Card Option Class

Option Class	Property Definition	Option Item	Property Value
Graphics Cards	Resolution	GC-1000	Resolution: VGA
		GC-2000	Resolution: XVGA
		GC-3000	Resolution: XVGA

Define the Resolution property at the Graphics Cards level of the model hierarchy as follows:

1. In the Model navigation panel, click the MXWS-7700 link.
The **General Info** tab displays.
2. Click New Option Class.
The New Option Class page displays.
3. Enter Graphics Cards in the Name field and in the Description field.
4. Click Save and Edit.
The **General Info** tab for the new Graphics Cards option class displays.
5. Click the **Properties** tab.
The **Properties** tab displays.
6. Click New Property.
The Define New Property pop-up window appears, as shown in the following figure.

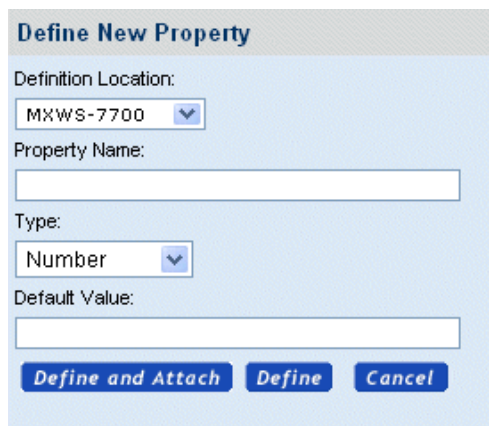
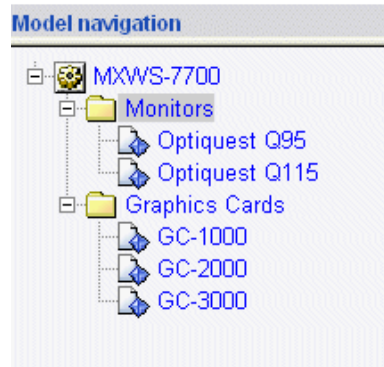


FIGURE 111. Visual Modeler Define New Property Pop-Up Window

7. Choose Model Groups from the Definition Location drop-down list.
8. Enter Resolution for the Property Name.
9. Choose String from the Type drop-down list.
10. Click Define.

Define the three new option items and attach the Resolution property to them using the values in the table "Graphics Card Option Class" on page 621. Define the option items as described in "Create the Model" on page 614 and attach the Resolution property as described in "Attaching Properties" on page 619. Be sure to click Save All Changes as you complete each step. When you are finished, the Model navigation panel displays as shown in the following figure.



In the Model navigation panel, click the Graphics Cards link, then click the **Display** tab. Choose Invisible from the Option Class Display drop-down list, then click Save All Changes.

On the Monitors option class, create a new property as follows:

TABLE 41. IsPicked Property Definition

Option Class	Property Definition	Option Item	Property Value
Monitors	IsPicked	Optiquest Q95	1
		Optiquest Q115	1

Creating a Rule

The first rule we make is to show the Graphics Card option class if a user selects one of the monitor option items.

1. Navigate to the model MXWS-7700.
2. Click the Rules tab.
3. Click the Define sub-tab.
4. Click **New...**

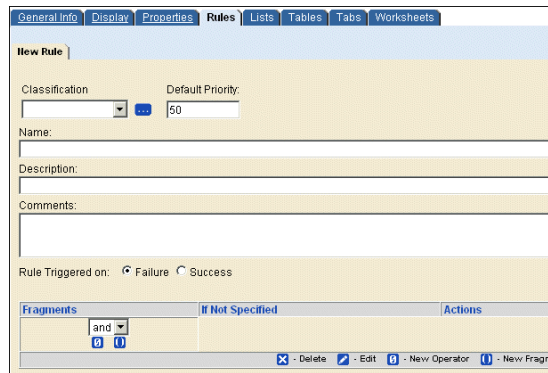


FIGURE 112. Model Detail Page: New Rule Tab

5. Enter the following information:
 - a. Name: Display Graphics Card
 - b. Description: Display the Graphics Card option class if a monitor is selected.
6. Select Rule Triggered on: Success.
7. Click **Save and Edit**.
8. Click the New Fragment icon.
The New Fragment page displays.
9. Define the fragment:
 - a. Choose value from the Function1 drop-down list.
 - b. Choose IsPicked from the Property1 drop-down list.
 - c. Choose = from the Operator drop-down list.
 - d. Choose literal from the Function2 drop-down list.
 - e. Enter 1 in the Property2 drop-down list.
 - f. Choose Rule is false from the If not specified drop-down list.
10. Click **Save and Return**.

We want the rule to assign the `_isVisibleable` property to the Graphics Cards option class if the rule is true, so define the assignment action like this:

11. In the Assignment Actions panel, select `_isVisibleable` property from the drop-down list.
12. Enter 1 for the Value.
13. Click the ... button next to the Assign To field, and in the pop-up window, navigate to the Graphics Card option class. Select it and click **Done**.
The Assign To field is populated with the value `“*.Graphics Cards”`.

14. Click **Add Item**.
15. Click **Save**.

Having defined the rule, now we must attach it to the model.

16. Select the Model MXWS-7700 from the navigation panel, and click the **Rules** tab.
17. On the **Attach** sub-tab, select the Display Graphics Card rule from the drop-down list, and click **Attach**.
18. Click **Save All Changes**.

Now compile and test the model. You see that the Graphics Card option class is hidden until you select a monitor, and then it is displayed so that a graphics card can be selected.

Using Rules to Control Display of Option Items

You can use assignment actions to control the display of option items: this gives you the ability to ensure that customers only make valid selections. In "Creating a Constraint Table" on page 629 we demonstrate an alternate approach using constraint tables. In this section, we use a rule with an assignment action to ensure that a customer will only pick valid combinations of graphics cards and monitors.

For the purposes of this example, our rule will say that if you select a monitor whose maximum resolution is 1280 x 1024, then you must select a graphics card that supports XVGA.

1. Navigate to the model MXWS-7700.
2. Click the Rules tab.
3. Click the Define sub-tab.
4. Click **New...**
5. Enter the following information:

- a. Name: Display Compatible Graphics Cards
 - b. Description: This rule ensures that only graphics cards that support each monitor are displayed.
6. Select Rule Triggered on: Success.
 7. Click **Save and Edit**.
 8. Click the **New Fragment** icon.
 9. Specify the fragment as follows:
 - a. Choose value from the Function1 drop-down list.
 - b. Choose MaximumResolution from the Property1 drop-down list.
 - c. Choose = from the Operator drop-down list.
 - d. Choose literal from the Function2 drop-down list.
 - e. Enter 1280x1024 in the Property2 field.

Specifying the fragment in this way is equivalent to the following formula:
`value(Maximum Resolution) = literal(1280 x 1024)`

 - f. Choose Rule is false from the If not specified drop-down list.
 10. Click **Save and Return**.

If the rule is true, then we want the rule to assign the RequiresXVGA property to the model, and then have a rule that ensures that only compatible graphics cards option items are displayed, so define the assignment action like this:

11. In the Assignment Actions panel, select the RequiresXVGA property from the drop-down list.
12. Enter 1 for the Value.
13. Click the ... button next to the Assign To field, and in the pop-up window, navigate to the MXWS-7700 model. Select it and click **Done**.

The Assign To field is populated with the value “MXWS-7700”.
14. Click Add Item.
15. Click **Save**.
16. Now attach this rule to the model.

So far, we have created a rule that tells the model that if certain monitors are selected, then the graphics card that is selected must support XVGA. Now we

create a rule that can be attached to each graphics card option item that determines whether it can be displayed.

1. Navigate to the model MXWS-7700.
2. Click the Rules tab.
3. Click the Define sub-tab.
4. Click **New...**
5. Enter the following information:
 - a. Name: Display if Support XVGA
 - b. Description: Display this card if XVGA support is required and the card supports XVGA.
6. Select Rule Triggered on: Failure.
7. Click **Save and Edit**.
8. Click the New Fragment icon.
9. Specify the fragment as follows:
 - a. Choose value from the Function1 drop-down list.
 - b. Choose RequiresXVGA from the Property1 drop-down list.
 - c. Choose = from the Operator drop-down list.
 - d. Choose literal from the Function2 drop-down list.
 - e. Enter 1 in the Property2 field.

Specifying the fragment in this way is equivalent to the following formula:
value(RequiresXVGA) = literal(1) in any location
10. Set **If not specified** to Rule is true.

In effect, this says that if the RequiresXVGA property is not set, then assume that all graphics cards are valid selections.
11. Click **Save and Return**.
12. Click the New Fragment icon.
13. Specify the fragment as follows:
 - a. Choose propval from the Function1 drop-down list.
 - b. Choose Resolution from the Property1 drop-down list.

- c. Choose = from the Operator drop-down list.
- d. Choose literal from the Function2 drop-down list.
- e. Enter XVGA in the Property2 field.

Specifying the fragment in this way is equivalent to the following formula:
`propval(Resolution) = literal(XVGA) in any location`

Note that we have to use the `propval` function here rather than `value`: this is because the option item will not have been picked at the time the rule fires.

- 14. Set **If not specified** to Rule is true.
- 15. Click **Save and Return**.
- 16. In the Assignment Actions panel, select `_isVisibleable` property from the drop-down list.
- 17. Enter 0 for the Value.
- 18. Leave the Assign To field value blank. This is to indicate that the property is at the node at which the rule is attached.
- 19. Click Add Item.
- 20. Click **Save**.
- 21. Now attach this rule to the each graphics card.

