

SUN SEEBEYOND  
**eGATE™ INTEGRATOR**  
**USER'S GUIDE**

**Release 5.1.1**



Copyright © 2006 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved. Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.sun.com/patents> and one or more additional patents or pending patent applications in the U.S. and in other countries. U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements. Use is subject to license terms. This distribution may include materials developed by third parties. Sun, Sun Microsystems, the Sun logo, Java, Sun Java Composite Application Platform Suite, SeeBeyond, eGate, eInsight, eVision, eTL, eXchange, eView, eIndex, eBAM, eWay, and JMS are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon architecture developed by Sun Microsystems, Inc. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd. This product is covered and controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

Copyright © 2006 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, Etats-Unis. Tous droits réservés. Sun Microsystems, Inc. détient les droits de propriété intellectuelle relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plus des brevets américains listés à l'adresse <http://www.sun.com/patents> et un ou les brevets supplémentaires ou les applications de brevet en attente aux Etats - Unis et dans les autres pays. L'utilisation est soumise aux termes de la Licence. Cette distribution peut comprendre des composants développés par des tierces parties. Sun, Sun Microsystems, le logo Sun, Java, Sun Java Composite Application Platform Suite, Sun, SeeBeyond, eGate, eInsight, eVision, eTL, eXchange, eView, eIndex, eBAM et eWay sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd. Ce produit est couvert à la législation américaine en matière de contrôle des exportations et peut être soumis à la réglementation en vigueur dans d'autres pays dans le domaine des exportations et importations. Les utilisations, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes biologiques et chimiques ou du nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers les pays sous embargo américain, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exhaustive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites.

Version 20060629180840

# Contents

---

## Chapter 1

<b>Introduction</b>	<b>32</b>
About eGate Integrator	32
What's New in This Release	32
About This User's Guide	33
What's in This User's Guide	33
Scope	34
Intended Audience	34
Text Conventions	34
Screen Captures	34
Related Documents	35
Sun Microsystems, Inc. Web Site	35
Documentation Feedback	35

---

## Chapter 2

<b>eGate Integrator Overview</b>	<b>36</b>
Introduction	36
Integration Model	37
System Architecture	38
Repository	39
Run-Time Environments	40
Enterprise Designer	41
Enterprise Manager	42

---

## Chapter 3

<b>Enterprise Designer Basics</b>	<b>43</b>
Overview	43
Enterprise Explorer	44
Enterprise Designer Editors	44
Additional Tools	44
Starting Enterprise Designer	45

<b>Navigating Enterprise Designer</b>	<b>46</b>
<b>Menus</b>	<b>46</b>
File Menu	46
Edit Menu	46
Tools Menu	47
View Menu	47
Window Menu	48
Help Menu	48
<b>Toolbar</b>	<b>49</b>
<b>Wizards</b>	<b>50</b>
<b>Browser Buttons</b>	<b>51</b>
<b>Property Configuration Dialogs</b>	<b>52</b>
<b>Configuring Enterprise Designer</b>	<b>53</b>
<b>Setting Up Options</b>	<b>53</b>
Options Setup	53
Language	54
Build Options	54
Preferences	55
<b>Customizing the User Interface</b>	<b>56</b>
Resizing Panes	56
Component Docking	56
Changing the Default Font Size	57

---

## Chapter 4

<b>Global Features</b>	<b>58</b>
<b>Overview</b>	<b>58</b>
<b>The Repository</b>	<b>59</b>
Using the Repository Context Menu	59
Project Explorer Version	59
Environment Explorer Version	60
<b>Using Export/Import Features</b>	<b>62</b>
Exporting a Project or Environment	62
Exporting Using Enterprise Designer	62
Exporting Using the Command Line	66
Importing a Project or Environment	67
Importing Using Enterprise Designer	68
Importing Using the Command Line	74
<b>Using Version Control Features</b>	<b>76</b>
Managing Project Versions	76
Creating Branches	76
Changing Branches	77
Managing Component Versions	78
Accessing Component Version Control Features	79
Checking a Component In	80
Checking a Component Out	82
Viewing a Component's Version History	84
Viewing a Read-Only Version of a Component	86
Retrieving a Component to Your Workspace	87



Checking a Component In Without Revisions	89
Checking In a Previous Version as the Latest Version	90
Tagging Components	92
Merging Changes from Another Version	94
Command-line Utilities	96
<b>Moving and Replicating Repository Objects</b>	<b>97</b>
Copying and Pasting	97
Projects	97
Components	97
Cutting and Pasting	98
Projects	98
Components	98
<b>Analyzing Component Dependencies</b>	<b>99</b>
About Component Dependencies	99
Using the Impact Analyzer	101

---

## Chapter 5

<b>Projects</b>	<b>102</b>
Overview	102
About Project Explorer	103
Project Explorer Icons	104
Project Components	105
Using the Project Context Menu	105
Using Project Component Context Menus	109
Creating a Project	111
Services	113
Business Processes	113
Collaborations	113
Message Destinations	115
Naming a Message Destination	116
Standard Characters	116
Special Characters	116
External Applications	117
Web Service External Applications	118
Schedulers	119
Configuring a Scheduler	119
Schedule	119
Time Zone	120
Naming a Scheduler	120
Standard Characters	120
Special Characters	120
Constants and Variables	121
Project Constants	122
Project Variables	123
Variables & Constants Object Group	124

<b>Connectivity Maps</b>	<b>125</b>
Using the Connectivity Map Editor	126
Connectivity Map Editor Toolbar	127
Connecting Components	128
Linking Multiple Destinations	129
<b>Deployment Profiles</b>	<b>130</b>
Using the Deployment Editor	130
Creating a Deployment Profile	132
Automapping	135
Mapping Variables	136
Version Control	137
Building an Application File	139
Building an Application File From the Command Line	140
Deploying the Project	145
Deploying to a Sun Java System Application Server	145

## Chapter 6

<b>Object Type Definitions I</b>	<b>146</b>
<b>Understanding OTDs</b>	<b>146</b>
OTD Fundamentals	147
Metadata and OTDs	147
Data Structure	148
Data Types	149
Name Properties	149
OTD Types	150
Externally-Defined OTDs	150
User-Defined OTDs	150
Predefined OTDs	150
<b>Specifying Data Conversion and Encoding</b>	<b>151</b>
Encoding Options	151
Marshaling and Unmarshaling Methods	152
marshal()	152
marshalToString()	153
marshalToBytes()	154
unmarshal()	155
unmarshalFromString()	156
unmarshalFromBytes()	157
Setting Delimiters	158
<b>Using the OTD Editor</b>	<b>159</b>
OTD Editor Toolbar Icons	160
OTD Tester	162
Input Panel	163
Output Panel	164
Status Panel	164
Verbose Panel	165
OTD Tester Controls	166
Marshal/Unmarshal Processes	168
Data Encoding	168
Using the OTD Tester	169

---

**Chapter 7**

<b>Object Type Definitions II</b>	<b>172</b>
Overview	172
Using the New OTD Wizard	173
Creating Externally-Defined OTDs	174
Using the DTD Wizard	174
Using the XSD Wizard	181
Creating User-Defined OTDs	191
Using the User-Defined OTD Wizard	191
Using the UD OTD from File Wizard	193
Using Predefined OTDs	196
Using JMS OTDs	197
Using Scheduler OTDs	200
Using the OTD Context Menu	202

---

**Chapter 8**

<b>Object Type Definitions III</b>	<b>204</b>
Overview	204
Creating and Managing OTD Nodes	205
Moving Nodes	207
Cutting, Copying, and Pasting Nodes	208
Creating and Using Templates	209
External Templates	209
Internal Templates	211
OTD Node Representations	212
Element Nodes	212
Field Nodes	213
Editing OTD Properties	214
Root Node Properties	214
Element Node Properties	216
Order Property	217
Field Node Properties	218
Specifying the Node Type	220
Default Values	221
Specifying Read/Write Access	222
Matching Data Patterns	223
Defining Byte Patterns	223
Specifying Pattern Alignment	223
Specifying Delimiters	225
Delimiter List	225
Delimiter Properties	226

Delimiter Levels	227
Delimiter Type	229
Escape Option	229
Delimiter Bytes	230
Escape Sequences	230
Multiple Delimiters	231
Precedence	232
Optional Mode	233
Terminator Mode	235
Creating a Delimiter List	236

## Chapter 9

<b>Collaboration Definitions (Java) I</b>	<b>242</b>
Overview	242
Using the Collaboration Definition Context Menu	244
Using the Collaboration Editor (Java)	246
Collaboration Editor Toolbar Icons	248
Business Rules Editor	251
Business Rules Editor Toolbar Icons	251
Business Rules Tree	253
Java Source Editor	254
Java Source Editor Toolbar Icons	254
Business Rules Designer	256
Mapping Between Repeating Nodes	256
Business Rules Designer Toolbar Icons	257
Node Menu	258
Method-Access Nodes	259
Collaboration Method Menus	260
Collaboration Method Boxes	261
Placing Collaboration Methods	262
Editing or Adding Literals	262
Graphical Collaboration Methods (Java)	263
Array Methods	263
Assignment	265
Boolean Methods	268
Comparison Methods	270
Math Methods	272
Object Methods	275
Operators	277
String Methods	279
Using the Class Browser	282
Class Browser Toolbar Icons	283
Using the Method Dialog	287
Auto Type Conversion	290
Character Conversion	292
String to Number Conversion	292
Character Encoding	294

Byte-to-String Conversion	294
String-to-Byte Conversion	295
<b>Using the Collaboration Tester (Java)</b>	<b>296</b>
Breakpoints	297
Code Validation	298
<b>Using the Java Debugger</b>	<b>303</b>
Starting the Java Debugger	303
Setting Breakpoints	305
Inspecting and Editing the Source Code	308
Stepping Into, Over, or Out	309
Inspecting Java Threads and Methods	310
Inspecting a Local Variable or Method	311
Saving and Resuming Debug Sessions	314

## Chapter 10

<b>Collaboration Definitions (Java) II</b>	<b>315</b>
<b>Creating a Java-based Collaboration Definition</b>	<b>316</b>
New Web Service	318
Existing Web Service	322
<b>Editing Collaboration Definition Properties</b>	<b>324</b>
<b>Creating Classes</b>	<b>327</b>
<b>Creating Methods</b>	<b>329</b>
Creating Parameters	331
<b>Creating Constructors</b>	<b>333</b>
Creating Parameters	335
<b>Creating Fields</b>	<b>336</b>
<b>Creating Arrays</b>	<b>338</b>
<b>Creating and Modifying Variables</b>	<b>340</b>
<b>Creating and Modifying Literals</b>	<b>344</b>
Using the Toolbar Menus	344
Using the Method Dialog	344
<b>Generating Alerts</b>	<b>345</b>
<b>Creating Log Entries</b>	<b>349</b>
<b>Importing Third-Party Java Classes</b>	<b>352</b>
<b>Adding Java classes to Collaboration Definitions</b>	<b>354</b>
<b>Copying OTD Nodes</b>	<b>356</b>
copyNode()	356
<b>Using Try-Catch</b>	<b>358</b>
<b>Using Byte-Array Converters</b>	<b>361</b>
Defining Encoding Converter Methods	362

Validating Java-based Collaboration Definitions	376
---	-----

---

## Chapter 11

<b>Collaboration Definitions (XSLT)</b>	<b>377</b>
Overview	377
Using the Collaboration Definition Context Menu	379
Creating an XSLT-based Collaboration Definition	381
New Web Service	383
Existing Web Service	386
Using the Collaboration Editor (XSLT)	387
XSLT Toolbar Icons	389
XSLT Tester Icons	390
Collaboration Method Menus	391
Collaboration Method Palette (XSLT)	392
Collaboration Method Boxes (XSLT)	392
Graphical Collaboration Methods (XSLT)	393
Boolean Methods	393
Nodes Methods	395
Number Methods	398
Operator Methods	400
String Methods	403
Design Procedures	406
Creating a For-Each Construct	406

---

## Chapter 12

<b>Web Services</b>	<b>409</b>
Overview	409
Terminology	410
Web Services Security	411
HTTP Basic Authentication	411
Transport Layer Security (TLS)	411
SOAP Message Security	411
Developing Web Services	413
Exposing Collaboration Definitions as Web Services	413
Creating a Web Service Project	414
Procedure	414
Publishing a Web Service Project	415
Procedure	415
Accessing Web Service Definitions	417
Creating XML Schema Definitions	418
Using the Context Menu	418
Using the XML Schema Editor	419
Overview	419

Toolbar	420
XML Schema Definition Panel	420
Properties Panel	420
XML Source Code Panel	420
XML Schema Editor Toolbar Icons	421
Previewing and Exporting XSD Nodes	424
Preview/Export Options	424
XML Schema Compilation	426
Building an XML Schema Definition	428
XML Schema Definition Root Node	429
Attribute Node	433
AttributeGroup Node	435
Element Node	437
Union Node	441
Group Node	443
ComplexType Node	444
SimpleType Node	446
SimpleType Subnode	449
ListType Node	451
Sequence Node	452
Choice Node	453
All Node	454
Any Node	455
ComplexContent Node	457
SimpleContent Node	458
Extension Node	459
Restriction Node	460
Importing Existing XML Schema Definitions	461
<b>Creating Web Service Definitions</b>	<b>466</b>
Overview	466
Using the Context Menu	466
<b>Using the Web Service Definition Editor</b>	<b>467</b>
Overview	467
Toolbar	468
Web Service Definition Panel	468
Properties Panel	468
Source Code Panel	468
Web Service Definition Editor Toolbar Icons	469
Building a Web Service Definition	472
Web Service Definition Root Node	473
Import Folder	474
Messages Folder	476
Messages	476
Message Parts	478
PortTypes Folder	480
PortTypes	480
Operations	481
Properties Folder	484
Property	484
Property Aliases Folder	486
Property Alias	487
Service Link Type Folder	488
Service Link Role	488

Types Folder	490
Importing Existing Web Service Definitions	491
<b>Configuring WSDL Binding Properties</b>	<b>496</b>
Configuring the PortType Binding	498
PortTypes	498
Configuring the Operation	499
Operations	499
Configuring Messages	500
SOAP Messages	500
Configuring the Message Body	501
Message Body	501
Configuring the Message Header	502
Message Header	502
Configuring a Header Fault	503
Header Fault	503
Configuring a Java Exception	504
Java Exception	504

---

## Chapter 13

<b>Run-time Environments</b>	<b>505</b>
Overview	505
Using the Environment Editor	506
Using Environment Explorer	507
Environment Explorer Icons	508
Environment Components	509
Using the Environment Context Menu	509
Using Environment Component Context Menus	511
Environmental Constants	512
Using the Constant Context Menu	513
Keystores	514
Using the Keystore Context Menu	514
Logical Hosts and Domains	515
Using the Logical Host Context Menu	515
Configuring Domains	517
Integration Servers	518
Using the Integration Server Context Menu	518
Configuring an Integration Server	519
Sun SeeBeyond Integration Server Configuration	519
eInsight Engine Configuration	520
Message Servers	521
Configuring a Message Server	522
Schedulers	523
SOAP/HTTP Web Service External Systems	524
Configuring a SOAP/HTTP Client	525



Web Service Security: HTTP Basic Authentication	525
Web Service Security: HTTP Proxy Server Authentication	526
Web Service Security: SOAP Message Security	527
Web Service External System Configuration	530
<b>Configuring a SOAP/HTTP Server</b>	<b>531</b>
Web Service Security: HTTP Basic Authentication	531
Web Service Security: Transport Layer Security	532
Web Service Security: SOAP Message Security	533
Web Service External System Configuration	537
<b>UDDI External Systems</b>	<b>538</b>
Configuring the UDDI External System	539
<b>Troubleshooting</b>	<b>540</b>
<b>Error Messages</b>	<b>540</b>
OutOfMemory	540
<b>CodeGen Validation Errors</b>	<b>541</b>
Operations ... cannot be exposed to external web services	541
The XSD OTD ... is missing target namespace	541
The WSDL ... is out of date ...	541
<b>Unexpected Behavior</b>	<b>542</b>
Node Names Change when Relaunching XSD OTDs	542

# Figures

Figure 1	eGate Integrator	36
Figure 2	eGate Integrator Implementation Model	38
Figure 3	Distributed eGate Integrator System	39
Figure 4	Enterprise Designer	41
Figure 5	Enterprise Manager	42
Figure 6	Enterprise Designer User Interface	43
Figure 7	Login Dialog	45
Figure 8	Options Setup - Heap Size Dialog	53
Figure 9	Options Setup - Language Dialog	54
Figure 10	Options Setup - Build Options Dialog	54
Figure 11	Options Setup - Preferences Dialog	55
Figure 12	Draggable Splitters	56
Figure 13	Component View Docking Options	57
Figure 14	Repository Context Menu (Project Explorer)	59
Figure 15	Repository Context Menu (Environment Explorer)	60
Figure 16	Export Manager Dialog (1a)	62
Figure 17	Export Manager Dialog (1b)	63
Figure 18	Export Manager Dialog (2)	64
Figure 19	Save As Dialog	64
Figure 20	Enter File Name Dialog (2)	65
Figure 21	Export Status Message Box	65
Figure 22	Repository Context Menu - Import Project Option	68
Figure 23	Project Context Menu - Import Options	68
Figure 24	Import Message Box	69
Figure 25	Import Manager Dialog (1)	69
Figure 26	Open File Dialog	70
Figure 27	Import Manager Dialog	71
Figure 28	Project Import Confirmation Dialog	72
Figure 29	Create New Version Confirmation Dialog	72
Figure 30	Import Status Message Box	73
Figure 31	Repository HEAD Branch	76
Figure 32	Repository Context Menu - Change Branch	77

Figure 33	Change Branch Dialog	77
Figure 34	Checked In Icon (OTD Example)	78
Figure 35	Checked Out Icon	78
Figure 36	Retrieved Icon	78
Figure 37	Connectivity Map Context Menu	79
Figure 38	Example Context Menu - Check In Option	80
Figure 39	Version Control - Check In Dialog	81
Figure 40	Example Context Menu - Check Out Option	82
Figure 41	Version Control - Check Out Dialog	83
Figure 42	Example Context Menu - Version History Option	84
Figure 43	Version Control - History Dialog	84
Figure 44	Version Control - History Dialog	86
Figure 45	Version Control - History Dialog	87
Figure 46	Access File Dialog	88
Figure 47	Confirm Version Replace Dialog	88
Figure 48	Example Context Menu - Undo Check Out Option	89
Figure 49	Version Control - Undo Check Out Dialog	89
Figure 50	Example Context Menu - Make Latest Option	90
Figure 51	Make Latest Dialog	90
Figure 52	Confirm Latest Version Override Dialog	91
Figure 53	Version Control Tag Dialog	92
Figure 54	Example Context Menu - Tag Option	93
Figure 55	Version Control Tag Dialog	93
Figure 56	Collaboration Definition Context Menu - Create Diff Option	94
Figure 57	Version Control - Create Diff Dialog	94
Figure 58	Collaboration Definition Context Menu - Merge Diff Option	95
Figure 59	Version Control - Merge Changes Dialog	95
Figure 60	Objects Referenced by Deployment Profile	99
Figure 61	Objects Referencing Deployment Profile	99
Figure 62	Objects Referenced by Connectivity Map	100
Figure 63	Objects Referencing Connectivity Map	100
Figure 64	Objects Referenced by Collaboration Definition	100
Figure 65	Objects Referencing Collaboration Definition	100
Figure 66	Impact Analyzer Dialog	101
Figure 67	Project Explorer	103
Figure 68	Project Context Menu	105
Figure 69	Project Component Context Menu	109
Figure 70	Repository Context Menu - New Project	111

Figure 71	Project Context Menu - New Connectivity Map	111
Figure 72	Defining the Service Component	113
Figure 73	Topic Icon	115
Figure 74	Queue Icon	115
Figure 75	Project Context Menu - New Queue	115
Figure 76	External Application Drop-Down Menu	117
Figure 77	Web Service External Application Icon	118
Figure 78	Project Context Menu - New Web Services Application	118
Figure 79	Scheduler Icon	119
Figure 80	Scheduler Properties Dialog - Schedule	119
Figure 81	Scheduler Properties Dialog - Time Zone	120
Figure 82	Project Context Menu - New Variable or Constant	121
Figure 83	Creating a Project Constant	122
Figure 84	Creating a Project Variable	123
Figure 85	Variables and Constants Object Group	124
Figure 86	Project Context Menu - New Connectivity Map	125
Figure 87	Connectivity Map Editor	126
Figure 88	Connection Icons in a Connectivity Map - Not Configured	128
Figure 89	Connection Icons in a Connectivity Map - Configured	129
Figure 90	Linking Multiple Destinations	129
Figure 91	eGate Integrator Implementation Model	130
Figure 92	Deployment Editor	131
Figure 93	Example Project Connectivity Map	132
Figure 94	Web Client Example Environment	133
Figure 95	Example Deployment Profile (1)	133
Figure 96	Example Deployment Profile (2)	134
Figure 97	Example Deployment Profile (3)	134
Figure 98	Automap Icon	135
Figure 99	Automap Results Dialog Box	135
Figure 100	Automap Options Dialog Box	135
Figure 101	Map Variables Button	136
Figure 102	Deployment Profile Mappings	136
Figure 103	Project Variable Value Entry	136
Figure 104	Spreadsheet View Icon	137
Figure 105	Spreadsheet View	137
Figure 106	Snapshot Dialog	138
Figure 107	Global Settings Dialog	138
Figure 108	Build Button	139

Figure 109	Deploy Button	145
Figure 110	OTD Generation	146
Figure 111	Message Processing	147
Figure 112	OTD Node Nomenclature	148
Figure 113	marshal()	152
Figure 114	marshalToString()	153
Figure 115	marshalToBytes()	154
Figure 116	unmarshal()	155
Figure 117	unmarshalFromString()	156
Figure 118	unmarshalFromBytes()	157
Figure 119	Setting Delimiters	158
Figure 120	OTD Editor	159
Figure 121	Tester Icon	162
Figure 122	OTD Tester Interface	162
Figure 123	Data Field Display	163
Figure 124	Output Panel	164
Figure 125	Status Panel	164
Figure 126	Verbose Panel	165
Figure 127	Input View Encoding Option Menu	168
Figure 128	Data Display - Status Panel	169
Figure 129	OTD Tester Node Table	170
Figure 130	Serialized Data in Output Panel	170
Figure 131	Saving Test Data	171
Figure 132	OTD Wizard Selection Dialog	173
Figure 133	OTD Wizard Selection: DTD Wizard	174
Figure 134	Select DTD File(s) Dialog	175
Figure 135	Cannot Create OTD Warning Box	175
Figure 136	Select Encoding Dialog	176
Figure 137	Select Document Elements Dialog	177
Figure 138	Select OTD Options Dialog	178
Figure 139	OTD Context Menu	180
Figure 140	OTD Wizard Selection: XSD Wizard	181
Figure 141	XSD Wizard: Select XSD File(s)	182
Figure 142	Cannot Create OTD Warning Box	182
Figure 143	Select Encoding Dialog	183
Figure 144	Select Document Elements Dialog	184
Figure 145	Select OTD Options Dialog	185
Figure 146	OTD Context Menu	190

Figure 147	OTD Wizard Selection: User-Defined OTD	191
Figure 148	Enter OTD Name	192
Figure 149	OTD Wizard Selection: UD OTD from File	194
Figure 150	Select a File Dialog	195
Figure 151	Project Explorer - Sun SeeBeyond Folder	196
Figure 152	Project Explorer - JMS OTD	197
Figure 153	JMS OTD Property Nodes	197
Figure 154	Existing Web Service: Select Operation	198
Figure 155	Existing Web Service: Select OTDs	199
Figure 156	Project Explorer - Scheduler OTD	200
Figure 157	Existing Web Service: Select OTDs	201
Figure 158	OTD Context Menu	202
Figure 159	OTD Node Context Menu	205
Figure 160	Node-moving Options	207
Figure 161	Node Context Menu - Paste Options	208
Figure 162	Import Dialog	209
Figure 163	Element Node Created for OTD	212
Figure 164	Element Node from External Template	212
Figure 165	Element Node from Internal Template	212
Figure 166	Field Node Created for OTD	213
Figure 167	Field Node from External Template	213
Figure 168	Internal Template - Field Node	213
Figure 169	Root Node Properties (delim nodeType)	214
Figure 170	Element Node Properties (delim nodeType)	216
Figure 171	Order Property Example	217
Figure 172	Field Node Properties (delim nodeType)	218
Figure 173	nodeType Property Menu (Root/Element Example)	220
Figure 174	Access Property Menu	222
Figure 175	Match Property	223
Figure 176	Align Property Menu	224
Figure 177	Delimiter List Editor	225
Figure 178	OTD Hierarchical and Delimiter Levels	227
Figure 179	Delimiter Type - Escape	229
Figure 180	Multiple Delimiter Example	231
Figure 181	Optional Mode Property (Example 1)	233
Figure 182	Optional Mode Property (Example 2)	234
Figure 183	Terminator Mode Property Example	235
Figure 184	Sample OTD Tree	236

Figure 185	Initial Root Node Properties	236
Figure 186	Activated delim Value Field	237
Figure 187	Delimiter List Editor - Delimiters Not Set	237
Figure 188	Delimiter List Editor - Insert Level	238
Figure 189	Delimiter List Editor - Add Delimiter	238
Figure 190	Delimiter List Editor - Add Levels and Delimiters	239
Figure 191	Root Node - Delimiter Specified	239
Figure 192	Element_1 Node Properties	240
Figure 193	Element_2 Node Properties	240
Figure 194	Field_1 Node Properties	241
Figure 195	Collaboration Definition Context Menu	244
Figure 196	Collaboration Editor (Java)	246
Figure 197	Collaboration Editor (Java) - Source Code Mode	247
Figure 198	Business Rules Editor	251
Figure 199	Java Source Editor	254
Figure 200	Business Rules Designer	256
Figure 201	Node Menu	258
Figure 202	Method-Access Nodes	259
Figure 203	Business Rules Designer: Method Menus and Boxes	260
Figure 204	Collaboration Method Palette (Java)	261
Figure 205	Expanded Method Box	261
Figure 206	Collapsed Method Box	261
Figure 207	Method Box Context Menu - Literal	262
Figure 208	Array Methods Menu	263
Figure 209	Assignment Menu	265
Figure 210	Boolean Methods Menu	268
Figure 211	Comparison Methods Menu	270
Figure 212	Math Methods Menu	272
Figure 213	Object Methods Menu	275
Figure 214	Operators Menu	277
Figure 215	String Methods Menu	279
Figure 216	Class Browser Icon	282
Figure 217	Class/Method Browser	282
Figure 218	Class and Field	284
Figure 219	Class Browser - Select Constructor	285
Figure 220	Constructor Box on Mapping Canvas	286
Figure 221	Node Context Menu	287
Figure 222	Method Dialog (for type TextMessage)	288

Figure 223	Method List for Text	289
Figure 224	Method Box in Mapping Panel	289
Figure 225	Exception Handling Options	290
Figure 226	Automatic Conversion Option	290
Figure 227	Type Conversion - String to Number	292
Figure 228	Type Conversion - Character Encoding	294
Figure 229	Test Mode Icon	296
Figure 230	Collaboration Tester (Java)	296
Figure 231	Breakpoints in Java Code	297
Figure 232	Breakpoint Menu	298
Figure 233	Validation Panel	298
Figure 234	Source Code Mode Icon	299
Figure 235	Test Mode Icon	299
Figure 236	Initial Collaboration Tree	299
Figure 237	Root Node Context Menu	300
Figure 238	Selecting Input Data File	301
Figure 239	Populated Value Fields	302
Figure 240	Java Debugger	303
Figure 241	File Menu	304
Figure 242	Attach to JVM Dialog	304
Figure 243	Java Source Code Display	305
Figure 244	Breakpoint Example	305
Figure 245	Debug Menu	306
Figure 246	Stop in Method Dialog	306
Figure 247	Choose Exception Dialog	307
Figure 248	Break on Exception Dialog	307
Figure 249	Breakpoint Indicator	308
Figure 250	Stepping Into, Over, and Out Commands	309
Figure 251	Java Thread and Method Display	310
Figure 252	Local Variables Tab	311
Figure 253	Evaluate Local Variable	312
Figure 254	Evaluate Method	313
Figure 255	Save Debugger Session Dialog	314
Figure 256	Resume Debugger Session Dialog	314
Figure 257	Project Context Menu - New Web Service Definition	316
Figure 258	Initial Wizard Dialog	317
Figure 259	New Web Service: Operation Name	318
Figure 260	New Web Service: Input Message	319



Figure 261	New Web Service: Output Message	320
Figure 262	New Web Service: Auxiliary OTD	321
Figure 263	Existing Web Service: Select Operation	322
Figure 264	Existing Web Service: Select OTD	323
Figure 265	Collaboration Definition (Java) Properties	324
Figure 266	Collaboration Definition (Java) Properties	325
Figure 267	Collaboration Definition (Java) Properties	326
Figure 268	Class Icon	327
Figure 269	Create Class Dialog	327
Figure 270	Method Icon	329
Figure 271	Create Method Dialog	329
Figure 272	Create Parameter Dialog	331
Figure 273	Constructor Icon	333
Figure 274	Create Constructor Dialog	333
Figure 275	Create Parameter Dialog	335
Figure 276	Field Icon	336
Figure 277	Create Field Dialog	336
Figure 278	New Array Icon	338
Figure 279	New Array Dialog	338
Figure 280	Local Variable Icon	340
Figure 281	Create Variable Dialog	340
Figure 282	Create Variable Dialog	341
Figure 283	Class Browser - Select Class	342
Figure 284	Business Rules with Variable	342
Figure 285	Variable Dialog	343
Figure 286	Modify Variable Dialog	343
Figure 287	Object Icon	344
Figure 288	String Icon	344
Figure 289	Empty File Test - If Condition	345
Figure 290	Empty File Test - Alerter Node Menu	346
Figure 291	Alerter Severity Method List	347
Figure 292	Empty File Test - critical Method	347
Figure 293	Empty File Test - Pass Alert Message	348
Figure 294	Logging Example - Logger Node Menu	349
Figure 295	Logging Level Method List	350
Figure 296	Logging Example - debug Method Box	350
Figure 297	Logging Example - Pass Log Message	351
Figure 298	Project Context Menu: Import File	352

Figure 299	File Import Dialog	353
Figure 300	File Context Menu - Update	353
Figure 301	JAR File Icon	354
Figure 302	Add/Remove Jar File Dialog (1)	354
Figure 303	Add/Remove Jar File Dialog (2)	355
Figure 304	copyNode Operation	356
Figure 305	Try Icon	358
Figure 306	Try Statement	358
Figure 307	Catch Icon	358
Figure 308	Create Exception Variable Dialog	359
Figure 309	Class Browser Dialog	359
Figure 310	Catch Statement	360
Figure 311	Encoding Converter Icon	362
Figure 312	Encoding Converter Methods Dialog	362
Figure 313	Using the unmarshalFromBytes Method	365
Figure 314	Create a Variable Dialog	366
Figure 315	Class Browser Dialog	367
Figure 316	Adding the MyConverter Local Variable	368
Figure 317	Encoding Converter Icon	368
Figure 318	Encoding Converter Method Dialog	369
Figure 319	Creating the Encoding Converter Rule	370
Figure 320	Selecting a Method	371
Figure 321	Convert Method List	372
Figure 322	Data Mapping	373
Figure 323	Marshaling Output Data	374
Figure 324	Output Data	375
Figure 325	Validate Icon	376
Figure 326	Validating Java Code	376
Figure 327	Collaboration Definition Context Menu	379
Figure 328	Project Context Menu - New Web Service Definition	381
Figure 329	Collaboration Wizard (XSLT) Dialog	382
Figure 330	New Web Service: Operation Name	383
Figure 331	New Web Service: Input Message	384
Figure 332	New Web Service: Output Message	385
Figure 333	Existing Web Service: Select Operation	386
Figure 334	Collaboration Editor (XSLT)	387
Figure 335	Encoding Icon	390
Figure 336	Collaboration Definition Editor (XSLT): Method Menus and Boxes	391

Figure 337	Collaboration Method Palette (XSLT)	392
Figure 338	Expanded Method Box	392
Figure 339	Collapsed Method Box	392
Figure 340	Boolean Methods Menu	393
Figure 341	Nodes Methods Menu	395
Figure 342	Number Methods Menu	398
Figure 343	Operator Methods Menu	400
Figure 344	String Methods Menu	403
Figure 345	For-Each Construct - Step 1	406
Figure 346	For-Each Construct - Step 2	407
Figure 347	For-Each Construct - Step 3	407
Figure 348	Development Flow - Creating a Web Service Project	414
Figure 349	Development Flow - Publishing a Web Service Definition	415
Figure 350	Web Services Access Manager	417
Figure 351	Project Context Menu - New XML Schema Definition	418
Figure 352	XML Schema Editor	419
Figure 353	XSD Node Preview/Export Options - Output Encoding	425
Figure 354	XSD Node Preview/Export Options - Synthetic Element Name	425
Figure 355	XSD Tree Example	428
Figure 356	XSD Root Node Icon	429
Figure 357	XSD Root Node Menu	429
Figure 358	Select XSD Dialog	431
Figure 359	XSD Root Node Properties	431
Figure 360	Attribute Node Icon	433
Figure 361	Attribute Node Properties	433
Figure 362	Attribute Group Node Icon	435
Figure 363	AttributeGroup Node Menu	435
Figure 364	AttributeGroup Node Properties (Global)	435
Figure 365	AttributeGroup Node Properties (Non-global)	435
Figure 366	Select Type Dialog	436
Figure 367	Element Node Icon	437
Figure 368	Element Node Menu	437
Figure 369	Element Node Properties (Global)	438
Figure 370	Element Node Properties (Non-global)	439
Figure 371	Union Node Icon	441
Figure 372	Union Node Properties	441
Figure 373	Select Member Types Dialog	442
Figure 374	Group Node Icon	443

Figure 375	Group Node Menu	443
Figure 376	ComplexType Node Icon	444
Figure 377	ComplexType Node Menu	444
Figure 378	SimpleType Node Icon	446
Figure 379	SimpleType Node Menu	446
Figure 380	SimpleType Node Properties	447
Figure 381	Select Final Value Dialog	448
Figure 382	SimpleType Subnode Menu	449
Figure 383	Enumeration Dialog	450
Figure 384	List Type Node Icon	451
Figure 385	ListType Node Properties	451
Figure 386	Sequence Node Icon	452
Figure 387	Sequence Node Menu	452
Figure 388	Choice Node Icon	453
Figure 389	Choice Node Menu	453
Figure 390	All Node Icon	454
Figure 391	All Node Menu	454
Figure 392	Any Node Icon	455
Figure 393	Any Node Properties	455
Figure 394	Complex Content Node Icon	457
Figure 395	ComplexContent Node Menu	457
Figure 396	Simple Content Node Icon	458
Figure 397	SimpleContent Node Menu	458
Figure 398	Extension Node Icon	459
Figure 399	Extension Node Properties	459
Figure 400	Restriction Node Icon	460
Figure 401	Restriction Node Properties	460
Figure 402	Project Context Menu - Import XML Schema Definition Option	461
Figure 403	Import XSD File Dialog - Step 1	462
Figure 404	Import XSD File Dialog - Step 2	462
Figure 405	Import XSD File Dialog - Step 3	463
Figure 406	Import XSD File Dialog - Step 4	464
Figure 407	Import XSD File Dialog - Step 4 Confirmation	464
Figure 408	Import XSD File Dialog - Step 5	465
Figure 409	Project Context Menu - New Web Service Definition	466
Figure 410	Web Service Definition Editor User Interface	467
Figure 411	WSDL Tree - All Options	472
Figure 412	Web Service Definition Root Node Menu	473