Before compiling our model, the last thing we have to do is to manage the order in which the rules fire. We want to ensure that the rule that tests to see if XVGA is required fires before the rules that determine if each graphics card option item is compatible.

- 22. Navigate to the Model node.
- 23. Click the Rules tab.
- 24. Click the Firing Sequence sub-tab.
- 25. Change the Priority value of the Display Compatible Graphics Cards rule to 10.

This ensures that this rule will fire first.

- 26. Click Save All Changes.

Now compile and test the model. You see that the Graphics Card option class is hidden until you select a monitor, and then it is displayed so that a graphics card can

be selected. If you select the Optquest Q115 monitor, then the GC-1000 graphics card is not displayed.

In the next section, "Creating a Constraint Table" on page 629, you will create a table which constrains which option items are compatible with each other. Before proceeding to the next section, remove the rule Display if Support XVGA from each of the graphics cards.

Creating a Constraint Table

Not all of the graphics cards may be compatible with all of the monitors, and so you want to specify what combinations of monitors and graphics cards are acceptable. In this section, we create a constraint table to express this.

Assume that the following combinations of Graphics Card and Monitor are compatible:

TABLE 42. Compatible Selections From Graphics Cards and Monitors

Compatible?	Optquest Q95	Optquest Q115
GC-1000	Yes	No
GC-2000	Yes	Yes
GC-3000	Yes	Yes

You can express constraint tables either by specifying what option items can be selected together or which cannot. In this example, it is easier to specify that GC-1000 and Optquest Q115 cannot be selected together: the other selections are assumed to be compatible.

1. Navigate to the model MXWS-7700.
2. Click the Tables tab.
3. Click **New...**
4. Enter the following information:
 - a. Name: Graphics
 - b. Description: Constrains the selection of graphics cards and monitors.
 - c. Message: You cannot select this combination of graphics card and monitor.
5. Click **Save Changes**.
6. Click the Records tab.

7. Select Monitors from the Table Column Name drop-down list and click **Add**.
8. Select Graphics Cards from the Table Column Name drop-down list and click **Add**.

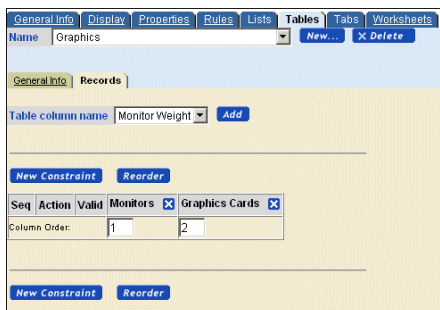


FIGURE 113. Model Detail Page: Tables Tab and Records Sub-Tab

9. Click **New Constraint**.
10. Click **Edit**.

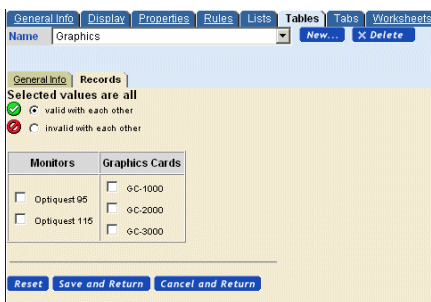


FIGURE 114. Model Detail Page: Tables Tab and Records Sub-Tab

11. Select the **invalid with each other** radio button, and check the Optiquest Q115 and GC-1000 check boxes.
12. Click **Save and Return**.

Now compile and test the model. You see that the Graphics Card option class is hidden until you select the Optiquest Q115 monitor, and then it is displayed so that a graphics card can be selected. You see that the GC-1000 option item is displayed

with a clickable icon that indicates that it should not be selected. If you do select it, then an error message is displayed. You can click the icon to ask the Visual Modeler to help resolve the conflict: in this case, it will suggest an alternate selection of monitor.

UI Controls

The Visual Modeler provides a rich set of controls that can provide a flexible and attractive UI to help customers make their selections. This section describes how to use them. It covers:

- "Display Properties" on page 631
- "Tabular Displays" on page 632
- "User-Entered Values" on page 636
- "Images" on page 639

Display Properties

Each model, option class, and option item has a set of properties that determine how the configurable product will be displayed to the customer: these are known as display properties. Every display property corresponds to a UI property as indicated below.

Pre-Pick Guiding Text

Suppose that you want to provide some text to help customers make a selection, but want to remove the text once the customer has done so.

1. Navigate to the Monitors option class.
2. Click the **Display** tab.
3. Enter the following in the Pre-Pick Guiding Text field: "The larger the monitor, the easier it is to manage multiple displays on it."
4. Click **Save All Changes**.

If you now compile and test the model, then you will see that this text is displayed when you first display the model. However, if you select one of the monitors, then when the page is re-displayed, you can see that the text is now removed.

This display property corresponds to the UI: PRE_PICK GUIDING TEXT property.

Ignore in Quote

By default, option classes and option items are displayed in the customer's cart when they completed their product configuration and put their configured product in their cart. If you do not want an option class to be displayed in the customer's cart, then do this:

1. Navigate to the Monitors option class.
2. Check the Ignore In Quote check box.
3. Click **Save**.

If you now compile and test the model, then you will see that the monitor class is displayed. However, if you click the Summary button, then the Monitors option class is not displayed on the Summary page (though any option items picked are).

This display property corresponds to the UI: IGNORE IN QUOTE property.

Tabular Displays

Suppose that you would like to display the monitor option items with some of their properties to help users choose among them. You can do this in the form of a table as follows.

1. Navigate to the Monitors option class.
2. Click the **Display** tab.
3. Select Tabular Display from the UI control drop-down list.
4. Click **Save All Changes**.
5. In the Tabular Display Control Settings section of this page, enter:
 - a. Column Headings: Size;Resolution
 - b. Column Properties: MonitorSize;MaximumResolution
 - c. Column Alignment: Left;Left
6. Click **Save All Changes**.
7. Click **Compile and Test**.

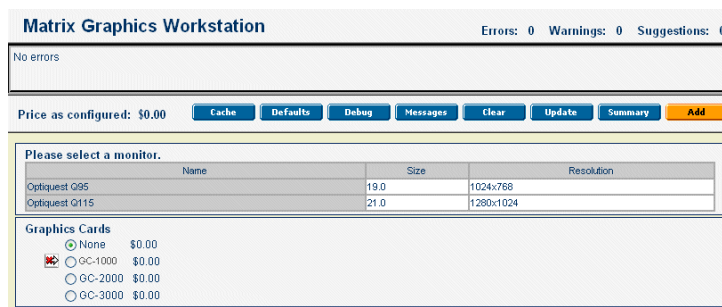


FIGURE 115. Tabular Display of Properties

Note that the option items are not selectable in this display. You probably want to add a second option class that makes them selectable.

Calculated Property Values

The Visual Modeler provides a simple, yet powerful, means to use property values to calculate other property values. In this section, we show how use this mechanism to display extra information to customers. Suppose that you know that for each monitor, you know the diagonal linear dimension (d) of the monitor, but you want to present to the customer the total area (A) of the monitor screen. This can be approximately calculated as $A = d^2/2$. You can do this along these lines:

1. Navigate to the Monitors option class.
2. Define a numeric property called Monitor Area.
3. Attach this property to the two monitors.
4. Navigate to the Monitors option class.
5. Click the Display tab.
6. Change the table column properties as follows:
 - a. Column Headings: Size;Area;Resolution
 - b. Column Properties: MonitorSize;MonitorArea;MaximumResolution
 - c. Column Alignment: Left;Left;Left
7. Click **Save All Changes**.
8. Navigate to the model node.
9. Click the **Worksheets** tab, and then create a worksheet as follows:

10. Click **New...**
11. Enter the following information:
 - a. Name: Calculate Area
 - b. Click **Create**.
12. Click **Add Column** and add first Monitor Size and then Monitor Area.
13. Click Add Row and, using the entity picker, add the monitor option items.
14. In the first row and Monitor Area column, click the Edit Property Value button.

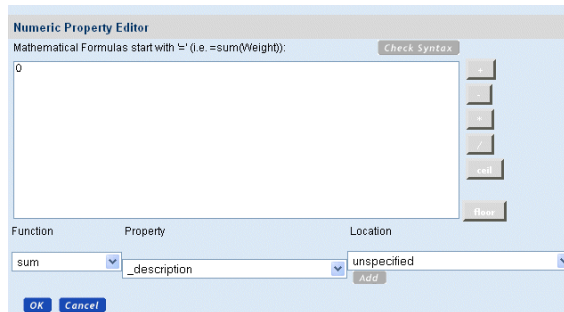


FIGURE 116. Numeric Property Editor Window

15. Enter “=” as the first character in the text area.
16. In the Numeric Property Editor window, select the Function value, MonitorSize Property, and the unspecified Location and click **Add**.
17. Click * from the mathematical symbols along the side.
18. In the Numeric Property Editor window, select the Function value, MonitorSize Property, and the unspecified Location and click **Add**.
19. Click / from the mathematical symbols along the side.
20. Enter 2.

You should see the following in the text area:

```
value("Monitor Size")*value("Monitor Size")/2
```
21. Click **OK**.
22. In the second row and Monitor Area column, click the Edit Property Value button and repeat the steps above for this monitor.

23. Click **Save All Changes**.

If you now compile and test the model, then you will see that the Monitors option class is displayed as a three-column table, and the Area column is calculated from the Size column.

You can also use Java classes in the Numeric and String Editor windows. For example, suppose that the monitors are circular, and the Monitor Size (d) property is the radius of the monitor. Then the area (A) should be calculated as $\pi * d * d$.