Figure 413	Import Folder Menu	474
Figure 414	Select WSDL, XSD or OTD Dialog	475
Figure 415	Messages Folder Menu	476
Figure 416	Message Menu	476
Figure 417	Add Parts Dialog	477
Figure 418	Message Part Properties (Initial)	478
Figure 419	Select Part Dialog	479
Figure 420	Message Value Properties (String)	479
Figure 421	PortTypes Folder Menu	480
Figure 422	PortType Menu	480
Figure 423	Operation Menu	481
Figure 424	Message Properties	482
Figure 425	Message Dialog	483
Figure 426	Properties Folder Menu	484
Figure 427	Property Properties	484
Figure 428	Select Type Dialog	485
Figure 429	Property Aliases Folder Menu	486
Figure 430	Add Property Aliases Dialog	487
Figure 431	Service Link Type Folder Menu	488
Figure 432	Role Menu	488
Figure 433	Schema Type Menu	490
Figure 434	Project Context Menu - Import Web Service Definition Option	491
Figure 435	WSDL Import Wizard - Step 1	492
Figure 436	WSDL Import Wizard - Step 2	492
Figure 437	WSDL Import Wizard - Step 3	493
Figure 438	WSDL Import Wizard - Step 4	494
Figure 439	WSDL Import Wizard - Step 4 Confirmation	494
Figure 440	WSDL Import Wizard - Step 5	495
Figure 441	Web Service Connection	496
Figure 442	WSDL Binding Editor Displaying Code	497
Figure 443	WSDL Binding Editor - PortType Binding Properties	498
Figure 444	WSDL Binding Editor - Operation Properties	499
Figure 445	WSDL Binding Editor - Message Properties	500
Figure 446	WSDL Binding Editor - Message Body Configuration	501
Figure 447	WSDL Binding Editor - Input Header Properties	502
Figure 448	WSDL Binding Editor - Header Fault Properties	503
Figure 449	WSDL Binding Editor - Java Exception Properties	504
Figure 450	Environment Editor	506

Figure 451	Enterprise Explorer: Environment Explorer View	507
Figure 452	Environment Context Menu	509
Figure 453	Environment Component Context Menu	511
Figure 454	Environment Context Menu - New Constant	512
Figure 455	Environmental Constants Panel	512
Figure 456	Environment Context Menu - New Keystore	514
Figure 457	Keystore Context Menu	514
Figure 458	Environment Menu - New Logical Host	515
Figure 459	Logical Host Context Menu	515
Figure 460	Logical Host Menu - New Integration Server	518
Figure 461	Integration Server Context Menu - Java Debugger	518
Figure 462	Integration Server Context Menu - Properties	519
Figure 463	Sun SeeBeyond Integration Server Configuration Properties	519
Figure 464	Logical Host Menu - New Message Server	521
Figure 465	Message Server Context Menu - Properties	522
Figure 466	Sun SeeBeyond JMS IQ Manager Configuration Properties	522
Figure 467	Environment Context Menu - New Scheduler	523
Figure 468	Environment Context Menu - New SOAP/HTTP Web Service External System	524
Figure 469	Create External System Dialog	524
Figure 470	SOAP/HTTP Web Service External System Context Menu - Properties	525
Figure 471	HTTP Basic Authentication Configuration	525
Figure 472	HTTP Proxy Server Configuration Dialog	526
Figure 473	SOAP Message Security Configuration Dialog	527
Figure 474	User Name Token Configuration Dialog	528
Figure 475	Digest Password Type Configuration Dialog	529
Figure 476	Web Service External System Configuration Dialog	530
Figure 477	SOAP/HTTP Web Service External System Context Menu - Properties	531
Figure 478	HTTP Basic Authentication Configuration Dialog	531
Figure 479	Transport Layer Security Configuration Dialog	532
Figure 480	SOAP Message Security Configuration Dialog	533
Figure 481	User Name Token Configuration Dialog	534
Figure 482	Admin Server Connection Configuration Dialog	535
Figure 483	Digest Password Type Configuration Dialog	536
Figure 484	Web Service External System Configuration Dialog	537
Figure 485	Environment Context Menu - New UDDI External System	538
Figure 486	Create UDDI External System Dialog	538
Figure 487	UDDI External System Context Menu - Properties	539
Figure 488	UDDI System Configuration Properties Dialog	539

# Tables

Table 1	Text Conventions	34
Table 2	File Menu Options	46
Table 3	Tools Menu Options	46
Table 4	Tools Menu Options	47
Table 5	View Menu Options	47
Table 6	Window Menu Options	48
Table 7	Help Menu Options	48
Table 8	Enterprise Designer Toolbar Icons	49
Table 9	Wizard Navigation Buttons	50
Table 10	Browser Buttons	51
Table 11	Configuration Dialog Toolbar Buttons	52
Table 12	Repository Context Menu Options (Project Explorer)	59
Table 13	Repository Context Menu Options (Environment Explorer)	60
Table 14	Connectivity Map Menu Options - Version Control	79
Table 15	Impact Analyzer Command Buttons	101
Table 16	Project Icons	104
Table 17	Project Context Menu Options	105
Table 18	Project Component Context Menu Options	109
Table 19	Project Variable Options	122
Table 20	Project Variable Options	123
Table 21	Connectivity Map Editor Toolbar Icons	127
Table 22	Deployment Editor Toolbar Commands	131
Table 23	Commandline Codegen Properties	140
Table 24	Encoding Options According to Locale	151
Table 25	OTD Editor User Interface	159
Table 26	OTD Editor Toolbar Icons	160
Table 27	OTD Tester Controls	166
Table 28	Marshal and Unmarshal Process Differences	168
Table 29	DTD OTD Options	178
Table 30	XSD OTD Options	186
Table 31	Strict Validation - XSD Facets	188
Table 32	Text File Variables	193



Table 33	JMS OTD Properties	198
Table 34	OTD Context Menu Options	202
Table 35	OTD Node Context Menu Options	205
Table 36	Root Node Properties	214
Table 37	Element Node Properties	216
Table 38	Order Property Example	217
Table 39	Field Node Properties	218
Table 40	nodeType Property Options	220
Table 41	nodeType Default Values	221
Table 42	Access Parameter Options	222
Table 43	Align Parameter Options	224
Table 44	Delimiter List Editor Command Buttons	226
Table 45	Delimiter Properties	226
Table 46	Delimiter Type Options	229
Table 47	Escape Sequences	230
Table 48	Optional Mode Options	233
Table 49	Terminator Mode Options	235
Table 50	Collaboration Definition Context Menu Options	244
Table 51	Collaboration Editor Toolbar Icons	248
Table 52	Business Rules Editor Toolbar Buttons	251
Table 53	Java Source Editor Toolbar Icons	254
Table 54	Business Rules Designer Toolbar Icons	257
Table 55	Business Rules Designer Node Menu Options	258
Table 56	Method Access Nodes	259
Table 57	Array Operation Collaboration Methods (Java)	263
Table 58	Assignment (Java)	265
Table 59	Boolean Collaboration Methods (Java)	268
Table 60	Additional Boolean Collaboration Methods (Java)	269
Table 61	Comparison Collaboration Methods (Java)	270
Table 62	Math Collaboration Methods (Java)	272
Table 63	Object Collaboration Methods (Java)	275
Table 64	Operators (Java)	277
Table 65	String Collaboration Methods (Java)	279
Table 66	Method Browser Toolbar Icons	283
Table 67	Method Dialog Toolbar Icons	288
Table 68	Supported Datatype Conversions	291
Table 69	Special Characters	293
Table 70	Collaboration Tester (Java) Commands	297



Table 71	Breakpoint Menu Options	298
Table 72	Class Options	328
Table 73	Method Options	330
Table 74	Parameter Options	332
Table 75	Constructor Options	334
Table 76	Parameter Options	335
Table 77	Field Options	337
Table 78	Array Options	338
Table 79	Variable Options	340
Table 80	Literals in Toolbar Menus	344
Table 81	Byte-Array Encoding Converters - Japanese	361
Table 82	Byte-Array Encoding Converters - Korean	361
Table 83	Encoding Converter Methods Dialog Options	363
Table 84	Collaboration Definition Context Menu Options	379
Table 85	XSLT Toolbar Icons	389
Table 86	XSLT Tester Icons	390
Table 87	Boolean Collaboration Methods (XSLT)	393
Table 88	Nodes Collaboration Methods (XSLT)	395
Table 89	Number Collaboration Methods (XSLT)	398
Table 90	Operator Collaboration Methods (XSLT)	400
Table 91	String Collaboration Methods (XSLT)	403
Table 92	Metadata Forms	410
Table 93	Password Types for User Name Token	412
Table 94	XML Schema Editor Toolbar Icons	421
Table 95	XSD Node Preview/Export Options	424
Table 96	Duplicated Global Schema Component Options	426
Table 97	Enforce UPA Rule Options	426
Table 98	Enforce Particle Restriction Rule Options	427
Table 99	XSD Root Node Menu Options	429
Table 100	XSD Root Node Property Options	432
Table 101	Attribute Node Property Options	433
Table 102	AttributeGroup Node Menu Options	435
Table 103	AttributeGroup Node Property Options	436
Table 104	Element Node Menu Options	437
Table 105	Element Node Property Options (Global)	438
Table 106	Element Node Property Options	439
Table 107	Union Node Property Options	441
Table 108	Group Node Menu Options	443

Table 109	ComplexType Node Menu Options	444
Table 110	SimpleType Node Menu Options	446
Table 111	SimpleType Node Property Options	447
Table 112	SimpleType Subnode Menu Options	449
Table 113	ListType Node Property Options	451
Table 114	Sequence Node Menu Options	452
Table 115	Choice Node Menu Options	453
Table 116	All Node Menu Options	454
Table 117	Any Node Property Options	455
Table 118	ComplexContent Node Menu Options	457
Table 119	SimpleContent Node Menu Options	458
Table 120	Extension Node Property Options	459
Table 121	Restriction Node Property Options	460
Table 122	Web Service Definition Editor Toolbar Icons	469
Table 123	Web Service Definition Root Node Menu Options	473
Table 124	Import Folder Menu Options	474
Table 125	Messages Folder Menu Options	476
Table 126	Message Menu Options	476
Table 127	Message Value Node Property Options	478
Table 128	PortTypes Folder Menu Options	480
Table 129	PortType Menu Options	480
Table 130	Operation Menu Options	481
Table 131	Properties Folder Menu Options	484
Table 132	Properties Folder Menu Options	486
Table 133	Service Link Type Folder Menu Options	488
Table 134	Role Menu Options	488
Table 135	Schema Type Menu Options	490
Table 136	WSDL Binding Editor Toolbar Icons	496
Table 137	PortType Binding Property Options	498
Table 138	Operation Property Options	499
Table 139	Message Property Options	500
Table 140	Message Body Property Options	501
Table 141	Header Property Options	502
Table 142	Header Fault Property Options	503
Table 143	Java Exception Property Options	504
Table 144	Environment Icons	508
Table 145	Environment Context Menu Options	509
Table 146	Environment Component Context Menu Options	511

Table 147	Environmental Constants Panel Icons	513
Table 148	Keystore Context Menu Options	514
Table 149	Logical Host Context Menu Options	516
Table 150	Sun SeeBeyond Integration Server Configuration Properties List	519
Table 151	Sun SeeBeyond JMS IQ Manager Configuration Properties List	522
Table 152	HTTP Basic Authentication Configuration Properties	526
Table 153	HTTP Proxy Server Configuration Properties	526
Table 154	SOAP Message Security Configuration Properties	527
Table 155	User Name Token Configuration Properties	528
Table 156	Digest Password Type Configuration Properties	529
Table 157	Web Service External System Configuration Properties	530
Table 158	HTTP Basic Authentication Configuration Properties	532
Table 159	Transport Layer Security Configuration Properties	532
Table 160	SOAP Message Security Configuration Properties	533
Table 161	User Name Token Configuration Properties	534
Table 162	Admin Server Connection Configuration Properties	535
Table 163	Digest Password Type Configuration Properties	536
Table 164	Web Service External System Configuration Properties	537
Table 165	UDDI External System Configuration Properties	539

# Introduction

This chapter describes the scope and organization of this document, and provides references to additional sources of relevant information.

### What's in This Chapter

- [About eGate Integrator](#) on page 32
- [What's New in This Release](#) on page 32
- [About This User's Guide](#) on page 33
- [Related Documents](#) on page 35
- [Sun Microsystems, Inc. Web Site](#) on page 35
- [Documentation Feedback](#) on page 35

---

## 1.1 About eGate Integrator

eGate Integrator is a distributed integration platform that serves as the foundation of the Sun Java™ Composite Application Platform Suite. The user interfaces — Sun SeeBeyond Enterprise Designer for Project design and Sun SeeBeyond Enterprise Manager for system management and monitoring — ensure a unified look and feel across all editors in the suite, with a single sign-on process for access to any product.

The run-time environment, which is J2EE™-compatible and certified, features high performance and dynamic scalability. The architecture uses Enterprise JavaBeans (EJBs) with Java™ Message Service (JMS) and Java™ Naming and Directory Interface (JNDI).

---

## 1.2 What's New in This Release

For new features and performance improvements, please refer to the *Sun SeeBeyond eGate™ Integrator Release Notes*.

For information regarding operating system and language support, please refer to the *Sun Java™ Composite Application Platform Suite Installation Guide*.

---

## 1.3 About This User's Guide

The following sections provide information about this document, including an overview of its contents, scope, and intended audience.

### 1.3.1 What's in This User's Guide

This User's Guide covers the following topics.

- 1 **Introduction** on page 32 describes the purpose of this User's Guide, including writing conventions and a list of related documents.
- 2 **eGate Integrator Overview** on page 36 provides an overview of the general structure, architecture, and operation of eGate Integrator, and its place within the Java™ Composite Application Platform Suite.
- 3 **Enterprise Designer Basics** on page 43 provides an overview of the Enterprise Designer, including its structure and operation.
- 4 **Global Features** on page 58 describes features used to manage Repository objects in both Project and Environment realms.
- 5 **Projects** on page 102 describes the process of building a Java™ Composite Application Platform Suite Project, including the use of Connectivity Maps and Deployment Profiles.
- 6 **Object Type Definitions I** on page 146 explains the basic concepts involved in Object Type Definitions (OTDs), including data encoding and marshaling/unmarshaling processes; also introduces the OTD Editor.
- 7 **Object Type Definitions II** on page 172 describes how to create Object Type Definitions using the OTD Wizards.
- 8 **Object Type Definitions III** on page 204 explores the details of User-Defined OTDs.
- 9 **Collaboration Definitions (Java) I** on page 242 describes how to build Collaboration Definitions described in Java™.
- 10 **Collaboration Definitions (Java) II** on page 315 describes a variety of specific design procedures used to create Java-based Collaboration Definitions.
- 11 **Collaboration Definitions (XSLT)** on page 377 describes how to build Collaboration Definitions described in XSLT.
- 12 **Web Services** on page 409 describes how to use eGate Integrator in concert with other Java™ Composite Application Platform Suite components to create web services.
- 13 **Run-time Environments** on page 505 explains how to create and populate run-time Environments.
- 14 **Troubleshooting** on page 540 contains tips on troubleshooting problems in eGate Integrator.

In addition, the **Glossary** on page 543 lists various terms used in this User's Guide.

## 1.3.2 Scope

This User's Guide provides general information about the features and operation of Enterprise Designer for creating and deploying Projects.

**Note:** *Any operational explanations provided in this document are generic, for reference purposes only, and do not necessarily address the specifics of setting up individual Projects.*

## 1.3.3 Intended Audience

This User's Guide is intended for developers using Enterprise Designer to build and maintain integration Projects. This guide assumes that readers are experienced computer users, who are familiar with Windows-style GUI operations and have an in-depth understanding of the operating system(s) on which eGate Integrator is installed.

## 1.3.4 Text Conventions

The following conventions are observed throughout this document.

**Table 1** Text Conventions

Text	Used For	Example
<b>Bold</b>	Names of buttons, commands, menu selections, and file names, to accentuate required user actions.	<ul style="list-style-type: none"><li>Click <b>OK</b> to save and close.</li><li>From the File menu, select <b>Exit</b>.</li><li>Select the <b>logicalhost.exe</b> file.</li></ul>
Monospaced	Command-line arguments and code samples (variables are shown within angle brackets).	<code>bootstrap -p &lt;password&gt;</code>
<b>Blue bold</b>	Hypertext links within document.	See <b>Text Conventions</b> on page 34
<u>Blue underlined</u>	Hypertext links for Web addresses (URLs) or email addresses.	<a href="http://www.seebeyond.com">http://www.seebeyond.com</a> <a href="mailto:docfeedback@seebeyond.com">docfeedback@seebeyond.com</a>

## 1.3.5 Screen Captures

Depending on what products you have installed, and how they are configured, the screen captures in this document may differ from what you see on your system.