You can modify the Monitor Area property formula to read:

```
=java.lang.Math.PI*value("Monitor Size")*value("Monitor Size")
```

To make this more readable, you can use a string-formatting expression to define a more readable property as follows.

1. Navigate to the Monitors option class.
2. Define a new String property called MonitorAreaString.
3. Navigate to the MXWS-7700 model and click the Worksheet tab.
4. Add a new column to the worksheet by adding the MonitorAreaString property.
5. In the first row and MonitorAreaString column, click the Edit Property Value button.

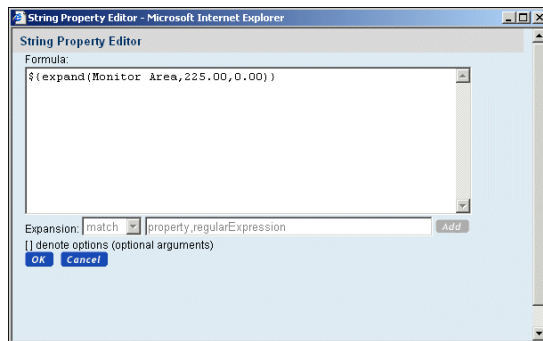


FIGURE 117. String Property Editor Window

6. Enter “`$(expand(Monitor Area,225.00,0.00) inches)`” in the text area.
7. Click **OK**.

8. Now navigate to the Column Properties field on the Monitors class Display tab, and change it to:

`MonitorSize;MonitorAreaString;MaximumResolution`

If you now compile and test the model, then you will see that the Monitors option class is displayed as a three-column table, and the Area column is calculated from the Size column and displayed in the form 1134.11 inches.

User-Entered Values

You may want to permit customers to enter values for properties: these can be used to check user requirements against rules that determine whether option items match the requirements. To do so, you must specify that the relevant option class UI Control is a User Entered Value, and then at the option item level, specify how the user-entered value is bound to a property. Typically, you want to bind the value to a property that can be used in a model rule.

For example, suppose that you want to enable users to specify a maximum weight for their computer monitor. You can do this as follows:

1. Create an option class called MonitorWeight.
2. On its Display tab, select User Entered Values from the UI Control drop-down list.
3. Set the Number of Columns display property to 2.
4. Click **Save All Changes**.
5. Create a single option item below this option class: call it Weight.
6. On the Display tab for Weight, in the User Entered Value Settings section, enter:
 - a. For the User Entered Value Type, select Numeric
 - b. Assign Value to Property: *.MaximumWeight
This references the MaximumWeight property which is attached at the model level.
 - c. Text before Entry Field: Enter the maximum weight for the monitor in kg.
7. Click Save All Changes.

If you compile and test this model, you will see the following option class section:

**FIGURE 118. User Entered Value Option Class**

To show how the user-entered value can be used:

8. Navigate to the Model level, and then create the following rule:
 - Name: Maximum Weight at Model
 - Triggered on: Failure
 - Fragment: In Function1, select value, MonitorWeight, any from the drop-down lists, select ">" from the Operator drop-down list, and in Function2, select value, MaximumWeight, relative, and select Rule is true from the If not specified drop-down list, so that the rule reads:

value(MonitorWeight) > value(.MaximumWeight)
 - Error message: The selected monitor exceeds your specified maximum weight.
9. Save this rule.
10. Attach this rule at the MXWS-7700 node.

When you compile and test this model, you will see that depending on your choice of monitor and the value you enter for the maximum weight of the monitor, an error message is displayed when the monitor weight exceeds the maximum weight.

What happens here is that when a customer specifies a value in the Weight text field and updates the model, the value they enter is assigned to the MXWS-7700.MaximumWeight property, and then the rule compares this value to the value of the MonitorWeight property defined anywhere in the model. In this case, the only place where MonitorWeight has been attached to the nodes is at the two monitor option items, and so the value of the MonitorWeight property at the selected monitor node is used.

An alternative approach to writing this rule is to specify that the MonitorWeight property should be retrieved from the point where the rule is attached and then attach this version of the rule to each of the monitor option items. Try this as follows:

1. Detach the rule Maximum Weight at Model rule from the model, so that it does not participate in the rule firing process.
2. Create the following rule:

- Name: Maximum Weight at Monitor
 - Triggered on: Failure
 - Fragment: In Function1, select value, MonitorWeight, relative from the Location1 drop-down list, select ">" from the Operator drop-down list, and in Function2, select value, MaximumWeight, relative, and select Rule is true from the If not specified drop-down list, so that the rule reads:

value(.MonitorWeight) > value(.MaximumWeight)
 - Error message: The selected monitor exceeds your specified maximum weight.
3. Save this rule.
 4. Attach this rule at the monitor option item nodes.

When you compile and test this model, you will see that depending on your choice of monitor and the value you enter for the maximum weight of the monitor, an error message is displayed when the monitor weight exceeds the maximum weight.

What happens in this case is that when a customer specifies a value in the Weight text field and updates the model, is that the value they enter is assigned to the MXWS-7700.Maximum Weight property, and then the rule compares this value to the value of the Monitor Weight property defined at the node where this rule is attached. In this case, the only place where Monitor Weight has been attached to the nodes is at the two monitor option items, and so the value of the Monitor Weight property at the selected monitor node is used.

Before proceeding with this lesson, detach the Maximum Weight at Monitor rule from the Monitor option items.

Restricting User Entered Values

Some of the time you may want to restrict the possible values that a customer can enter in a user-entered value field. You can do this using the Allowed Values display property.

1. Navigate to the Weight option item of the Monitor Weight option class and click the Display tab.
2. In the Allowed Values display property field, enter: 0-20.

When you compile and test this model, you see that the Weight text field is now a drop-down list which is populated by integer values from 0.0. to 20.0. A customer can only select one of these values, and when they do, their selection is automatically submitted to the server.

As another example, suppose that you want another user-entered value field for a customer's color preference, and again you want to indicate that the customer must select only from a choice of colors. Do the following:

1. Create an option class called Color.
2. On its Display tab, select User Entered Values from the UI Control drop-down list.
3. Click **Save All Changes**.
4. Create a single option item below this option class: call it Color Choice.
5. On the Display tab, in the User Entered Value Settings section, enter:
 - a. For the User Entered Value Type, select String
 - b. Enter Black,Blue,Green,Red,White in the Allowed Values field.
 - c. Assign Value to Property: *.Color
This references the Color property which you can define and attach at the model level.
 - d. Text before Entry Field: Select your preferred color.
6. On the Properties tab, select the UI: SUPPRESS UEV NONE VALUE property and enter yes for its value, and click **Attach**.
7. Click **Save All Changes**.

When you compile and test this model, you see the new Color option class, and a drop-down list of values from which the customer can make their selection. Note that None is not a selectable item.

Images

You can associate images with a model, option class, or option item simply by specifying the Image display property: this takes as values relative URLs or absolute URLs:

- If you begin the URL with “http://”, then the URL is assumed to be absolute;
- If you begin the URL with “/”, then the URL is interpreted relative to the servlet container;

- If you begin the URL without either, then the URL is interpreted relative to the current URL.

TABLE 43.

Image Value	URL
http://webserver:port/images/4Stars.gif	http://webserver:port/images/4Stars.gif
/images/4Stars.gif	http://server:port/images/4Stars.gif
4Stars.gif	http://server:port/en/US/images/4Stars.gif

At the model or option class level, the Image display property corresponds to the UI: ICON GRAPHIC property. At the option item level, the Image display property corresponds to the UI: ITEM IMAGE NAME property. Note that if you define an Image display property at the option item level, then you must also set the UI: SHOW ITEM IMAGES property to be true at the option class level.

Layout Management

You can use UI properties to manage the basic layout of a configurable product. By specifying the numbers of rows and columns each option class occupies, and by specifying the number of columns on the page, you can fine-tune the look-and-feel of your page without touching the underlying JSP page.

In this section, we will add another option class, and then manage the page layout so that the Monitors option class occupies all of the first row and the other two option classes occupy the second row.

First, create new option classes as follows:

TABLE 44. Processors Option Class

Option Item Name	Description
Pentium 4 2 GHz	Pentium 4 2.8 GHz
Pentium 4 2A GHz	Pentium 4 2.8A GHz
Pentium 4 2C GHz	Pentium 4 2.8C GHz

TABLE 45. RAM Option Class

Option Item Name	Description
SDRAM 256MB	SDRAM 256MB
DDR 256MB	DDR 256MB
RDRAM 256MB	RDRAM 256MB

TABLE 46. Keyboards Option Class

Option Item Name	Description
Logitech 967300	Logitech 967300
Gyration GP170	Gyration GP170
Adesso 595	Adesso 595

Now manage the layout of the configurable product as follows:

1. Navigate to the MXWS-7700 model.
2. On the Display tab, set the Number of Columns property to 3. This is equivalent to setting the UI: NUMBER OF COLUMNS property to 3.
3. Navigate to the Monitors option class.
4. On the Display tab, set the Number of Columns property to 3. This is equivalent to setting the UI: NUMBER OF COLUMNS property to 3.
5. Navigate to the Graphics Cards option class.
6. On the Display tab, set the Number of Columns property to 1.
7. Navigate to the MonitorWeight option class.
8. On the Display tab, set the Option Class Display to Invisible.
9. Navigate to the Processors option class.
10. On the Display tab, set the Number of Columns property to 1.
11. On the Properties tab, enter Processors in the value field of the `_description` property.
12. Navigate to the RAM option class.
13. On the Display tab, set the Number of Columns property to 1.
14. On the Properties tab, enter RAM in the value field of the `_description` property.
15. Navigate to the Keyboards option class.
16. On the Display tab, set the Number of Columns property to 1.
17. On the Properties tab, enter Keyboards in the value field of the `_description` property.