---

## 1.4 Related Documents

The following documents provide additional information about the eGate Integrator system as explained in this guide:

- *Sun Java™ Composite Application Platform Suite Installation Guide*
- *Sun Java™ Composite Application Platform Suite Primer*
- *Sun SeeBeyond eGate™ Integrator JMS Reference Guide*
- *Sun SeeBeyond eGate™ Integrator Release Notes*
- *Sun SeeBeyond eGate™ Integrator System Administration Guide*
- *Sun SeeBeyond eGate™ Integrator Tutorial*

For information on a specific add-on product (for example, an eWay Intelligent Adapter), see the User's Guide for that product. A complete list of Java™ Composite Application Platform Suite documentation is included in the *Sun Java™ Composite Application Platform Suite Primer*.

The documentation for the Java™ Composite Application Platform Suite is distributed as a collection of online documents, which can be accessed through the Enterprise Manager (see the *Sun Java™ Composite Application Platform Suite Installation Guide*). These documents are in Adobe Acrobat format, which requires that Acrobat Reader be installed on your computer. Acrobat Reader can be from Adobe Systems as a free download from the following URL:

<http://www.adobe.com>

---

## 1.5 Sun Microsystems, Inc. Web Site

The Sun Microsystems web site is your best source for up-to-the-minute product news and technical support information. The site's URL is:

<http://www.sun.com>

---

## 1.6 Documentation Feedback

We appreciate your feedback. Please send any comments or suggestions regarding this document to:

[CAPS\\_docsfeedback@sun.com](mailto:CAPS_docsfeedback@sun.com)

# eGate Integrator Overview

This chapter provides an overview of the conceptual operation and general architecture of the Sun SeeBeyond eGate Integrator system.

## What's in This Chapter

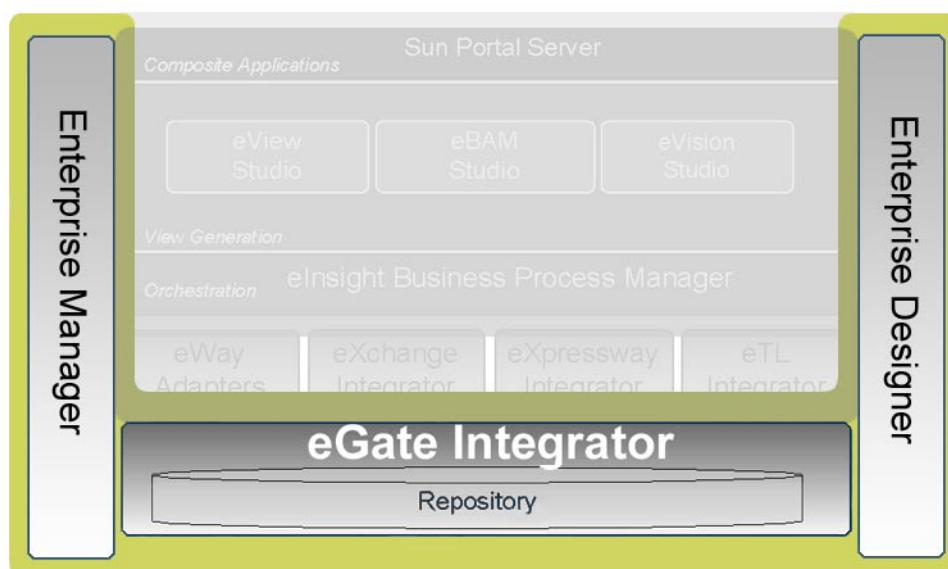
- **Introduction** on page 36
- **Integration Model** on page 37
- **System Architecture** on page 38

---

## 2.1 Introduction

Sun SeeBeyond eGate Integrator is a fully J2EE-certified, web services-based integration platform that serves as the foundation of the Java™ Composite Application Platform Suite. It provides the core integration function, incorporating comprehensive systems connectivity, guaranteed messaging, and robust transformation capabilities. eGate Integrator also provides a unified, single sign-on environment for integration development, deployment, monitoring and management.

**Figure 1** eGate Integrator





As shown in Figure 1, the heart of eGate Integrator is the Repository, which is a comprehensive store of information common to the entire Java™ Composite Application Platform Suite. A separate UDDI registry allows publication and discovery of web services.

The run-time environment employs J2EE-compatible integration servers as operational engines and JMS-compatible message servers for the propagation of messages. The flexibility of the eGate Integrator system allows the option of deployment to a run-time environment extending across a distributed network of hardware platforms.

Enterprise Manager provides a unified, browser-based framework for managing all aspects of the run-time environment, as well as installing and updating all Java™ Composite Application Platform Suite components. Enterprise Designer provides a unified, graphical development environment for integrating systems and developing composite applications using web services.

eGate Integrator can communicate with and link multiple applications and databases across a variety of different operating systems. eGate Integrator performs with a wide variety of hardware, message standards, operating systems, databases, and communication protocols in both real-time and batch (scheduled) integration modes.

---

## 2.2 Integration Model

The Java™ Composite Application Platform Suite addresses application integration by means of an integration Project, which contains the business logic required to solve the specific problem. The Project contains the various logical components and supporting information required to perform the routing, processing, and caching of messages containing the relevant data from one application to another. All Project information is stored in the Repository.

Projects are created using tools contained within Enterprise Designer and, once deployed, can be run and monitored using Enterprise Manager. Projects can also be set up to be run from the business process level using the Sun SeeBeyond eInsight Business Process Manager, if that product is also installed.

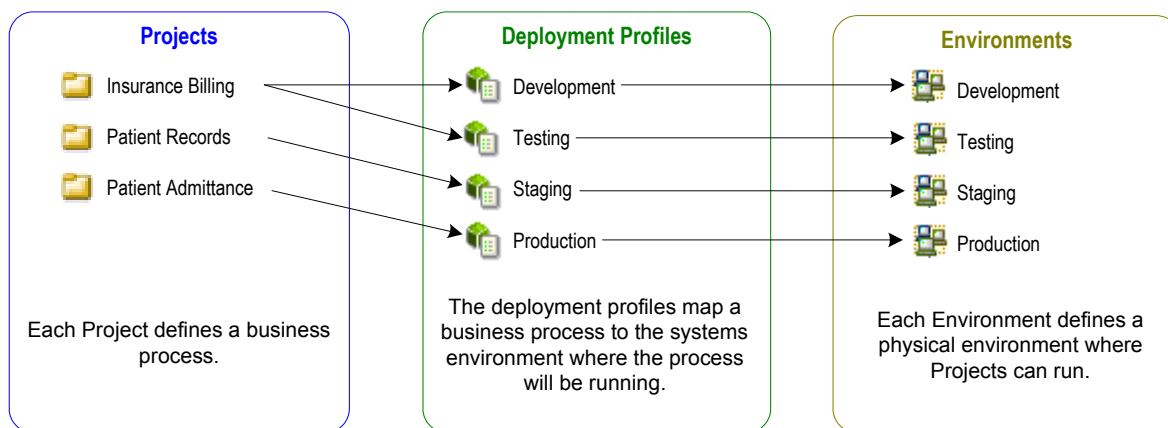
Projects are run within individual sets of system definitions, referred to as Logical Hosts. These are defined within Environments, which represent the physical resources required to implement the Project. Projects are mapped to the individual Environments by means of deployment profiles, which are defined within Enterprise Designer and become part of the Project. Activating the deployment profile deploys the Project to the associated Environment.

This structure of Projects, Environments, and deployment profiles isolates each implementation into logical and physical realms. This provides you with extensive flexibility and efficiency in implementing integration Projects. For example, once you build your Projects and Environments, you have the flexibility to change each realm without having to make changes to the other realm.

The finished Project, of course, will run in your production Environment; separate Environments, having the same structure as the production Environment, should be created for development and testing. You may also want some additional

Environments, such as staging. The following figure illustrates the eGate Integrator implementation model using a healthcare-related example.

**Figure 2** eGate Integrator Implementation Model



Any of the Projects shown in the figure can be deployed to any of the Environments via the mapping defined in the deployment profiles. The example in the figure above shows that the patient admittance Project is already in the production phase and therefore was deployed using the production deployment profile. The patient records Project is in the staging phase and was therefore deployed to the staging Environment using the staging deployment profile. The insurance billing Project is still being developed and tested, and therefore it is deployed to development and testing via the development and testing profiles.

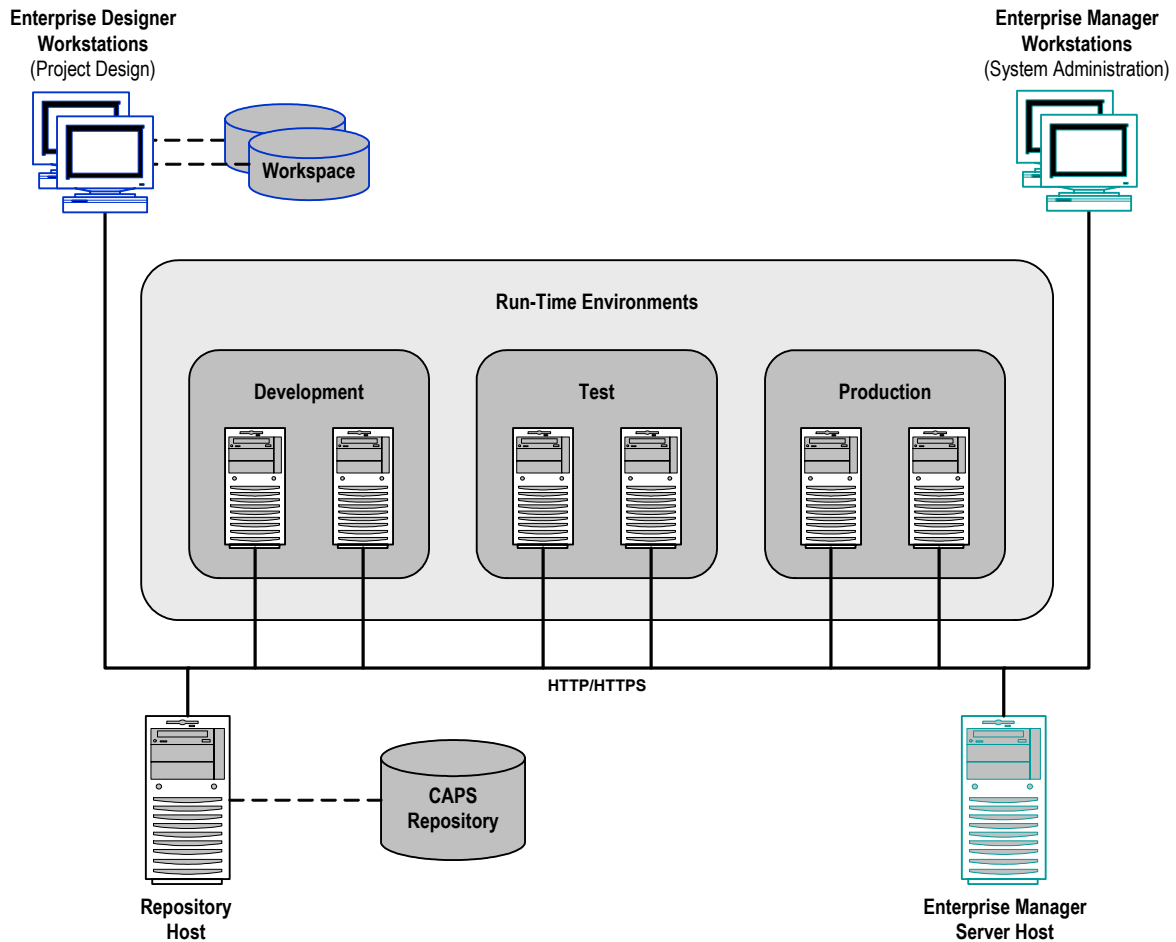
Typically, a particular Project will exist in different versions — a mature version in the production environment, and a follow-on version in the development environment, for example. Isolating these different versions from each other is handled by maintaining separate branches of the Repository.

## 2.3 System Architecture

eGate Integrator employs a versatile architecture that is ideally suited to distributed computing environments. As a result, the various components of an eGate Integrator system can reside on the same hardware platform (assuming adequate system resources), or be distributed across several different hardware platforms in the enterprise network.

Figure 3 shows an example of a distributed system implementation having six separate run-time servers in addition to the Repository server and, optionally, an Enterprise Manager server. Each run-time server can host multiple domains, each consisting of an application server-message server pair.

**Figure 3** Distributed eGate Integrator System



**Note:** This scenario assumes that all instances of eGate Integrator are of the same release.

### 2.3.1 Repository

The setup, components, and configuration information for the elements of a Project are stored in the Repository. As shown in Figure 3, a single Repository serves the entire enterprise—this common Repository is used for development, testing, and production purposes. Communication between the Repository and other Java™ Composite Application Platform Suite components can be configured to use either HTTP or HTTPS. The Enterprise Designer and Enterprise Manager clients can communicate with the Repository through a firewall.

**Note:** The eGate run-time components, which include domains (instances of Logical Hosts) and any Projects or components deployed to them, provide their full functionality independent of the Repository.

### 2.3.2 Run-Time Environments

An Environment represents the total system required to implement a Project. It consists of a collection of Logical Hosts, capable of hosting components of the Java™ Composite Application Platform Suite, along with information about external systems involved in the implementation.

- **Logical Hosts and Domains**

Each Environment contains one or more system definitions. Each definition must include one or more **application servers** such as the Sun SeeBeyond Integration Server, which are the engines that run Collaborations and eWays, and one or more **message servers** such as the Sun SeeBeyond JMS IQ Manager, which manage JMS topics (publish-and-subscribe messaging) and queues (point-to-point messaging). Each collection of integration servers and message servers, plus additional software modules, comprise what is known as a *Logical Host*. The run-time instance of a Logical Host is known as a *domain*.

- **External Systems**

An external system is a representation of a real, physical system that exists within the specific Environment, with configuration properties for locating and accessing that system.

In the example system shown in Figure 3, the production environment is split across two hardware platforms. Separate environments for development and testing should duplicate the structure of the production environment. The test environment should be supported by hardware similar to that supporting the production environment, to allow performance and load testing to give representative throughput results. The hardware supporting the development environment, however, does not usually have the same performance requirements as that supporting the test and production environments.

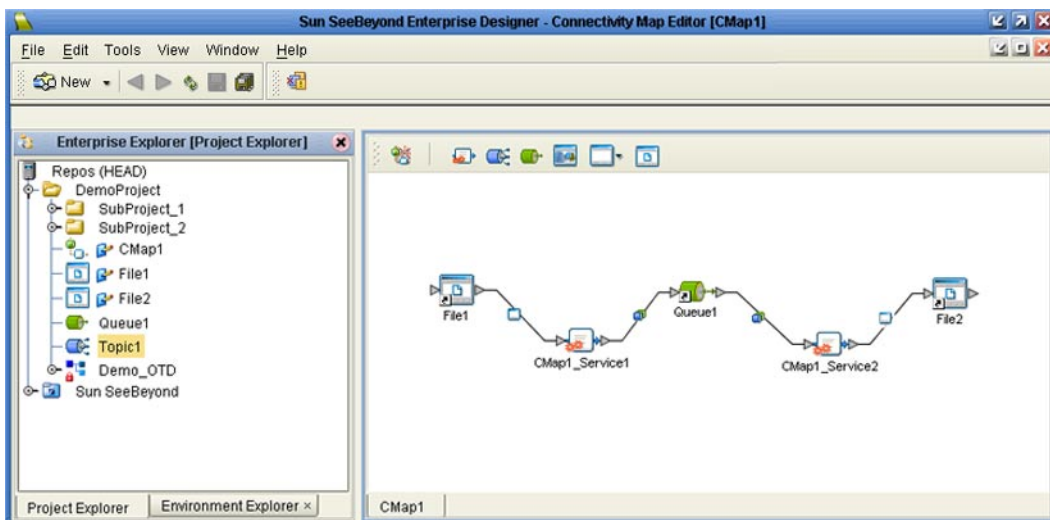
An integration Project is created within the development environment, then migrated to the test environment, and finally to the production environment. This migration path is a necessary and highly critical practice in implementing a working system.

Note again that there is no requirement for the components shown in Figure 3 to run on separate systems; all could run on a single system, provided that resources (processing, memory and storage) are sufficient to support concurrent usage.

### 2.3.3 Enterprise Designer

Enterprise Designer — the primary subject of this User's Guide — is used to create and configure the logical components and physical resources of a Project. Using this development tool (see Figure 4), you can develop Projects to process and route data between external applications or systems.

**Figure 4** Enterprise Designer



The major features of Enterprise Designer are an explorer panel on the left, and an editor panel on the right (which is initially blank). Enterprise Explorer follows the familiar Windows Explorer format, displaying a tree structure. When you open a component in Enterprise Explorer, the appropriate editor appears in the editor panel, displaying the selected component.

The Connectivity Map Editor shown in Figure 4 provides a simple example of one of these editors, in which logical components of a Project can be created and connected. eGate Integrator uses Connectivity Maps to intuitively configure the end-to-end flow of messages within an integration scenario. You can drag and drop the various Project components onto the Connectivity Map canvas, and link them together to specify message flow. The features and usage of the Connectivity Map Editor are described in [Projects](#) on page 102.

Enterprise Designer also serves as the design platform for other products in the Java™ Composite Application Platform Suite, such as eInsight and eXchange. For more information on using Enterprise Designer with these other Java™ Composite Application Platform Suite products, see the documentation for the individual products.

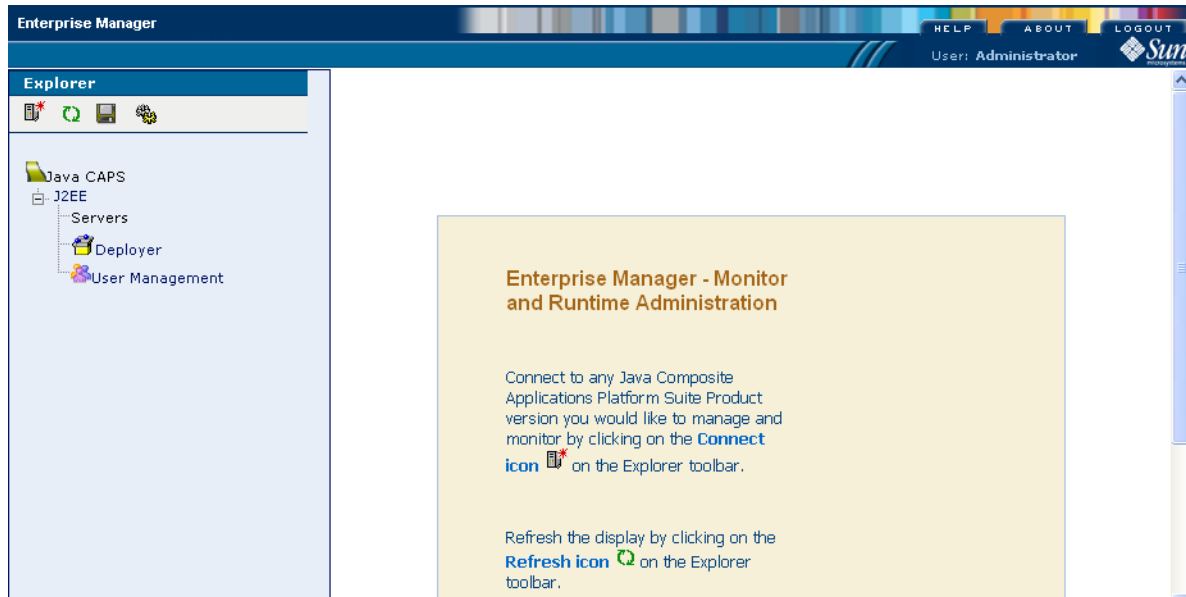
Enterprise Designer additionally supports security-related system administration tasks such as managing access to various components and features in the Java™ Composite Application Platform Suite. The use of Enterprise Designer in performing these tasks is described in the *Sun SeeBeyond eGate™ Integrator System Administration Guide*.

## 2.3.4 Enterprise Manager

Enterprise Manager is a web-based interface with which system administrators can manage running Java™ Composite Application Platform Suite applications for both the Java™ 2 Enterprise Edition (J2EE™) and the Schema Runtime Environment (SRE).

Enterprise Manager contains an Explorer panel on the left and a Details panel on the right. The home page of Enterprise Manager is shown in Figure 5.

**Figure 5** Enterprise Manager



The various features of Enterprise Manager are described in detail in the *Sun Java™ Composite Application Platform Suite Installation Guide* and the *Sun SeeBeyond eGate™ Integrator System Administration Guide*.

# Enterprise Designer Basics

This chapter presents an introduction to Enterprise Designer, and describes those features that apply to Enterprise Designer itself.

## What's in This Chapter

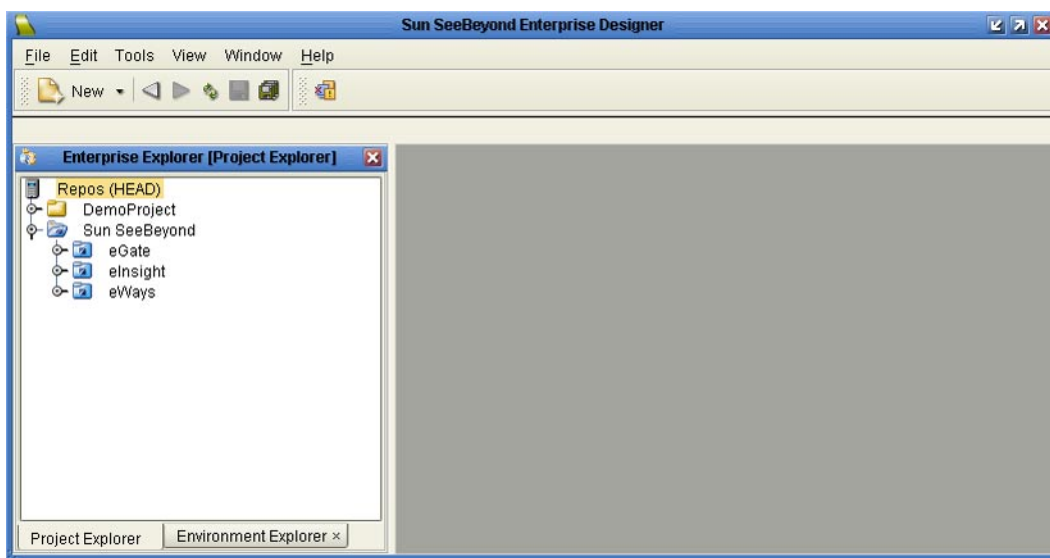
- [Overview](#) on page 43
- [Starting Enterprise Designer](#) on page 45
- [Navigating Enterprise Designer](#) on page 46
- [Configuring Enterprise Designer](#) on page 53

---

## 3.1 Overview

Enterprise Designer provides a graphical user interface with which you can create and configure your Projects. The major features of Enterprise Designer are the Enterprise Explorer on the left, and an editor panel on the right — which is initially blank (see Figure 6). At the top are a menu bar and a toolbar.

**Figure 6** Enterprise Designer User Interface



### 3.1.1 Enterprise Explorer

Enterprise Explorer, encompassing both Project Explorer and Environment Explorer, displays the contents of the selected Repository branch. The two views display tree structures respectively representing the logical components defined for the Projects and physical resources defined for the Environments to which the Projects are deployed. The two explorer views are described in detail in:

- [About Project Explorer](#) on page 103
- [Using Environment Explorer](#) on page 507

### 3.1.2 Enterprise Designer Editors

Enterprise Designer contains several editors, each specialized for a specific purpose. When you open Enterprise Designer, the editor panel is initially blank, as shown in Figure 6. When you open a component in Enterprise Explorer, the appropriate editor appears in the editor panel, displaying the selected component. Additional facilities are also displayed here, such as the Java™ Debugger. The various editors and tools are described in:

- [Using the Connectivity Map Editor](#) on page 126
- [Using the Deployment Editor](#) on page 130
- [Using the OTD Editor](#) on page 159
- [Using the Collaboration Editor \(Java\)](#) on page 246
- [Using the Collaboration Editor \(XSLT\)](#) on page 387
- [Using the XML Schema Editor](#) on page 419
- [Using the Web Service Definition Editor](#) on page 467
- [Using the Environment Editor](#) on page 506
- [Using the Java Debugger](#) on page 303

### 3.1.3 Additional Tools

Enterprise Designer includes several additional tools, accessed either from the menu bar or Enterprise Explorer, that are used for:

- Importing and exporting Projects and Environments (see [Using Export/Import Features](#) on page 62).
- Managing the versions of Project and Environment components (see [Using Version Control Features](#) on page 76).
- Analyzing the impact of removing components (see [Analyzing Component Dependencies](#) on page 99).



## 3.2 Starting Enterprise Designer

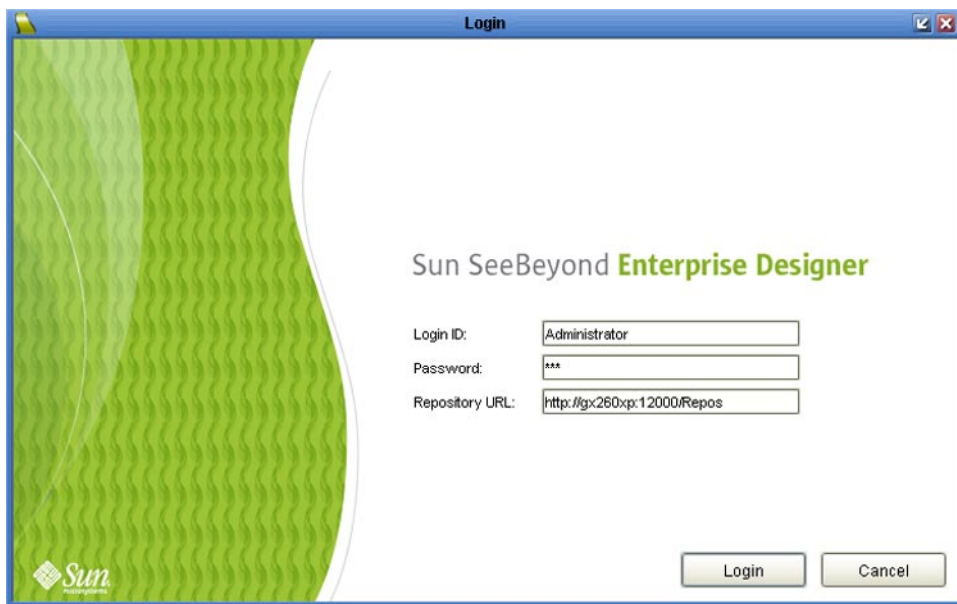
Before we proceed further, perhaps you would like to know how to start Enterprise Designer. It's described more completely in the *Sun Java™ Composite Application Platform Suite Installation Guide*, but here's a condensed version:

### To start Enterprise Designer

- 1 Run the following batch file to display the *Login* dialog shown in Figure 7 (placing a shortcut on your desktop streamlines this procedure):

```
<Sun_JavaCAPS_install_dir>\edesigner\bin\runed.bat
```

**Figure 7** Login Dialog



- 2 Type your *Login ID* and *Password* into the respective boxes (the Login ID from the last successful login is entered automatically).
- 3 The URL for the Repository should be displayed in the *Repository URL* text box. If it is incorrect, edit the URL before proceeding. See the *Sun Java™ Composite Application Platform Suite Installation Guide* for details.
- 4 Click **Login** to complete the login process and display Enterprise Designer as shown in Figure 6. A progress monitor will appear while the login process is running.

**Note:** *Enterprise Designer currently runs exclusively on Microsoft Windows.*

## 3.3 Navigating Enterprise Designer

In this section we describe the functionality of the various buttons, icons, and menu options contained in Enterprise Designer.

### 3.3.1 Menus

The menu bar provides access to a variety of options for managing your Project or Environment. The following tables describe the actions resulting from selecting the indicated options.

#### File Menu

**Table 2** File Menu Options

Option	Action
New	Displays a submenu that lists the objects that can be added to the selected node (duplicates the <b>New</b> button in the toolbar and the <b>New</b> option in the selected node's context menu). Enabled only for Repository, Project, Environment, and Logical Host nodes.
Save	Saves changes to the selected objects (to the local workspace).
Save All	Saves changes to all objects currently open in the editor (to the local workspace).
Exit	Closes Enterprise Designer.

#### Edit Menu

The Edit menu is enabled for Enterprise Explorer and selected editors, including the Connectivity Map Editor, Collaboration Editor (Java), OTD Editor, and the eInsight BPEL Editor. Only allowed operations are enabled in any case.

**Table 3** Tools Menu Options

Option	Action
Cut	Copies the selected object and removes it from the current location, after which you can paste it to another location (once only). All changes must be committed before you can cut the object. Cut and paste is disabled for other users when you have the object checked out. See <a href="#">Moving and Replicating Repository Objects</a> on page 97 for additional information.
Copy	Copies the selected object, after which you can paste it to other locations (multiple times). All changes must be committed before you can copy the object. You can copy and paste an object even when another user has the object checked out. See <a href="#">Moving and Replicating Repository Objects</a> on page 97 for additional information.

**Table 3** Tools Menu Options

Option	Action
Paste	Pastes a cut or copied object into the selected location (Projects are pasted as Subprojects). Pasting is disabled for other users when you have the Project checked out. See <a href="#">Moving and Replicating Repository Objects</a> on page 97 for additional information.
Delete	<p>Deletes the selected object, subject to the following conditions:</p> <ul style="list-style-type: none"> <li>You have <i>write</i> privileges for the object.</li> <li>The object is not checked out by anyone other than yourself.</li> </ul> <p>If these conditions are true, a dialog is displayed in which you confirm that you want to delete the selected object. Clicking <b>Yes</b> then deletes the object.</p> <p>If the selected object contains active Deployment Profiles, a dialog is displayed listing those deployments and requesting that you deactivate them. You must select deactivation to enable the <b>OK</b> button.</p>
Rename	Activates the field, allowing you to rename the selected object.

## Tools Menu

**Table 4** Tools Menu Options

Option	Action
Impact Analyzer	Presents a dialog in which you can view how one component of a Project impacts other components. See <a href="#">Analyzing Component Dependencies</a> on page 99.
Options	Presents the Options Setup dialog, in which you can specify selected options such as heap sizes and language extensions. See <a href="#">Setting Up Options</a> on page 53.
Update Center	Presents a series of dialogs in which you can check for program updates. See the <i>Sun Java™ Composite Application Platform Suite Installation Guide</i> .

## View Menu

**Table 5** View Menu Options

Option	Action
Environment Explorer	Activates the <b>Environment Explorer</b> tab on the Enterprise Explorer.
Project Explorer	Activates the <b>Project Explorer</b> tab on the Enterprise Explorer.
Show/Hide Business Rules Designer	Toggle command to display or hide the Business Rules Designer in the open Editor, if applicable. Not fully operational in eGate Integrator.

**Table 5** View Menu Options

Option	Action
Show/Hide Code Text Editor	Toggle command to display or hide the Code Text Editor in the open Editor, if applicable. Not fully operational in eGate Integrator.
Show/Hide Properties Editor	Toggle command to display or hide the Properties panel in the open Editor, if applicable. Not fully operational in eGate Integrator.
Grid	Not operational in eGate Integrator.
Link Style	Not operational in eGate Integrator.
Align	Not operational in eGate Integrator.
Distribute	Not operational in eGate Integrator.
Zoom	Not operational in eGate Integrator.

## Window Menu

**Table 6** Window Menu Options

Option	Action
Cascade	Displays all open windows so that each window slightly overlaps the others in the Project Editor.
Tile	Displays all open windows in a stacked tile pattern.
Horizontal Layout	Displays all open windows from top to bottom.
Vertical Layout	Displays all open windows from left to right.
Minimize All	Minimizes all open windows so that only the title bar is shown at the bottom of the Editor canvas.
Restore All	Returns minimized windows to their original position on the Editor canvas.
Close All	Closes all open windows.








## Help Menu

**Table 7** Help Menu Options

Option	Action
About Enterprise Designer	Presents an information box giving the version number, copyright information, and Repository connection information.
Contents	Presents the online help for Enterprise Designer.
Help Sets	Presents the online help for all installed components of the Java™ Composite Application Platform Suite that operate within Enterprise Designer.

### 3.3.2 Toolbar

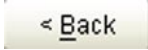




**Table 8** Enterprise Designer Toolbar Icons

Icon	Command	Action
	New	Displays a submenu that lists the objects that can be added to the selected node (duplicates the <b>New</b> menu in the menu bar and the <b>New</b> option in the selected node's context menu). Enabled only for Repository, Project, Environment, and Logical Host nodes.
	Back	Steps back in your usage history for the current session.
	Forward	Steps forward in your usage history for the current session.
	Refresh All from Repository	Refreshes the Project Explorer and Environment Explorer to display the current contents of the Repository. (You are prompted to save any changes before the refresh occurs.) Open editors are not refreshed.
	Save	Saves changes made to the selected Project to the local workspace only — the Repository is <i>not</i> updated. This icon is inactive if no changes have been made.
	Save All	Saves changes made to all open Projects to the local workspace only — the Repository is <i>not</i> updated. This icon is inactive if no changes have been made.
	Impact Analyzer	Presents the <i>Impact Analyzer</i> dialog, which allows you to view how deleting one Project component affects other components.

### 3.3.3 Wizards

Wizards are used extensively in Enterprise Designer for building, or initiating the building process, for OTDs, Collaborations, and various web service-related items. The wizards contain standard navigational features, as shown below.






**Table 9** Wizard Navigation Buttons

Button	Function
	Returns to the previous step in the wizard. This button is disabled on the first step.
	Goes to the next step in the wizard. This button is disabled on the last step.
	Saves all settings for the component and closes the wizard. This button is only enabled on the last step.
	Closes the wizard without saving the component.
	Presents the online help documentation for the wizard dialog.

### 3.3.4 Browser Buttons

The following buttons are used throughout the Enterprise Designer, in wizards and file selection dialogs. They correspond to standard Windows browser buttons.







**Table 10** Browser Buttons

Button	Command	Action
	Up One Level	Returns you to the parent folder or directory.
	Home	Returns you to the root folder or directory, or — depending upon the context — the default object.
	Create New Folder	Creates a new folder under the current folder.
	List	Displays folder/file names only.
	Details	Displays details of the folders or files (name, type, date last modified, etc.).

### 3.3.5 Property Configuration Dialogs

The following buttons are used throughout the Enterprise Designer, in component property configuration dialogs.

**Table 11** Configuration Dialog Toolbar Buttons

Button	Command	Function
	Unsorted	Displays configuration properties in their default order.
	Sort by Name	Sorts configuration properties alphabetically by name.
	Sort by Type	Displays configuration properties by property type.
	Show Editable Properties Only	Displays only the properties of a component that can be modified.
	Customizer	Presents the <b>Customizer</b> dialog, which you can use to customize the selected component.
	Help	Presents the online help documentation for the Configuration Editor.