When you compile and test this model, and then select a monitor, then the page is laid out as illustrated below.

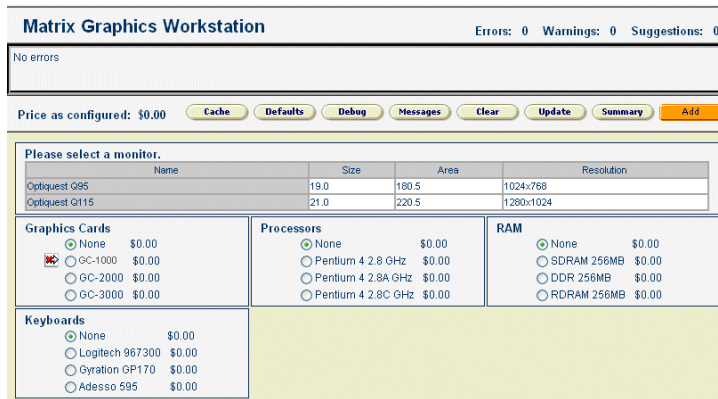


FIGURE 119. Three-Column Layout

Suppose instead that you want the Graphics Cards option class to be displayed in two rows and one column, and then have the Processors class take up two columns. Then do this:

18. Navigate to the Graphics Cards option class.
19. On the Display tab, set the Number of Rows property to 2. This is equivalent to setting the UI: ROW SPAN property to 2.
20. Navigate to the Processors option class.
21. On the Display tab, set the Number of Columns property to 2.

When you compile and test this model, and then select a monitor, then the page is laid out as illustrated below.

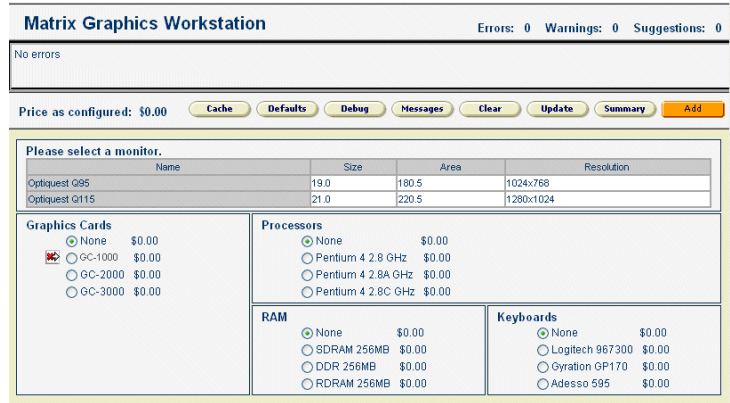


FIGURE 120. Revised Three-Column Layout

Finally, add one more option class. This is a user-entered value class called Case with one option item called Color. Set up the option item as a String-valued property that can take a value from this list: Black,Blue,Green,Red,White.

When you compile and test this model you see that the layout is skewed by the Case option class sticking out on the third row. To correct this, you need to specify that the RAM option class must skip a column: this accounts for the fact that the Graphics Cards option class takes up two rows.

22. Navigate to the RAM option class.
23. On the Display tab, set the Number of Columns to Skip property to 1. This is equivalent to setting the UI: SKIP COLUMNS property to 1.

When you compile and test the model, you now see that the rows and columns are again what you expect.

MXWS-7700 Errors: 0 Warnings: 0 Suggestions: 0

No errors

Price as configured: \$899.00 Lead Time: 0.0 day(s) [Set Defaults](#) [Show Trace Log](#) [Show Messages](#) [Clear](#) [Update](#) [Summary](#) [Add to List](#)

Monitors

None \$0.00
 Optiquest Q95 \$0.00
 Optiquest Q115 \$0.00

[Update](#)

<p>Graphics Cards</p> <p><input checked="" type="radio"/> None \$0.00 <input type="radio"/> GC-1000 \$0.00 <input type="radio"/> GC-2000 \$0.00 <input type="radio"/> GC-3000 \$0.00</p>	<p>Processors</p> <p><input checked="" type="radio"/> None \$0.00 <input type="radio"/> Pentium 4 2.8 GHz \$0.00 <input type="radio"/> Pentium 4 2.8A GHz \$0.00 <input type="radio"/> Pentium 4 2.8C GHz \$0.00</p>
	<p>RAM</p> <p><input checked="" type="radio"/> None \$0.00 <input type="radio"/> SDRAM 256MB \$0.00 <input type="radio"/> DDR 256MB \$0.00 <input type="radio"/> RDRAM 256MB \$0.00</p>
	<p>Keyboards</p> <p><input checked="" type="radio"/> None \$0.00 <input type="radio"/> Logitech 967300 \$0.00 <input type="radio"/> Gyration GP170 \$0.00 <input type="radio"/> Adesso 595 \$0.00</p>

Case
Please enter a color:

FIGURE 121. Revised Three-Column Layout with Skipped Column

Index

Symbols

\$
notation in models 504

Numerics

503 response 184

A

AbstractCronJob class 352
access
defined 411
access control lists 293, 412
access entitlements 272
access policies 293, 296
conditions 297
resource 297
access policy
inheritance 296
access services 298
AccessChecker element 298
accessor methods
effect of Writable attribute 276
AccessPolicy element 296, 297
AccessPolicy.xml configuration file 297
AccessServiceDefinition element 298
actions types Assignment Action
defined 538

actions types Expansion Action
defined 538
actions types Message Action defined 538
activated attribute 209
ACTIVE_FLAG column 276
use to mark objects as deleted 275
addChild method 288
adding a role to a user type 295
adjustFileName method 234, 242, 244
admin user
change password 138
Administrative domain
application server 364
database server 364
DMZ 363
entities 363
external network 363
internal network 363
network zones 363
networks 363
roles 364
servers 363
Web server 364
Alias element 180
Allowed Decimal Places for displaying
extended prices property 587

Allowed Decimal Places for displaying list
 prices property 588
Allowed Values display property 638
Alternate element 285
Analyzing debs.log 394
ancestors
 of storefronts 407
any value of Level attribute 174
Apache 119
 keepalive 123
 serving static content 123
 using to compress output 125
Apache web server
 expires module 123
app.name property 78
AppContextCache class 243
AppExecutionEnv class 233, 241
application beans 238, 273
application cron jobs 585
ApplicationCron class 350, 352
apps.dir property 75
apps.name property 75
AppsLookupHelper class 242
Assignment Action
 creating 542
Assignment Action defined 538
assignment action properties 543
attached properties
 reporting on 558
attached rules
 reporting on 558
attaching properties 494
attributes
 DataService 285
 DataSourceName 284
 Encryption 178
 ExternalFieldName 282
 ID 239
 IsOverlay 233
 MaxPoolSize 241
 Name 232, 282
 state 134
 Version 282, 290
audit trail 309
AuthenticationAPI 107
Automatic Post display property 572
Automating Log Analyzer reports 397
Availability Data Access Method
 property 587

B

backing up the Sterling System 145
Backup and recovery strategies 380
Backups
 checkpoint backup 380
 daily backup 380
 full 380
 incremental backup 380
 weekly backups 380
binding to a port 142
BizAPI classes 39
bizAPI classes 320
Bizlet class 232
BizletMapping
 default value for message group 234
BizletMapping element 232
BizRouter class 232
BLC abstract class 320
BooleanExpression element 298, 299
browser support 52
Build All Segments cron job 595
bundle attribute 330
business logic classes 39, 268, 319
 implementation 265, 319
business objects
 lists 272
 User 235
business rules
 managing tasks 589
 overview 416
BusinessObject class 290

C

C3_Commerce_Manager element 114
C3PrimaryRW data object 268
Cache Cleanup (cron job) 595
calendar 338
calendar widget
 localizing 338
callJSP method 245
canRequest method 294
carts
 moving a user 439
cascading style sheets 339
certificate authorities 177
certificates
 use in the SSL protocol 177
character set 54
character sets 152, 327

characters
 invalid in models 443
 Check method 246
 checkPolicy method 304
 child data objects 279
 child partners
 moving 435
 ChildDataObject element 279
 children method 288
 cipher-update script 188
 Class attribute 181
 classes 236
 AbstractCronJob 352
 AppExecutionEnv 233, 241
 ApplicationCron 350, 352
 Bizlet 232
 BizobjBean 272
 BizRouter 232
 BusinessObject 290
 ComerentSession 235
 ComergentAppEnv 235, 242
 ComergentContext 234
 ComergentDispatcher 234
 ComergentException 343
 ComergentRequest 234
 ComergentResponse 234
 ComergentRuntimeException 344
 Controller 39
 ConverterFactory 248
 CronConfigBean 350
 DataBean 237, 238
 DataContext 268
 DataManager 284, 287
 DataMap 288
 DataService 285
 DebsDispatchServlet 236
 DispatchServlet 231, 235
 Driver 141
 DsElement 288
 Env 234
 Exception 343
 GeneralObjectFactory 236
 HttpRequest 234
 HttpResponse 234
 HttpServletRequest 352
 HttpSession 235
 ICCEException 343
 InitServlet 231, 235, 243
 MessagingController 236, 237
 MetaData 289
 NamingManager 321
 NamingResult 322
 NamingServiceDatabase 321
 NamingServiceProperties 321
 ObjectManager 238, 265, 268
 OMWrapper 238, 265
 OracleDriver 141
 RequestDispatcher 234
 ResourceBundle 339
 RuntimeException 344
 SimpleController 237
 SystemCron 349, 352
 User 294
 classifications
 rules 514
 ClassName element 238, 239
 classpath 141
 Clear button in UI controls 581
 cloneDsElement method 288
 Closed
 user status 414
 cluster node polling interval 212
 cluster setup
 servlet for serving static content 207
 clustered deployment 115
 clustered environment 243
 clustered implementation 42
 clustering 115
 clustering setup 197
 clustering support 52
 clusters
 running cron jobs 586
 CMGT_ANALYZER_TEXT table 137
 CMGT_CURRENCIES table 137
 CMGT_LOCALE table 137
 CMGT_LOCALE_CURRENCY table 137
 CMGT_LOCALE_NAMES table 137
 CMGT_LOOKUPS table 137, 242
 CMGT_SYS_PROPERTIES table 415
 cmgt-logging.jar JAR file 147
 cmgtText method 329
 code examples
 using locale properties files 334
 com.comergent.api.dataservices
 package 255
 com.comergent.api.dcm.cryptography.Digest
 ster interface 182