## 3.4 Configuring Enterprise Designer

### 3.4.1 Setting Up Options

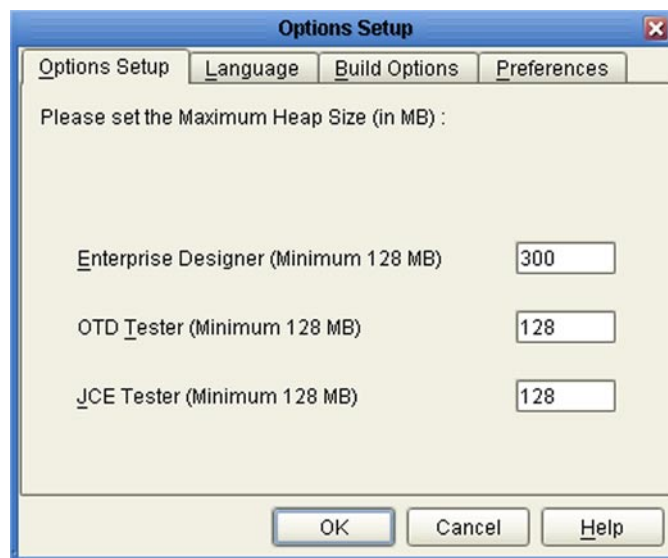
Enterprise Designer offers several global setup and configuration options, which are accessed by clicking **Options** on the *Tools* menu.

#### Options Setup

The *Options Setup* tab (see Figure 8) allows you to increase the heap size of Enterprise Designer itself, and selected modules, to accommodate large file sizes. You can change the respective heap sizes by simply editing the values in the boxes and clicking **OK**. Note that while the minimum recommended size for Enterprise Designer is 128 MB, it is set to 300 MB by default.

**Note:** If you encounter an out-of-memory error, try increasing the heap size in increments of 50Mb. See also [OutOfMemory](#) on page 540.

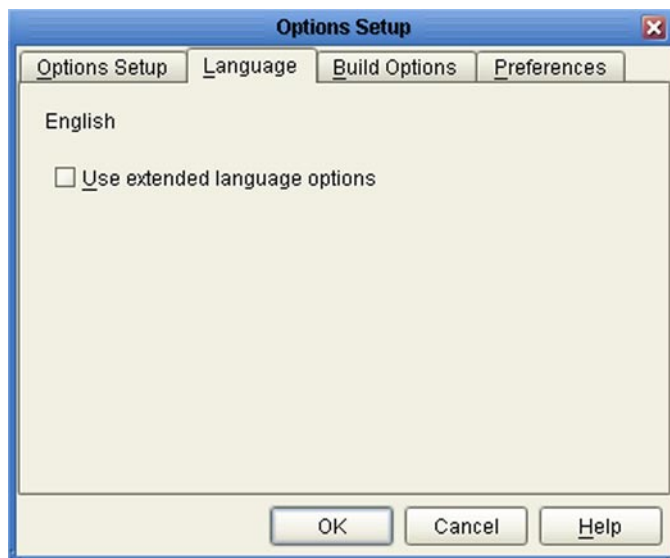
**Figure 8** Options Setup - Heap Size Dialog



## Language

The *Language* tab (see Figure 9) allows you to enable extended language options for handling Chinese, Japanese, and Korean characters in the corresponding locale. Selecting the check box enables the options, which are disabled by default.

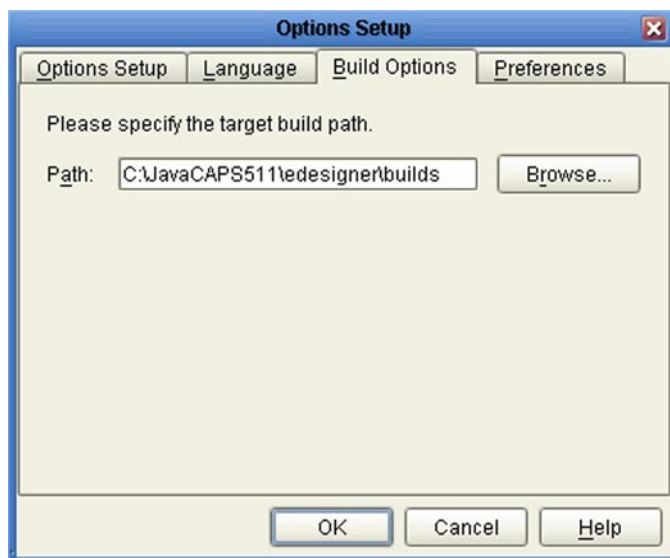
**Figure 9** Options Setup - Language Dialog



## Build Options

The *Build Options* tab (see Figure 10) allows you to specify the path for storing your Project builds. The default value is shown in the figure.

**Figure 10** Options Setup - Build Options Dialog



## Preferences

The *Preferences* tab (see Figure 11) allows you to set selected operating preferences for:

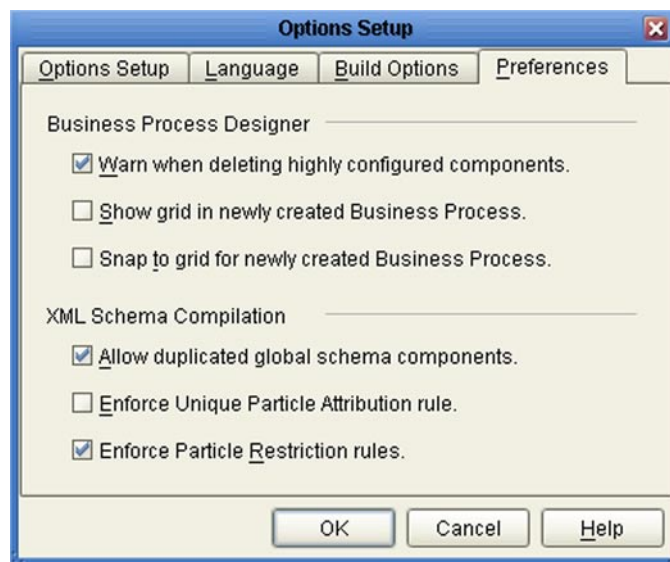
- Business Process Designer

These options apply only if you have installed Sun SeeBeyond eInsight Business Process Manager. See the *Sun SeeBeyond eInsight™ Business Process Manager User's Guide* for information.

- XML Schema Compilation

These options specify the settings for specific XML Schema validation rules (see [XML Schema Compilation](#) on page 426).

**Figure 11** Options Setup - Preferences Dialog



### 3.4.2 Customizing the User Interface

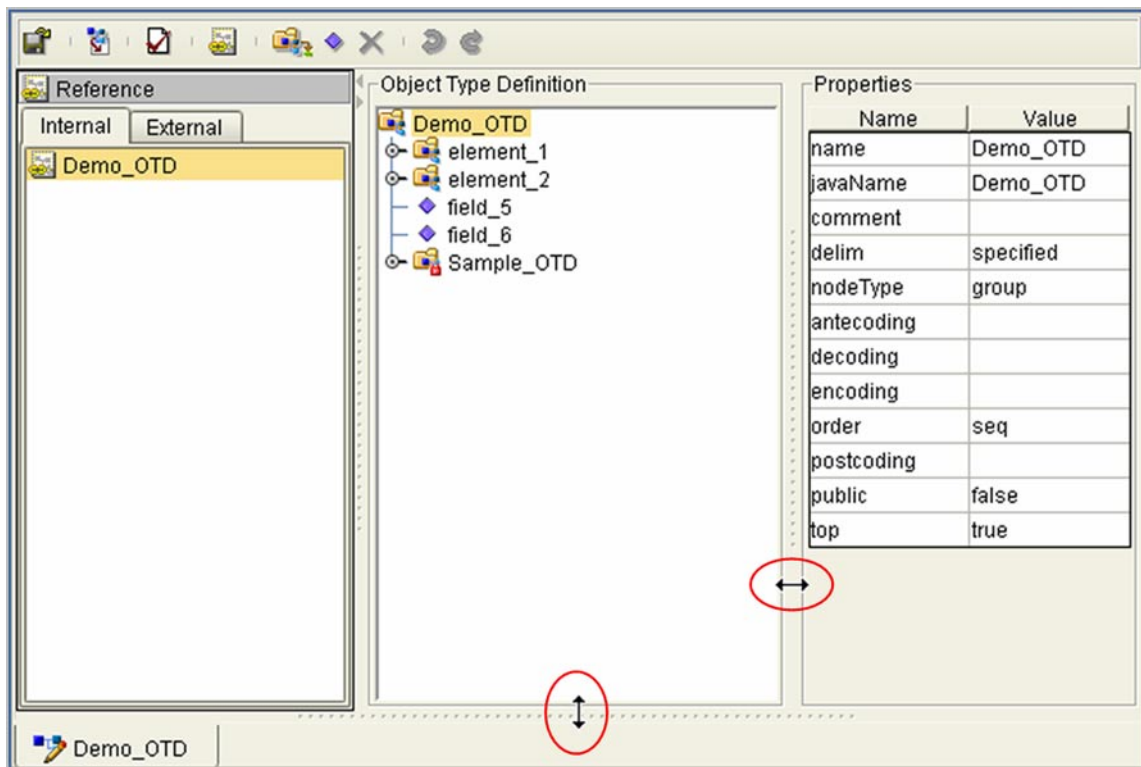
Enterprise Designer is based on the NetBeans windowing system, and offers most of the standard NetBeans window management features. These features allow you to customize the appearance of Enterprise Designer (within limits, of course) to suit your work style. Extensive modification of the interface is discouraged, however, unless you are well acquainted with the NetBeans system and with Enterprise Designer, since the appearance will no longer resemble the examples shown in this User's Guide.

Two basic features are described in the following sections.

#### Resizing Panes

The relative sizes of window panes can be adjusted by dragging the splitters that separate the panes, as shown in Figure 12, for example.

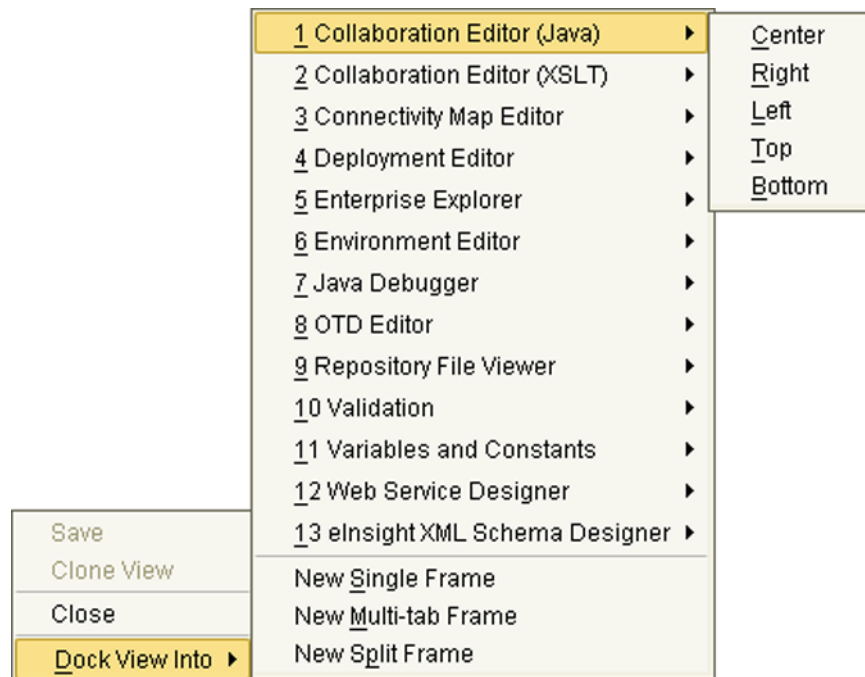
**Figure 12** Draggable Splitters



#### Component Docking

Enterprise Designer allows the docking of views of Project components into different Editor or Explorer windows. As a result, you can mix various components inside a window, using the tabs at the bottom of the window to access the components. You also have the option of specifying the location within the window where the component view is docked. The menu for accomplishing this is accessed by right-clicking the tab for the selected component. A typical expanded menu is shown in Figure 13.

**Figure 13** Component View Docking Options



## Changing the Default Font Size

The default font size of Enterprise Designer is 11 points. You can increase or decrease the font size by modifying the batch file that starts Enterprise Designer.

### To change the default font size

- 1 On the computer where Enterprise Designer is installed, open the following file (in a text editor):  
`<Sun_JavaCAPS_install_dir>\edesigner\bin\runed.bat`
- 2 Add the **-fontsize** argument followed by the font size. For example:  
`-jdkhome %JAVA_HOME% -fontsize 12 -branding stc`
- 3 Save the file.
- 4 If Enterprise Designer is currently running, exit Enterprise Designer and log in again.

# Global Features

This chapter describes the use of features that are used to manage various Repository objects.

### What's in This Chapter

- [Overview](#) on page 58
- [The Repository](#) on page 59
- [Using Export/Import Features](#) on page 62
- [Using Version Control Features](#) on page 76
- [Moving and Replicating Repository Objects](#) on page 97
- [Analyzing Component Dependencies](#) on page 99

---

## 4.1 Overview

Enterprise Designer contains several features that transcend the scope of the following chapters in this User's Guide. For example, export/import and version control features apply to both Projects and Environments.

The cut/copy/paste features described in this chapter pertain to Project Explorer, which represents the contents of the Repository. Additional (or different) cut/copy/paste features are associated with specific editors, such as the OTD Editor, and are described in the appropriate chapters.

The Impact Analyzer is used to determine dependencies between components, for example which Collaborations use a certain OTD. This information is critical if you are about to modify or delete the component.

## 4.2 The Repository

The setup, components, and configuration information for the elements of a Project are stored in the Repository. A single Repository serves the entire enterprise—this common Repository is used for development, testing, and production purposes. Communication between the Repository and other Java™ Composite Application Platform Suite components can be configured to use either HTTP or HTTPS. The Enterprise Designer client can communicate with the Repository through a firewall.

**Note:** *The eGate run-time components, which include domains (instances of Logical Hosts) and any Projects or components deployed to them, provide their full functionality independent of the Repository.*

### 4.2.1 Using the Repository Context Menu

#### Project Explorer Version

Right-clicking the **Repository** in Project Explorer displays the context menu shown in Figure 14.

**Figure 14** Repository Context Menu (Project Explorer)



New Project
Sort by Type
Sort by Name
Sort by Date
Import Project
Export Project
Change Branch...
Create Branch...
Refresh All from Repository
User Management
Paste
Properties

**Table 12** Repository Context Menu Options (Project Explorer)

Option	Function
New Project	Adds a new Project to the current repository branch (see <i>Note</i> below).
Sort by Type	Places all objects in order by grouping object types.
Sort by Name	Places all objects in alphabetical order.
Sort by Date	Places all objects in order by creation date, from oldest to newest.

**Table 12** Repository Context Menu Options (Project Explorer)

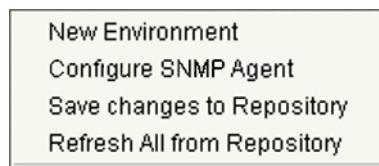
Option	Function
Import Project	Presents a dialog with which you can import a Project into the current repository branch. See <a href="#">Importing a Project or Environment</a> on page 67.
Export Project	Presents a dialog with which you can export a Project from the current repository branch to an archive file. <a href="#">Exporting a Project or Environment</a> on page 62
Change Branch	Presents a dialog with which you can switch to a different repository branch (if one exists). See <a href="#">Changing Branches</a> on page 77.
Create Branch	Presents a dialog with which you can create a new repository branch. See the <i>Sun SeeBeyond eGate™ Integrator System Administration Guide</i> .
Refresh All from Repository	Refreshes the Project Explorer to display the current contents of the current repository branch. (Open editors are not refreshed.)
User Management	Presents the User Management dialog, where an Administrator can manage user access to the current repository branch with options for adding, modifying, and deleting users. See the <i>Sun SeeBeyond eGate™ Integrator System Administration Guide</i> .
Paste	Pastes a cut or copied Project into the current repository branch.
Properties	Presents a dialog showing the configuration properties of the current repository branch. See the <i>Sun SeeBeyond eGate™ Integrator System Administration Guide</i> .

**Important:** Project names should contain only alphanumeric characters (letters and numbers), dashes, and underscores.

## Environment Explorer Version

Right-clicking the **Repository** in Environment Explorer displays the context menu shown in Figure 15.

**Figure 15** Repository Context Menu (Environment Explorer)



**Table 13** Repository Context Menu Options (Environment Explorer)

Option	Function
New Environment	Presents a dialog with which you can create a new Environment.
Configure SNMP Agent	Presents a dialog with which you can modify the SNMP agent properties.



**Table 13** Repository Context Menu Options (Environment Explorer)

Option	Function
Save Changes to Repository	Saves any changes you have made in the Environment Editor to the Repository.
Refresh All from Repository	Refreshes the Environment Explorer to display the current contents of the Repository. (Open editors are not refreshed.)

**Important:** *Environment names should contain only alphanumeric characters (letters and numbers), dashes, and underscores.*

## 4.3 Using Export/Import Features

### 4.3.1 Exporting a Project or Environment

The export function allows you to export a Project and/or Environment to an archive file using either the Enterprise Designer or a command-line script. The archive file can then be imported into another Repository.