com.comergent.api.dcm.cryptography.SymmetricEncrypter interface 181
 com.comergent.api.dispatchAuthorization package 261
 com.comergent.api.msgservice package 264
 com.comergent.dcm.caf.controller.Controller class 236
 com.comergent.dcm.core.filters package 354
 com.comergent.dcm.objmgr package 241
 com.comergent.dispatchAuthorization package 261
 com.comergent.msgservice package 264
 com.comergent.reference.jsp package 330
 comergent.preferences.store system property 84, 202, 204
 Comergent.xml configuration file 232
 setting SSL port 175
 ComergentAppEnv class 235, 242
 ComergentContext class 234
 ComergentDispatcher class 234
 ComergentHelpBroker class 255
 ComergentI18N class 333
 ComergentRequest class 234
 ComergentResponse class 234
 ComergentSession class 235
 command
 instanceof 273
 Commerce tab 428
 Company Web site address required for Partner Selector 425
 ComparativeExpression element 299
 compile all models 487
 compiled stylesheets 247
 compileStyleSheets system property 247
 compiling models 486
 compressed output 125
 conditions
 access policies 297
 CONFIG
 REPEAT FIRING property 523
 SUBMODEL NAME property 484
 SUBMODEL RETURN property 484
 ConfigCompiler
 entry in ObjectMap.xml 83
 configuration files 43, 118, 222
 Comergent.xml 192, 195, 231, 232
 DataServices.xml 108, 135, 141, 200
 DataSources.xml 133, 141, 195
 defined 415
 DsBusinessObjects.xml 282
 DsConstraints.xml 357
 DsRecipes.xml 282
 Entitlements.xml 583
 Internationalization.xml 134, 153, 328
 KeyGenerators.xml 142
 MessageMap.xml 248
 MessageTypes.xml 231, 236
 MsSqlDataSources.xml 135
 MsSqlKeyGenerators.xml 133, 142
 ObjectMap.xml 238
 OracleDataSources.xml 135
 OracleKeyGenerators.xml 133, 135, 142
 web.xml 113, 191, 222, 223, 231
 configuring global application cache 115
 connection pooling 109
 ConnectionTimeout element 112
 ConnectTimeout element 111
 Constant Guiding Text display
 property 572
 constrained data field 357
 Constraint Selections display property 574
 constraints table
 creating 545
 deleting 552
 modifying 546
 container.home property 75, 76
 content type 236
 context
 setting attributes 234
 Control 572
 Control display property 572
 Controller classes 236
 as part of reference implementation 312
 ControllerMapping
 default value for message group 234
 ControllerMapping element 232
 ConverterFactory class 264
 copyBean method 275
 copying a rule 519
 copying model groups 451
 countries
 standard abbreviations 153
 country codes 117, 118

createController method 236
 createDB target 80, 130
 creating a message action 538
 creating a model 453
 creating a worksheet 502
 creating an Assignment Action 542
 creating an Expansion Action 539
 creating cron jobs 593
 creating profile addresses 435
 creating profiles 432
 cron job
 Nightly Segments Build 91
 cron jobs 349, 591
 application 585
 creating 593
 deleting 595
 displaying 592
 message URL 586
 modifying 594
 predefined jobs 595
 running immediately 594
 running in a cluster 586
 running in cluster environments 592
 system 585
 system settings 585
 viewing 592
 viewing history 595
 CronConfig data object 586
 CronConfigBean class 350
 CronJob interface 350
 CronManager class 350
 cronRefreshTime Comergent.xml
 entry 212
 CronScheduler class 350
 Cross-Site Request Forgery Filter 189
 Cryptographically secure digests 367
 CryptographyService.xml configuration
 file 180, 181
 currencies 327, 337
 used in testing models 486
 currency
 used in pricing 246
 current locale 153
 custom tag libraries 223
 customer types
 used in pricing 246
 used in testing models 486
 customer users 405
 customize target 323

D

Data
 protecting 379
 timeline for recovery 380
 data fields metadata 289
 data objects 38, 268
 accessing child data objects 279
 customizing 268
 extending 239, 267
 ordinality 267
 stored procedures 274
 database
 indexes 129
 password 48
 username 48
 Database management 385
 performance 386
 practices 385
 database password
 encrypted 77
 database searches 108
 database server 141
 database servers 133
 Database tables
 monitoring size 386
 database userid 54
 databases
 client software 52
 DataBean class 237, 238
 DataContext class 268, 275
 use in restore 273
 DataField element 282
 DataManager
 initialization error 141
 DataManager class 132
 DataObject attribute 297
 DataObject element 284
 DataObjects 142
 DataService attribute 285
 DataService class 285
 DataServices element 136
 DataServices.General.LimitDBResults
 preference 271
 DataServices.General.ServerId system
 property
 use in clustering 205
 DataServices.xml configuration file 108,
 112, 135, 154, 200
 DataSourceName attribute 284

DataSources.xml configuration file 133
 dates 337
 DBCache class
 used for session caching 210
 dcmsKey.ser file 132, 200
 DebsDispatchServlet class 236
 debug method 308
 debugging JSP resource bundles 332
 debugJSPResourceBundle element 332
 default locale
 failover mechanism 333
 Default Selection 573
 Default Selection display property 572
 default values for properties 494
 default XML user 103
 defaultCountry element 153, 333
 defaultSystemLocale element 134, 153,
 328, 329, 333
 defaultType element 322
 define and attach properties 618
 defining display properties 571
 defining lists 507
 defining option constraints 547, 548
 delete method 275, 288, 290
 deleteChild method 289
 deleting a cron job 595
 deleting children of models 455
 deleting fragments 537
 deleting lists 509
 deleting model groups 450
 deleting models 455
 deleting option constraints 552
 deleting option constraints table 552
 deleting users 421
 deploy.environment property 78
 Deployment architecture 378
 build environment 378
 QA area 378
 staging area 378
 deployment files
 Sterling.war 230
 descendants
 of storefronts 407
 Digester element 181
 disableAccessCheck method 277
 disallow results from triggers 55
 DispatchServlet class 235
 initialization 184
 display properties 562, 631
 Automatic Post 572
 Constant Guiding Text 572
 Control 572
 Default Selection 572
 defining values 571
 Display Template 572
 Help URL 572
 Icon Graphics 572
 Ignore In Quote 573
 Lead Time 573
 Option Class Required 573
 Option Class View 574
 Popup-Qty Values 574
 Post-Pick Guiding Text 574
 Pre-Pick Guiding Text 574
 Price 575
 Pricing Style 575
 Return From Submodel 575
 User Entered Value Allowed
 Values 576
 User Entered Value Postfix 576
 User Entered Value Prefix 576
 User Entered Value Type 575
 Validate Submodel 576
 display settings
 reporting on 558
 Display Template 573
 Display Template display property 572
 dist target 78, 192
 distributed installation
 see clustering 115
 distributor 424
 distWar target 78, 204
 doFilter method 353
 DosFilter class 354
 DsDataElements.xml configuration file
 setting the lengths of data fields 334
 DsDataSources element 136
 DsElement
 child 287
 parent 287
 root 287
 DsElement tree 287
 legacy applications only 286
 DsElements 286
 DsKeyGenerators element 133
 DsQuery class 277
 use in restore 273
 dXML message family 247