When exporting, note that:

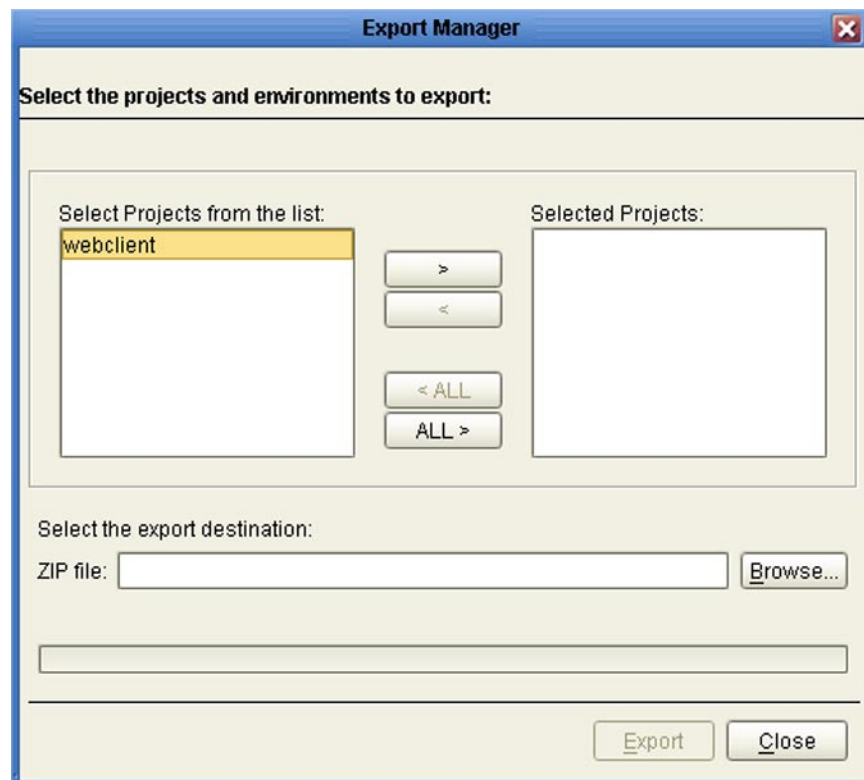
- The exported Project may have references to elements that are in other Projects. A list of such references is generated during the export process.
- Project deployment objects are *not* exported, since they have references to both Project and Environment elements that are not required at the Project level.

### Exporting Using Enterprise Designer

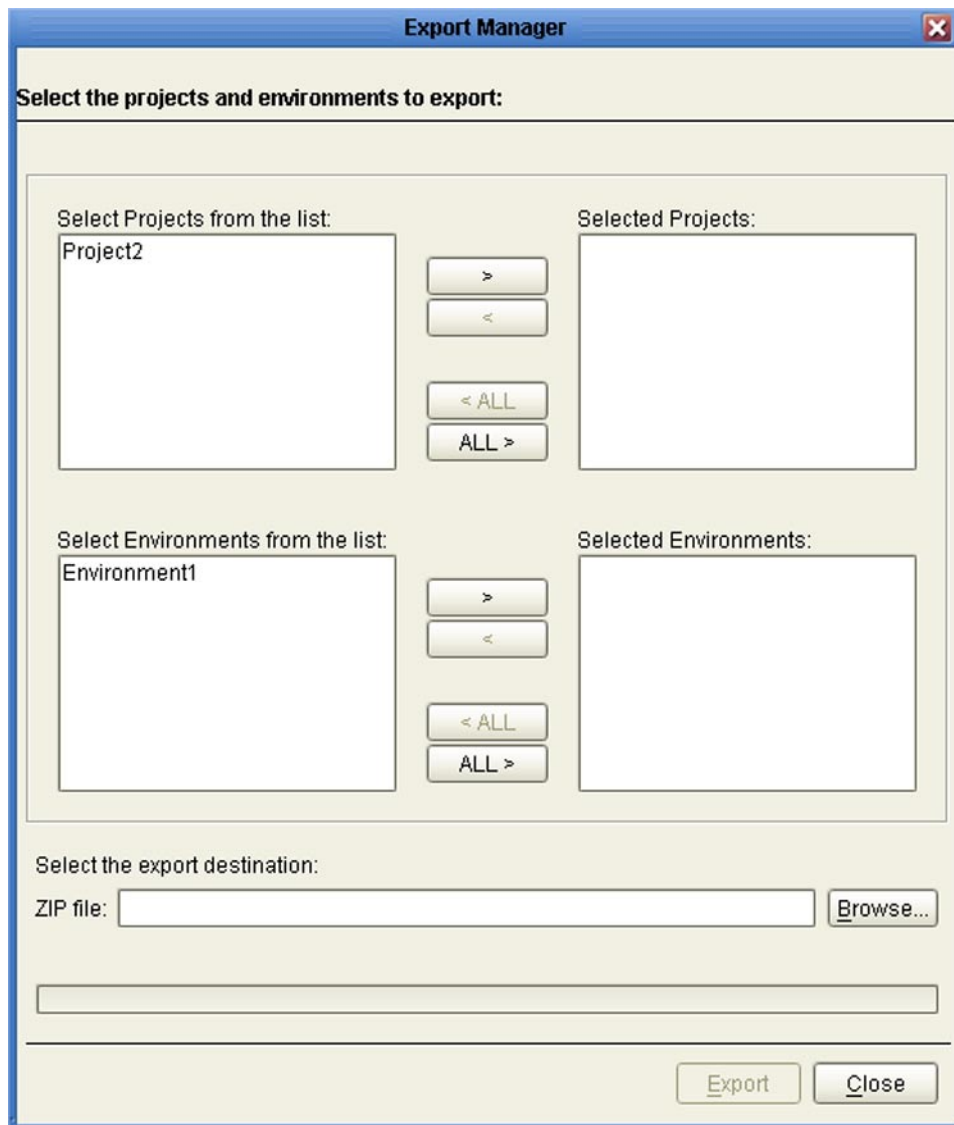
To export a Project or Environment using Enterprise Designer

- 1 From the Repository or Project context menu, select **Export Project** to display the Export Manager dialog. If you do not have any existing Environments in your Repository, you will see the dialog shown in Figure 16. If you do, you will see the dialog shown in Figure 17.

**Figure 16** Export Manager Dialog (1a)

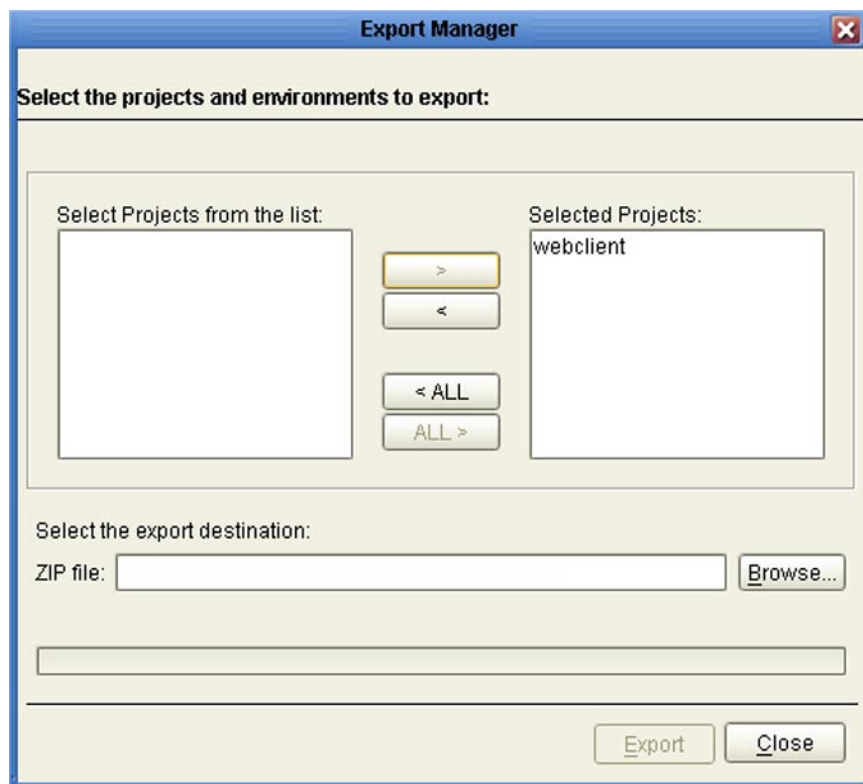


**Figure 17** Export Manager Dialog (1b)



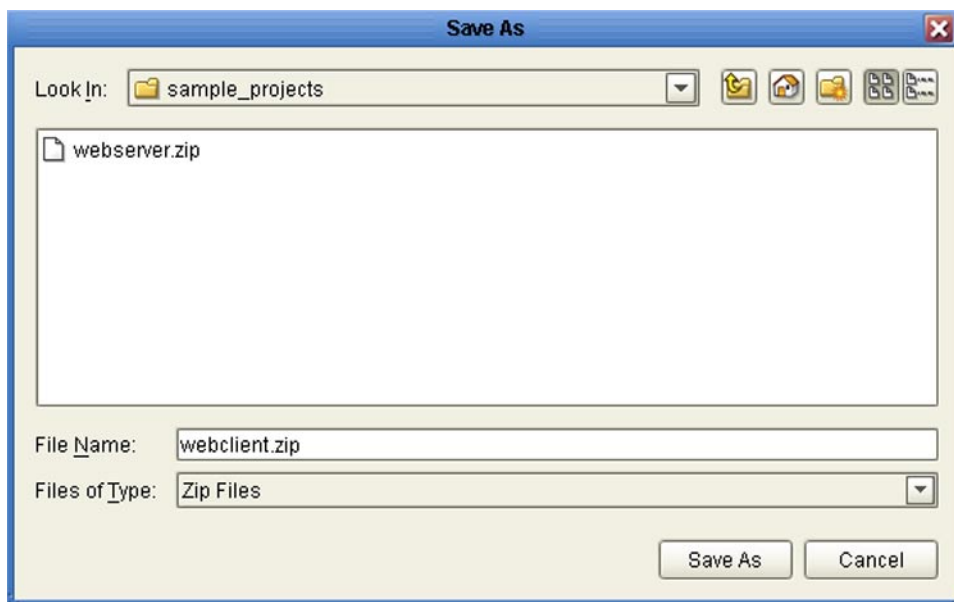
- 2 Highlight the desired Project(s) or Environment(s) in the displayed list, and transfer them to the *Selected Projects* or *Selected Environments* panel using the arrow buttons (see Figure 18).

**Figure 18** Export Manager Dialog (2)



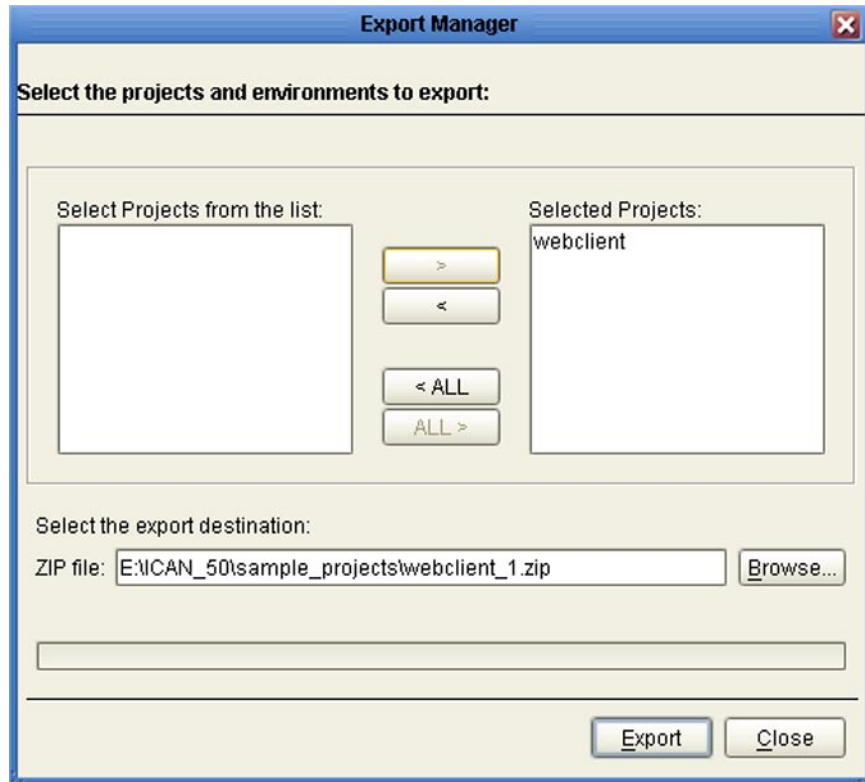
- 3 Click the **Browse** button to display the *Save As* dialog, as shown in Figure 19.

**Figure 19** Save As Dialog



- 4 Select the export destination and change the export file name, if desired.
- 5 Click **Save As** to enter the file name.

**Figure 20** Enter File Name Dialog (2)



- 6 Click **Export** to export the Project file (this process may take a few minutes).
- 7 The Export Status message box shown in Figure 21 appears after the file has been exported successfully.

**Figure 21** Export Status Message Box



- 8 Click **OK** to close the message box.

## Exporting Using the Command Line

You can also export a Project or Environment using the following command-line script.

Location of script file:

```
<Sun_JavaCAPS_install_dir>\repository\util\exportProject.bat  
(or exportProject.sh)
```

Command Syntax:

```
exportProject <username> <password> <exportfile> <projectname>  
            <environmentname>
```

where:

- **<exportfile>** is the name and path for the archive file to contain the Project and/or Environment you are exporting.
- **<projectname>** is the name of the Project you are exporting. If you are exporting an Environment only, leave this parameter as an empty string ("").
- **<environmentname>** is the name of the Environment you are exporting. If you are exporting a Project only, leave this parameter as an empty string ("").

**Important:** Project names should not include special characters such as a comma (,), apostrophe ('), or quotation ("). If any of these characters are used, you must escape them when using the importProject command-line utility.

To export a Project using the export script

- 1 Open a command prompt and change directory to:

```
<Sun_JavaCAPS_install_dir>\repository\util
```

- 2 To save the Project **myProject** to the file **c:\project4export.zip**, type:

```
exportProject <username> <password> c:\project4export.zip  
myProject ""
```

To export an Environment using the export script

- 1 Open a command prompt and change directory to:

```
<Sun_JavaCAPS_install_dir>\repository\util
```

- 2 To save the Environment **myEnvironment** to the file **c:\environment4export.zip**, type:

```
exportProject <username> <password> c:\environment4export.zip ""  
myEnvironment
```

To export a Project and an Environment using the export script

- 1 Open a command prompt and change directory to:

```
<Sun_JavaCAPS_install_dir>\repository\util
```

- 2 To save the Project **myProject** and Environment **myEnvironment** to the file **c:\projenv4export.zip**, type:

```
exportProject <username> <password> c:\projenv4export.zip  
myProject myEnvironment
```

### 4.3.2 Importing a Project or Environment

The import function allows you to import an external Project or Environment into the currently open Repository branch using Enterprise Designer. The Project or Environment to be imported must have been previously exported from the source Repository into an archive file (see [Exporting a Project or Environment](#) on page 62).

When importing a Project, note that:

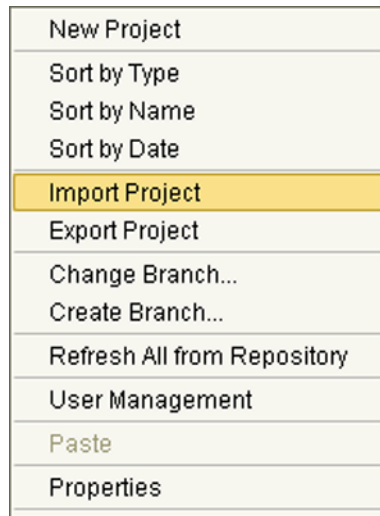
- Existing Projects are not affected by the imported Project.
- During import, if another Project having the same name exists in the target Repository and *all* of the following conditions are met, you are asked to confirm whether or not you want the existing Project to be overwritten. Otherwise, you will receive an error message and the existing file will not be overwritten.
  - ♦ The file you are importing has a single top-level Project and/or a single Environment.
  - ♦ None of the Project or Environment components contains any reference outside the hierarchy in which it resides.
  - ♦ The Project or Environment you are importing was previously exported from the source Repository using eGate Integrator 5.1.
- If you have not installed all of the necessary products (such as eWays) that the Project requires, you will *not* be able to import that Project and you will get an error message.
- References are validated during import.
- Project deployment objects are *not* imported, since they have references to both Project and Environment elements that are not required at the Project level.

## Importing Using Enterprise Designer

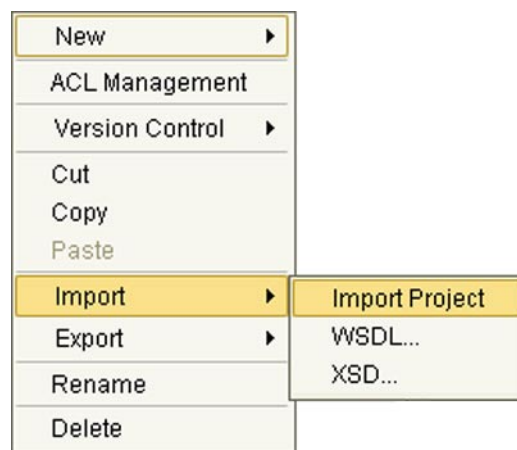
To import a Project using Enterprise Designer

- 1 Select **Import Project** from the appropriate context menu.
  - A For a top-level Project, use the Repository context menu (see Figure 22).
  - B For a Sub-Project, use the Project context menu (see Figure 23).