dynamic instantiation 555

E

effective locales 585

effective status

user 414

effectivity dates

used in testing models 486

Ehcache 115, 199

Ehcache.xml file 115

elements

Alternate 285

BizletMapping 232

C3_Commerce_Manager 114

ControllerMapping 232

DataElements 284

re-use 284

DataField 282, 284

DataObject 284

DataServices 136

defaultCountry 153

defaultSystemLocale 134, 153, 328

DsDataSources 136

DsKeyGenerators 133

ExternalName 274

GeneralObjectFactory 232

globalCacheImplClass 243

JdbcDriver 135

JdbcDriver1 135

JSPMapping 232

KeyGenerator 142

Languages 153

LowerCase 108

memoryThreshold 114

MessageType 232

messageTypeFilename 232

MessageTypeRef 143

messageTypeValidate 140, 193

Microsoft 108

Primary 285

propertiesFile 141, 231

ServerId 200

session-timeout 113

SMTPHost 139

UpperCase 108

UseLocalizedSort 154

web-app 114

WebPathToPublicLoadableWritableDirectory 116

WebPathToPublicNoLoadableWritableDirectory 116

email 139

email addresses in properties files 76

email clients

problems displaying UTF-8

characters 139

email templates 335

location 336

encrypting data in the Knowledgebase 132

encrypting fields used in reports 178

encrypting the database password 77

Encryption

algorithms 366

DES 168 369, 370

encryption keys 367

Key Store 367

JCE Key Store 366

password protection 366

two phase initialization

support 366

libraries 366

MD5 message digest 367, 370

mechanisms 369

persistent data 366

secret key 367

SHA-1 message digest 367, 370

WAR local key store file 367

enterprise system administration 416

enterprise users 405

entitlement roles

defined 411

Entitlements.xml 411

Entitlement services 364

access policy service 364

data-level access 364

dispatch service 364

page flow privileges 364

EntitlementFactory class 261

entitlements 293

Entitlements.xml 412, 422

assigning access roles 411

Entitlements.xml configuration file 294

entity beans 273

entries

NamingManager 195

SSLListener 177

entry point 305

EntryPoint attribute 305

Env class 234
environment variables
 JAVA_HOME 74
 JDK_HOME 63
erase method 275
ERPAdmin user
 change passwords 138
ERPAdministrator user type 420
error method 308
evaluating property values 504
exception handling 343
Exceptions 343
 displaying 347
expand
 use in string property window 498
expand function 635
 syntax 544
expand groups
 reporting on 558
Expansion Action
 creating 539
Expansion Action defined 538
expires_module module 123
exporting a worksheet 503
exporting model groups 554
exporting models 554
Expression element 298
extended prices
 displaying 587
Extends attribute 267
external compiler 82
ExternalFieldName attribute 282
ExternalName element 274

F

Factory pattern 238
failover behavior 332
failover mechanism for JSP pages 333
failover mechanism for resource
 bundles 332
fallback redirect message type 305
FallbackRedirect element 305
fastdeploy target 87
fatal method 308
filters
 J2EE filters 353
findPresentationLocale method 334
fonts 339
foreach 526

fragments
 deleting 537
 modifying 536
 nested fragments, creating 530
 simple fragments, creating 527
frameset
 using OnlineOrderingPageDisplay
 message type 586
frameset system property 586
frequency (cron jobs) 594
Function drop-down list
 used in Numeric Property Editor
 window 497
function labels 411
functions 294, 411, 497
 defining 422
 LOWER 108
 UPPER 108

G

gather
 use in string property window 498
GeneralObjectFactory class 236
GeneralObjectFactory element 232
generateBean target 192, 237, 268, 273,
 285, 315
generated interfaces
 use in application beans 274
generateDTD target 117, 192, 268
generateKeys method 275
get method 322
getAllowedValueIterator method 289
getAssignedPriceListKey method 247
getBizObj method 278
getBoolean method 245
getCacheId method 270
getComergentLocale method 333
getCountAllowedValues method 289
getDataBean method 274
getDataType method 289
getDefaultLocale method 333
getDefaultValue method 289
getDouble method 245
getElementByName method 288
getFloat method 245
getInContextPricePriceListKey
 method 247
getInstance method 321
getInt method 245, 262

getIRdProduct method 274
 getLong method 245
 getMaxCharLength method 289
 getMaxLength method 289
 getMaxPaginatedResult 270
 getMaxResults method 270
 getMaxValue method 289
 getMetaData method 289
 getMinValue method 289
 getName method 289
 getNumPerPage method 270
 getObject method 238
 getParameter method 352
 getParameters method 352
 getParent method 289
 getPreferences method 245
 getRealPath method 244
 getResourceAsStream method 234
 getRootElement method 287, 288, 290
 getSession method
 ComergentSession class 235
 getString method 245
 getType method 289, 291
 global caching and JavaSpaces 210
 Global class
 deprecated use for logging 307
 replaced by Preferences 243
 GlobalCache interface 243
 groups
 attached groups, viewing 479
 modifying 470
 product IDs, assigning 470
 start and end dates for option
 items 470

H

hasError method 304
 Help URL 573
 Help URL display property 572
 History tables 386
 _H naming convention 387
 creating 388
 transferring data to 388
 HTML templates 117
 HttpRequest class 234
 HttpResponse class 234
 HttpServletRequest class 352
 HttpSession class 235

I

IAcc interface 276
 Icon Graphic 573
 Icon Graphic field 579
 Icon Graphics display property 572
 ID attribute 239
 id attribute
 used in text tag 330
 IData interface 275, 276
 accessing metadata 289
 Ignore In Quote display property 573
 images 117
 flickering 123
 used in models 579
 IMetaData interface 289
 importing a worksheet 504
 importing model groups 553
 importing models 553
 including sub-models in models 483
 Incremental garbage collection
 -Xincgc 394
 info method 308
 Infrastructure
 application tier 379
 database tier 379
 typical 378
 Web tier 378
 initialization 141
 InitManager class 256
 InitServlet class 235
 initialization of cryptography
 service 184
 InMemoryRuleCompiler
 entry in ObjectMap.xml 83
 install target 76
 installation directory
 container_home 47
 installMsSql target 108
 installMSSQLJDBC target 77
 installOracle target 76
 instanceof command 273
 interfaces
 Converter 247
 GlobalCache 243
 IAcc 276
 IData 275
 Ird 276
 NamingService 321
 poolable 240

internal compiler 82
 internationalization 134
 cascading style sheets 339
 failover mechanism for JSP pages 333
 failover mechanism for resource bundles 332
 Internationalization.xml configuration file 134, 153, 328, 332, 333
 IPasswordPolicy interface 304
 iPlanet
 troubleshooting 145, 195
 IPolicyClass interface 303, 304
 IRd interface 276
 ISO standards 152
 ISO-3166 country codes 584
 ISO-639 language codes 584
 IsOverlay attribute 233
 isPersistable method 276
 isRequestDenied method 354
 IsRestorable method 276

J

J2EE 37, 221
 Java 2 Platform, Enterprise Edition 37, 221
 Java Cryptography Extension 178
 Java Servlet Specification 47
 support for 2.2 or 2.3 61
 JAVA_HOME environment variable 64, 74
 JavaSpaces
 used for global caching 210
 JCE 178
 JCEDigester class 185
 JCESymmetricEncrypter class 185
 JDBC 52
 JDBC drivers 141
 JdbcDriver element 135
 JdbcDriver1 element 135
 JDK 1.3.1 51
 JDK versions 51
 JDK_HOME environment variable 63, 64
 Jini Lookup server
 use in global caching 210
 JK connector 119
 job scheduling
 predefined jobs 595
 job scheduling tasks 592
 JoinKey element 279
 JSP pages 38, 221
 as part of reference implementation 312
 debugging localization 332
 forcing their recompilation 140
 localization 337
 page buffer 348
 used in email templates 244
 JSPMapping
 default value for message group 234
 JSPMapping element 232, 333
 JVM
 -server 393
 -verbose gc 393
 -Xincgc 393

K

keepalive settings 123
 keepgenerated parameter 89
 Key attribute 305
 Key performance indicators 394
 key rotation 185
 key store 181
 Key Store file 367
 key stores 184
 initialization 184
 KeyGenerators.xml configuration file 142
 KeyManager attribute 181
 KeyManager element 181, 188
 KeyName attribute 181
 KeyStorePath attribute 188
 keystores
 JDK versions 51
 Knowledgebase 48, 54, 127, 128, 350
 schema creation 128

L

labels
 of functions 411
 language codes 117, 118
 languages 152, 327
 standard abbreviations 152
 Languages element 153
 LDAP 45
 Lead Time display property 573
 LegacyFileUtils class 234, 244
 LegacyPreferences class 243
 length of data fields 334
 Level attribute 174

Lines Per Page in List Displays
 property 588
 link method 174
 Linux, Red Hat 50
 List as property type 493
 list business objects 272
 list definitions
 reporting on 558
 list prices
 displaying 588
 lists 506
 defining 507
 deleting 509
 modifying 508
 load-balancing
 session-sticky 200
 loadDB target 80, 81, 131, 136
 loadMatrixDB target 80, 81, 131, 136
 locales
 case-sensitive searching 108
 database support 128
 directories 117, 118
 display names 153
 loading locale-specific data 134
 preferred locale 328
 presentation 329
 removing from implementation 137
 session 329
 setting display names 584
 sorting 153
 standard abbreviations 153
 support for different 134
 system default 153
 localization 327
 images 336
 Javascript 337
 localRedirect method 234
 Location drop-down list
 used in Numeric Property Editor
 window 498
 Log Analyzer tool 394
 directory structure 398
 properties file 400
 Log configuration in clustered
 environments 149
 log files
 standard locations 150
 log files get too large 149
 log method 308
 log4j API 307
 log4j.debug system property 308
 log4j.properties configuration file 307
 logging 147
 logging levels 148
 logging methods
 debug 308
 error 308
 info 308
 log 308
 warning 308
 Logging settings, configuring 599
 login form
 passing request parameters 305
 logLevel methods 307
 logout method 235
 lookup codes 241, 248, 249
 mapping to strings 242
 lookup types 242, 248
 LowerCase element 108

M

Maintain Configuration 596
 managers 414
 managing business rules 589
 MandatoryRoleSet element 295
 match
 use in string property window 498
 Max Reps Per Account field 441
 MaxConnections property 111, 112
 maximum number of users
 assigned to an account 433
 MaxPermSize setting 114
 MaxPoolSize attribute 112, 241
 MaxPoolSize property 111
 MaxResults element 269
 Memory allocation
 areas to check 394
 memory allocation 114
 memoryThreshold element 114
 merge target 78, 192
 message action
 creating 538
 Message Action defined 538
 message category 247
 message family 247
 message group
 fallback redirect message type 305
 message groups 232

- assigning to new role 295
 - entry points 305
 - used to specify default mappings 234
- message types 232, 247, 293
 - accessing through SSL 175
 - validating on startup 140, 193
- Message URL 195
- message URL for cron jobs 586
- message version 247
- messages 319
- MessageType element 232
 - child elements 232
- messageTypeFilename element 232, 233
- MessageTypeRef element 143, 233
- MessageTypes.xml configuration file 231
- messageTypeValidate element 140, 193
- MessagingController 236
- MessagingController class 236, 237
- MessagingServlet class 231
- metadata
 - for data fields 289
- MethodNotFound exceptions 140
- methods
 - addChild 288
 - adjustFileName 242
 - calculate 237
 - canRequest 294
 - children 288
 - cloneDsElement 288
 - constructExternalURL 242
 - copyBean 275
 - createController 235, 236
 - delete 275, 288, 290
 - deleteChild 289
 - dispatch 235
 - erase 275
 - forward 234
 - generateKeys 275
 - get 322
 - getContext 243
 - getConverter 248
 - getDataBean 274
 - getElementByName 288
 - getEnv 242
 - getInstance 321
 - getName 289
 - getObject 238
 - getParameter 352
 - getParameters 352
 - getParent 289
 - getPartnerKey 235
 - getRootElement 287, 288, 290
 - getType 289, 291
 - getUser 235
 - getUserKey 235
 - include 234
 - init 236, 243
 - isPersistable 276
 - IsRestorable 276
 - link 174
 - persist 238, 275, 276, 278, 285, 290, 320
 - prune 275
 - reset 241
 - restore 238, 275, 276, 277, 285, 289, 320
 - return 241
 - runAppJob 233
 - runAppObj 241
 - service 320, 352
 - setCacheId 269
 - setDataContext 275
 - setRetry 352
 - setRootElement 291
 - update 275
- methods setExecutionOutcome 352
- Microsoft element 108
- Microsoft SQL Server 129
- mod_expires
 - use to prevent image flicker 123
- mod_jk 120
- model groups 448
 - copying 451
 - copying models into 456
 - creating 449
 - deleting children 450
 - modifying 449
- model images 579
- models
 - assigning a product 456
 - attaching to an option class 477
 - compiling 486
 - compiling all 487
 - copying 456
 - copying a model reference 458
 - copying option classes into 464
 - creating 453
 - deleting 455

- deleting children 455
- effectivity dates 453
- embedding 460
- end date 453
- invalid characters 443
- modifying 454
- setting prices 575
- start date 453
- tabbed user interface 488
- testing 485
- modifying a cron job 594
- modifying cron jobs 594
- modifying models 454
- modifying option constraints 551
- modifying properties 498
- modifying property definitions 499
- modifying rule fragments 536
- moving a rule 519
- moving child partners 435
- MsgContext interface 264
- MsgService interface, 264
- MsgServiceException class 264
- MsgServiceFactory class 264
- MsSqlDataSources.xml configuration file 135
- MsSqlJNI.dll file 135
- MsSqlKeyGenerators.xml configuration file 133, 142
- multi-byte characters 334
- multiple-pass rule firing 523
- my_sdk.properties file 75

N

- Name attribute 180, 181, 232, 282
- names
 - of profiles 424
- naming service 321
- NamingManager class 321
- NamingResult class 322
- NamingServiceDatabase 321
- NamingServiceDatabase class 321
- NamingServiceProperties class 321
- nested fragments in rules 530
- newproject target 76, 323
- Not element 299
- number and date formats 152, 327
- Number as property type 493
- Numeric Property Editor window 496
- NumPerCachePage element 269

O

- Object element 238, 239
- object pools 240
- ObjectManager class 238, 265, 268
- ObjectMap.xml configuration file 83, 238
- OCI driver 56
- ODBC connection 52
- OMWrapper class 238, 265
- On Credit Hold
 - user status 414
- On Hold
 - user status 414
- OneWayEncrypter element 181
- online help 117
- Online users 364
- OnlineOrderingPageDisplay message type used in frameset 586
- Open
 - user status 414
- Open Web Application Security Project (OWASP) 362
- Operator attribute 298, 299
- option class groups
 - attached, viewing 479
 - attaching to a model or group 476
 - attaching to an option class 477
 - copying option classes into 464
 - creating 469
 - deleting 482
 - removing an attachment to a group 483
 - removing an attachment to a model 455
 - removing an attachment to an option class 468
- Option Class Required display property 573
- Option Class Viewdisplay property 574
- option classes
 - adding option items to 464
 - assigning a product 456
 - copying 464
 - copying option items into 467
 - creating 461
 - defining option constraints 548
 - deleting 468
 - display properties, setting 463
 - embedding models into 460
 - product IDs, assigning 463

- properties, attaching 463
- ratio, setting 463
- rules, attaching 463
- setting prices 575
- option constraints
 - defining 547
 - deleting 552
 - modifying 551
- option constraints table
 - creating 545
 - deleting 552
 - modifying 546
- option item groups
 - attached, viewing 479
 - attaching to a groups 476
 - attaching to an option class 477
 - copying option items into 467
 - creating 469
 - deleting 482
 - removing an attachment to a group 483
 - removing an attachment to an option class 468
- option items
 - adding to option classes 464
 - assigning a product 456
 - copying 467
 - display properties, setting 466
 - in option constraints 550
 - product IDs, assigning 466
 - rules, attaching 466
 - setting prices 575
- Oracle
 - OCI driver 107
 - Oracle Server 129
 - OracleDataSources.xml configuration
 - file 135
 - OracleKeyGenerators.xml configuration
 - file 133, 135, 142
- orders
 - moving users 439
- org.apache.log4j.Level class 259
- organizational functions 410
- OriginalName attribute 180
- OutOfBandHelper class 244
- P**
- packages
 - com.comergent.dcm.objmgr 241
- pagination
 - specifying number of items on each page 588
- pagination settings 112
- ParamList element 303
- partner addresses
 - deleting 436
- partner key
 - used in pricing 246
- partner profiles 195, 440
- Partner Selector
 - Company Web site address required field 425
- partner user
 - assigning attributes 437
- partner users
 - channel administrator 411
 - creating by enterprise administrator 437
- Partner.com Partners
 - Commerce tab 428
- partners
 - contract 428
 - deleting addresses 436
 - modifying 440
 - territory 428
 - used in testing models 486
- PartnerTypeDefinition element 295
- password 54
- Password data field 586
- password policies 189, 302
- password policy types 303
- PasswordPolicies.xml configuration
 - file 302, 303
- PasswordPolicy element 302
- passwords 411, 420, 437
 - changing password of admin and ERPAdmin users 138
 - policies 302
- PATH environment variable 64
- payment accounts 426
- payment processors 428
- Performance
 - memory issues 391
 - out of memory error 394
- Performance tuning
 - garbage collection 393
- persist method 238, 275, 276, 278, 285, 290, 320

call after delete method 275
 PolicyCheckResult class 304
 PolicyClass element 303
 poolable interface 240
 pooling objects 240
 Popup-Qty Values display property 574
 Post-Pick Guiding Text display
 property 574
 precision
 displaying extended prices 587
 displaying list prices 588
 pre-compiling JSP pages 89
 predictive access control 296
 Preferences API 245
 preferences for users 415
 preferred locale 153, 420
 prefs.xml configuration file 83
 Pre-Pick Guiding Text display
 property 574
 presentation beans 273
 presentation locale 329
 Prevent Selection of Items Resulting in
 Constraint Errors 574
 price and availability requests 195
 Price display property 575
 PriceCheckAPI class 246
 prices
 setting for a model, option class, or
 option item 575
 pricing
 Check method 246
 getting prices for products 246
 Pricing Style display property 575
 PricingLineItem class 246
 Primary element 285
 PrincipalQualifier attribute 298
 PrincipalQualifier interface 297
 PrincipalQualifierDefinition element 297
 principals
 access policies 297
 priorities
 for rule firing 520
 production
 moving compiled models 487
 products
 assigning to a model, option class,
 option item 456
 profile addresses
 creating 435
 profile levels 425
 profile names 424
 uniqueness 424
 profile types 424
 selection for distributors 425
 profiles
 creating 432
 creating addresses 435
 level 425
 type 424
 Program Management function 589
 properties 492, 561
 attaching 494
 default values 494
 defining 493
 modifying a definition 499
 modifying an attached value 498
 removing an attachment 498
 properties, attaching 466
 propertiesFile element 141
 property definitions
 reporting on 558
 Property drop-down list
 used in Numeric Property Editor
 window 497
 property editor window 496
 property types
 List 493
 Number 493
 String 493
 property values 497
 evaluating 504
 propval function 628
 protocols
 https 177
 prune method 275
 putInt method 262
 putString method 245

Q

quote characters in names 541
 quotes
 moving a user 439

R

ratio, setting a 463
 Recipe element
 declaring ordinality 272
 recipes 268

ReconnectOnTimeout element 111
 Recovery policies 381
 Recovery scenario 380
 Redirect element 305
 redirecting a request 235
 Redundancy 378
 Regulatory guidelines
 AICPA 362
 ISO 1799 362
 OWASP 362
 PCI 362
 Sarbanes-Oxley 362
 SAS 70 362
 Relationship element 279
 relative
 property locations 498
 reloadFilePeriod attribute 209
 removing locales 137
 reports
 Visual Modeler 558
 request dispatcher 223
 RequestDispatcher class 234
 requests 319
 requireHttps value of Level attribute 174
 requirements 42
 database server 54
 hardware 49
 network 52
 software 50
 reset method 241
 resource bundles 331
 ResourceClass element 296
 resources
 controlling access 297
 restore method 238, 275, 277, 285, 289,
 320
 example using DataContext and
 DsQuery 277
 stored procedures 274
 use in list beans 273
 restoring the Sterling System 145
 Return From Submodel display
 property 575
 return method 241
 Return to General 477, 478
 RFC 1918 compliance 363
 Roles
 Data Center administrative roles 365
 Responsibilities
 Database administrator 364
 Developer 365
 Network administrator 365
 System administrator 364
 roles 43, 294
 defining 422
 entitlement 411
 partner administrator 437
 RosettaNet 118
 rsCachePath element 112
 rsCachePathIsAbsolute element 112
 rule classifications 514
 rule definitions
 reporting on 558
 rule firing 515
 controlling 522
 testing 521, 523
 rule firing sequence 520
 RuleCompiler
 entry in ObjectMap.xml 83
 rules 82
 copying 519
 moving 519
 rules (Visual Modeler)
 firing sequence 515
 rule actions 537
 rules in Visual Modeler
 attached rules, viewing 516
 attaching 514
 defining 510
 deleting a rule 519
 modifying 513
 unattaching 518
 runAppJob method 233
 running a cron job immediately 594

S

saveOnRestart attribute
 Tomcat setting 87
 schema
 http and https 174
 schemaRepositoryExtn element 324
 scripting elements 223
 scriptlets 223
 scripts
 oracle_indexes.sql 129
 XMLLoader.bat 132
 SDK 323
 See Software Development Kit

- search index builder
 - run as application cron job 586
- SearchConfigurationProperties.xml file
 - in clustered environment 209
- searches
 - case-sensitive 108
- searching for properties 556
- searching for users 421
- Secure logging 367
- Secure storage
 - account information 367
 - passwords 370
 - user information 368
- Security
 - regulatory guidelines
 - ISO 17799 362
 - OWASP 362
 - PCI 362
 - Sarbanes-Oxley 362
 - SAS 70 362
- security 43, 45
 - changing passwords for admin and ERPAdmin 173
- Security model 361
- SecurityLevel element 174
- segments
 - Matrix reference segments 91
- serializable context attributes 234
- Serializable interface 235
- ServerId element 108, 200
- ServerId property 206
- ServerSSLPort element 175
- service method 264, 320, 352
- servlet container
 - root directory 63
 - support for clusters 52
- servlet containers
 - requirements 51
- servlet context 48
 - setting attributes 234
- session locale 329
- session timeout 113
- sessions 113
- SESSIONS.ser
 - Tomcat session file 87
 - troubleshooting 140
- session-sticky load-balancing 200
- session-timeout element 113
- setAttribute method
 - ComergentSession class 235
- setCacheId method 269, 270
- setDataContext method 275
- setExecutionOutcome method 352
- SetExpression element 299
- setMaxPaginatedResult 270
- setMaxResults method 270
- setNumPerPage method 270
- setRetry method 352
- setRootElement method 291
- SharedPublicServlet class 207
- SimpleController class 237
- single-pass rule firing 522
- site system administration 416
- SMTP Host Machine property 587
- SMTP mail server 139
- SMTPHost element 139
- Software Development Kit 323
 - installing 74
- Solaris 50
- sorting data 153
- SourceType attribute 274
- special characters
 - encoding in Visual Modeler 484
- spreadsheet format
 - partner lists 432
- SQL Server
 - clustering step 205
 - in a clustered environment 200
 - requirements 55
- SQL Server use of Unicode 55
- SSL 173, 176
 - client 176
 - protocol 176
 - server 177
 - setting up Apache 122
- SSL port in Comergent.xml file 175
- staging
 - moving compiled models 487
- standard locations of log files 150
- state attribute 131, 134
- static content
 - servicing up using Web server 123
- status
 - user 414
- STDOUT appender 148
- Sterling Analyzer reports
 - encrypting fields 178
- Sterling Configurator

use of in-memory compiler 83
 Stop Firing (column) 515
 stored procedures 274
 storefront
 pricing for storefronts 246
 Storefront administrator limitations 407
 storefront administrator partner 440
 Storefront administrator tasks 406
 storefront administrator tasks 406
 storefront administrators 440
 storefront enterprise users 405
 Storefront Partners
 Commerce tab 428
 storefronts 405
 creating 440
 Strategies
 backup and recovery 380
 String as property type 493
 string property editor 635
 String Property Editor window 496
 stylesheets
 compiled 247
 sub-models 483
 subsystem 344
 SymmetricEncrypter element 180
 system administration 415
 system administration URL 598
 system administrators 597
 system cron jobs 585
 system default locale 153
 System Logging (log4j dynamic) Page 149
 system properties
 frameset 586
 SystemCron class 349, 352

T

tabbed user interface 488, 573
 table
 used to display properties 576
 TABLESPACE name 96
 tablespaces 129
 tabular display of properties 576
 Tag attribute 180, 181
 tag libraries 223
 tag library descriptor 223, 231
 targets
 createDB 130
 distWar 78
 generateBean 237, 268, 273, 285, 315
 generateDTD 117, 268
 install 76
 installMSSQLJDBC 77
 installOracle 76
 newproject 76
 testing models 485
 text tag 329
 Threat scenarios 370
 TLD. See tag library descriptor
 TNS alias 62
 Tomcat
 disable session persistence 87
 problem with graceful shutdown 87
 toolbar 444
 tracing rule firing 521, 523
 Transaction class 247
 Transient logging configuration
 changes 149
 TwoWayEncrypter element 179, 180
 type attribute 131
 types
 password policies 303

U

UI

ICON GRAPHIC property 579
 ITEM IMAGE NAME property 579
 PRICE property 567
 PRICING SKU property 567
 PRICING STYLE property 567
 ROW SPAN property 568
 SHOW ITEM IMAGES
 property 568, 579
 UI: ICON GRAPHIC property 640
 UI: IGNORE IN QUOTE property 632
 UI: ITEM IMAGE NAME property 640
 UI: NUMBER OF COLUMNS 566
 UI: NUMBER OF COLUMNS
 property 641
 UI: OPTION CLASS VIEW 566
 UI: POPUP-QTY ALLOWED
 VALUES 566
 UI: PRE_PICK GUIDING TEXT
 property 631
 UI: PRODUCT DESCRIPTION
 property 568
 UI: PRODUCT ID property 568
 UI: PRODUCT NAME property 568

- UI: ROW SPAN property 642
- UI: SHOW ITEM IMAGES property 640
- UI: SKIP COLUMNS property 643
- UI: SUPPRESS NAME DISPLAY 569
- UI: SUPPRESS NONE SELECTION 569
- UI: SUPPRESS UEV NONE VALUE 569
- UI: SUPPRESS UEV NONE VALUE property 639
- UI: UEV ALLOWED VALUES 569
- UI: UEV ASSIGNMENT PROPERTY 570
- UI: UEV POSTFIX 570
- UI: UEV PREFIX 570
- Unicode 54
 - use with SQL Server 55
- Unicode characters
 - browser support 53
- Unicode support 328
- UNIX 50
- unspecified
 - property locations 498
- UPDATE counts 55
- update method 275
- Update statistics
 - Oracle 389
 - SQL Server 389
- updating profiles 440
- upgrading
 - data migration 127
- UpperCase element 108
- URL patterns
 - mapping to servlets 222
- URLs
 - system administration 598
- useCountryDefaulting element 329, 332
- useGeneralDefaulting element 329, 332
- useHttp value of Level attribute 174
- useHttps value of Level attribute 174
- UseLocalizedSort element 154
- user administration 419
- user administrators 419
- User class 294
- User Detail Page 420
- user effective status 414
- User Entered Value Allowed Values display property 576
- User Entered Value Postfix display property 576
- User Entered Value Prefix display property 576
- User Entered Value Type display property 575
- user functions 420
- user preferences 415
 - cart mode 415
 - cart view 415
- user status 414
- user types 295
- UserContact data object 183
- user-entered values 636
- username 54, 420
- usernames 437
 - requirements for 411
 - restrictions 437
- users 235
 - accessing pages directly 304
 - creating 420
 - deleting 421
 - deleting (effect on username) 411
 - modifying 421
 - moving by enterprise administrator 438
 - overview 410
 - relation to entitlement roles 411
 - retrieving from session 235
 - searching for 421
 - setting preferred locale 420
- UserType element 296
- UserTypeDefinition element 295
- useSessionCaching system property 200
- using JSP pages as templates 244
- using restore in list beans 273
- UTF-8 54
 - database setting 56

V

- Validate Submodel display property 576
- values
 - of properties 497
- Version attribute 290
- viewing a cron job 592
- viewing the history of a cron job 595
- Visual Modeler
 - accessing 446
 - reports 558
 - tabbed user interface 488, 573
- visual modeler

toolbar 444

W

warning method 308
web.xml configuration file 113, 354
web-app element 114
WebLogic
 clustered implementation 203
WebPathToPublicLoadableWritableDirectory element 116
WebPathToPublicNoLoadableWritableDirectory element 116
Windows 2000 50
workingDir parameter
 in weblogic.xml 89
worksheets 502
 creating 502
 exporting 503
 importing 504
Writable attribute 276
WritableDirectory element 244
writeExternal method 279
WSDLFilter class 355

X

XML message versions 425
XML messages 236
XML representations of data beans 279
XML schema 287
XML transformation 247
XMLLoader.bat script 132
-Xmx setting 394