**Figure 22** Repository Context Menu - Import Project Option



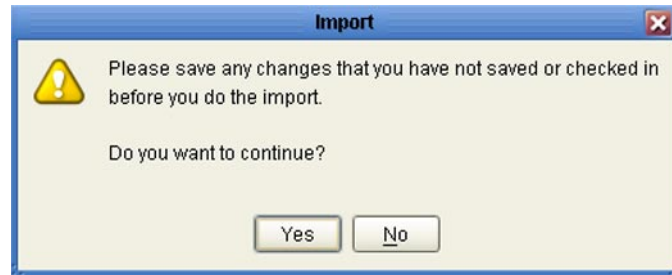
**Figure 23** Project Context Menu - Import Options



- 2 The message box shown in Figure 24 appears, prompting you to save your changes.

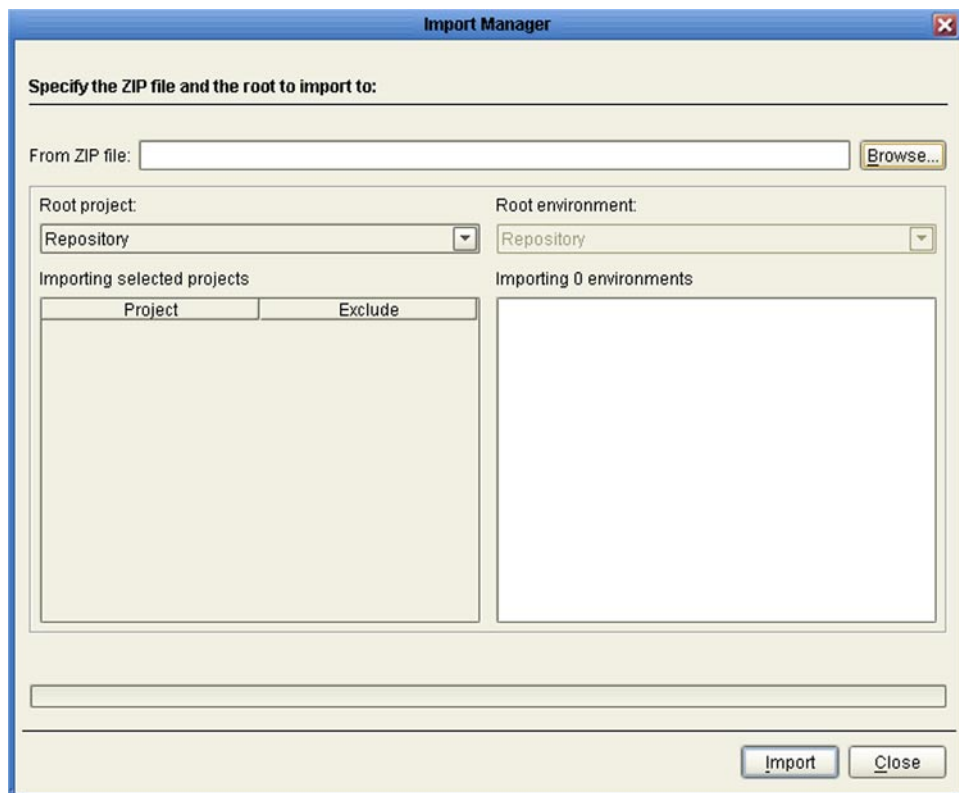


**Figure 24** Import Message Box



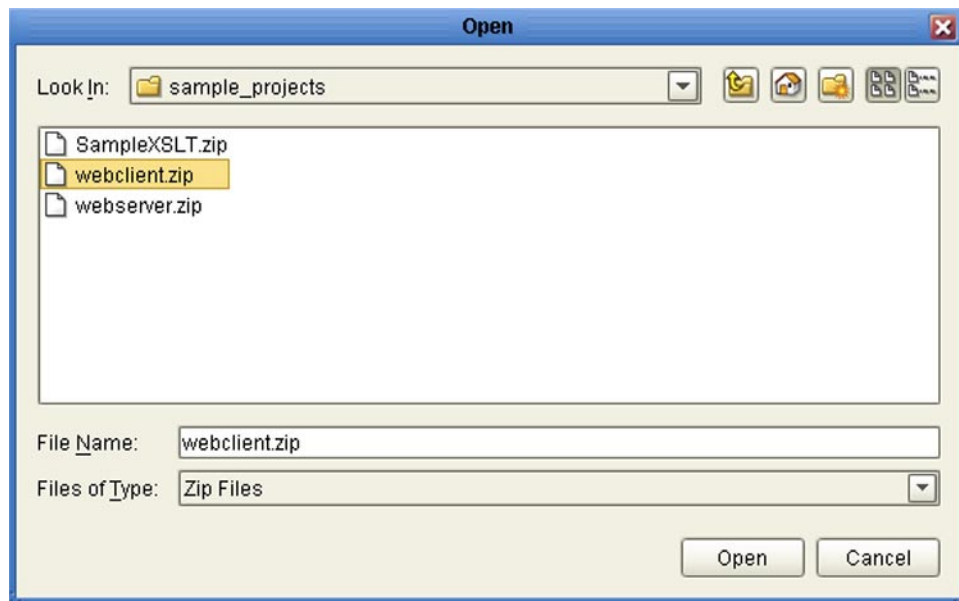
- A If you want to save your changes, but have not already done so, click **No**. Save your changes, and then re-select **Import**, as in step 1.
- B If you have saved any desired changes, click **Yes** to display the dialog shown in Figure 25.

**Figure 25** Import Manager Dialog (1)



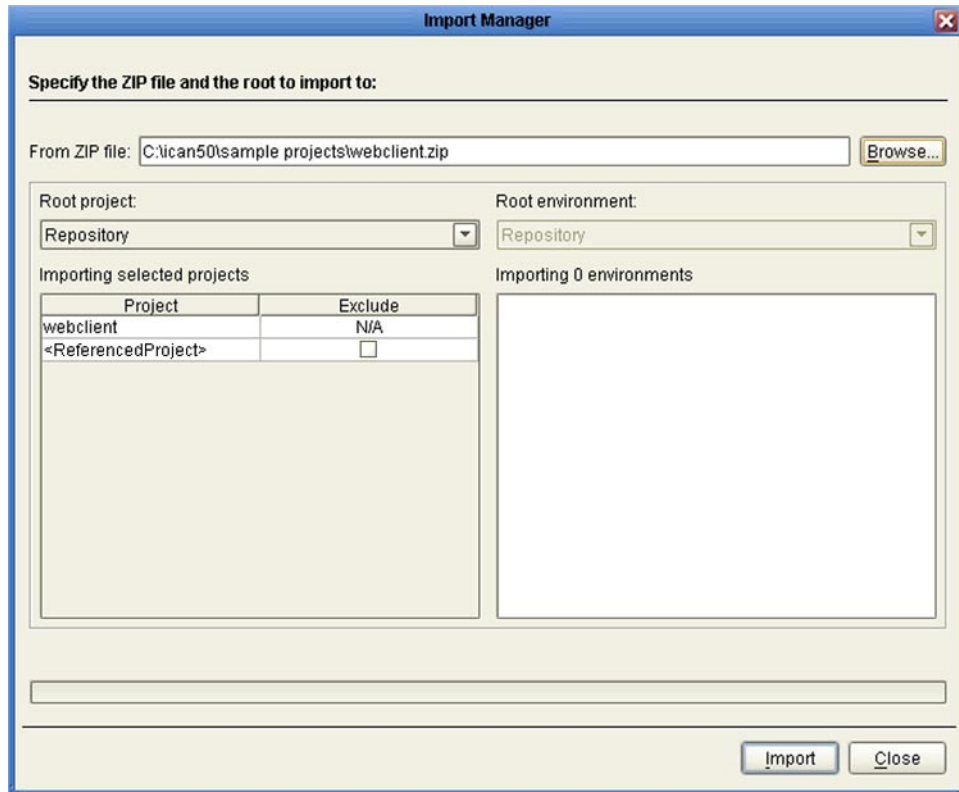
- 3 Click the **Browse** button to display the *Open File* dialog, as shown in Figure 26. If you browse to an Environment file, the *Root environment* field will be enabled.

**Figure 26** Open File Dialog



- 4 Locate and select the Project or Environment file that you want to import.
- 5 Click **Open** to import the file.
- 6 The Import Manager dialog appears as shown in Figure 27.

**Figure 27** Import Manager Dialog



**Note:** If the Project you are importing contains references to another Project, and the other Project already resides in your Repository, you have the option of excluding the referenced Project from the import by checking the box that appears in the Exclude column. The references will be retargeted to the Project existing in the Repository.

- 7 Click **Import** to import the file. There are three scenarios for what occurs next:
  - A There is no conflict with existing Project or Environment names, and the import process proceeds. Go to step 9.
  - B If a Project or Environment having the same name as the one you are importing already exists in the target Repository, and the conditions shown in step C are not met, you are presented with an error message and the import process aborts. Click **OK** to close the message box.
  - C If a Project or Environment having the same name as the one you are importing already exists in the target Repository, and *all three* conditions listed below are met, you are presented with the dialog shown in Figure 28. Proceed to step 8.
    - ♦ The file you are importing has a single top Project and/or a single Environment.
    - ♦ None of the Project or Environment components contains any reference outside the hierarchy in which it resides.
    - ♦ The Project or Environment you are importing was previously exported from the source Repository using eGate Integrator 5.1.

**Figure 28** Project Import Confirmation Dialog



- 8 The dialog shown in Figure 28 offers three options:
- A If you want to keep the Project or Environment currently in your Repository, click **No**.
  - B If you want to replace the Project or Environment currently in your Repository, along with all contained objects having the same names, click **Yes to All**.
  - C If you want to replace the Project or Environment currently in your Repository, but want to manually select which objects get replaced, click **Yes**. You will subsequently be presented with a series of confirmation dialogs for each of the like-named objects in the Project or Environment (see Figure 29 for an example).

**Figure 29** Create New Version Confirmation Dialog



- 9 The Import Status message box shown in Figure 30 appears after the file has been imported successfully.

**Figure 30** Import Status Message Box



- 10 Click **OK** to close the message box.
- 11 When you are finished importing files, click **Close** to close the Import Manager dialog. The Project Explorer will now automatically be refreshed from the Repository.

## Importing Using the Command Line

You can also import a Project or Environment using the following command-line script.

**Note:** *When importing an Environment, the Environment and all Projects deployed to it are imported together.*

### Location of script file:

```
<Sun_JavaCAPS_install_dir>\repository\util\importProject.bat (or  
importProject.sh)
```

### Command Syntax:

```
importProject <username> <password> <exportfile> <rootprojectpath>  
<KeepAll/ReplaceAll> <ExcludeList>
```

where:

- **<exportfile>** is the name and path of the archive file containing the Project or Environment you are importing.
- **<rootprojectpath>** is the location to which the Project or Environment is to be imported.
  - ♦ If the imported Project is to become a top-level Project under the Repository, specify this parameter as an empty string ("").
  - ♦ If the imported Project is to become a sub-Project under an existing Project, use the format **ProjectName/SubProjectName**.
- **<KeepAll/ReplaceAll>** is the option to use in case of a conflict, where an object of the same name already exists in the Repository or parent Project.
  - ♦ To keep all existing objects (except for those in the ExcludeList) as-is, use **KeepAll**.
  - ♦ To replace the Project and all contained objects (except for those in the ExcludeList) with like-named objects from the archive file, use **ReplaceAll**.
- **<ExcludeList>** is a list of all objects to be excluded from the **KeepAll** or **ReplaceAll** option. Specify the entire path for each object, enclose in quotes, and use the delimiter + between objects. If no objects are to be excluded, specify this parameter as an empty string ("").

### Conditions:

Remember that the **Keep / Replace / Exclude** functionality is available only if *all* of the following conditions are met; otherwise, the corresponding parameters have no effect.

- The file you are importing has a single top Project and/or a single Environment.
- None of the Project or Environment components contains any reference outside the hierarchy in which it resides.
- The Project or Environment you are importing was previously exported from the source Repository using eGate Integrator 5.1.

**Note:** When specifying the path names for environmental objects in Logical Hosts, **do not** include the Logical Host name in the path. Instead, show the object as being directly under the Environment, for example: “**Environment1/IntegrationSvr1**”

### To use the command-line import utility

- 1 Open a command prompt and change the directory to:

```
<Sun_JavaCAPS_install_dir>\repository\util
```

- 2 Type the command syntax specifying the parameters as necessary to achieve the desired import behavior (see the following examples).

To extract a Project contained in the file **c:\project4import.zip** and import it into the Repository as a top-level Project, keeping all like-named objects in the Repository as-is, type:

```
importProject <username> <password> c:\project4import.zip ""
KeepAll ""
```

To extract a Project contained in the file **c:\project4import.zip** and import it into the Repository as a sub-Project of **mainProject**, keeping all like-named objects in the Repository as-is, type:

```
importProject <username> <password> c:\project4import.zip
mainProject/subProject KeepAll ""
```

To extract a Project contained in the file **c:\project4import.zip** and import it into the Repository as a top-level Project, keeping only selected like-named objects in the Repository as-is, type:

```
importProject <username> <password> c:\project4import.zip ""
ReplaceAll "KeepObject1+KeepObject2+...+KeepObjectN"
```

To extract a Project contained in the file **c:\project4import.zip** and import it into the Repository as a top-level Project, replacing only selected like-named objects in the Repository as-is, type:

```
importProject <username> <password> c:\project4import.zip ""
KeepAll "ReplaceObject1+ReplaceObject2+...+ReplaceObjectN"
```

To extract a Project contained in the file **c:\project4import.zip** and import it into the Repository as a top-level Project, replacing all like-named objects in the Repository, type:

```
importProject <username> <password> c:\project4import.zip ""
ReplaceAll ""
```

- 3 Select **Enter** to run the utility.

**Important:** Project names should not include special characters such as a comma (,), apostrophe (’), or quotation (“). If any of these characters are used, you must escape them when using the `importProject` command-line utility.

## 4.4 Using Version Control Features

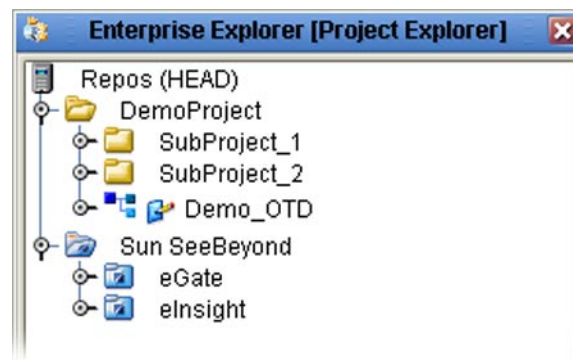
Extensive version control features have been built into Enterprise Designer for managing multiple versions of Projects and components. In addition to the global features described in this section, features specific to Deployment Profiles are described in [Version Control](#) on page 137.

### 4.4.1 Managing Project Versions

Repository branches enable you to isolate different versions of a specific Project from each other. This allows you to maintain a stable version deployed to your production environment, for example, while working on an updated version in your development environment.

When you install the Java™ Composite Application Platform Suite, the Repository has a main branch called HEAD, as shown in Figure 31.

**Figure 31** Repository HEAD Branch



Typically, you develop a Project in the HEAD branch. When you are ready to deploy the Project to your production environment, you create a separate branch to contain that version of the Project. You then continue to develop the next version of the Project in the HEAD branch.

When you modify a component in any branch, the changes are confined to that branch; other branches are not affected. You cannot copy and paste components between branches.

### Creating Branches

If you have administrator privileges, you can create a new branch using the Repository context menu (see [Using the Repository Context Menu](#) on page 59). Refer to the *Sun SeeBeyond eGate™ Integrator System Administration Guide* for more information.



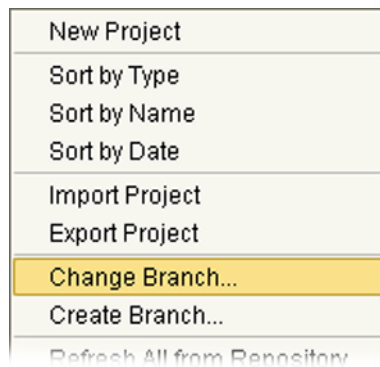
## Changing Branches

Enterprise Designer displays one branch at a time. You can change the currently displayed branch using the Repository context menu.

To change the Repository branch view

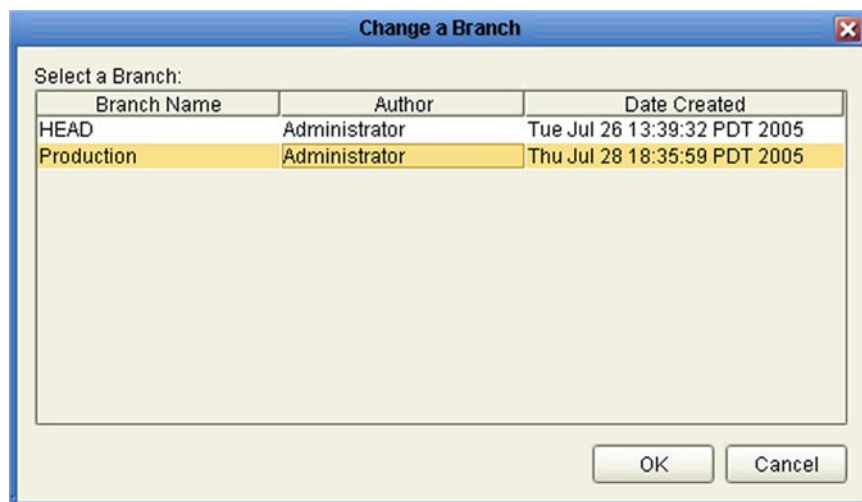
- 1 Display the Repository context menu (see Figure 32) by right-clicking the Repository in Project Explorer.

**Figure 32** Repository Context Menu - Change Branch



- 2 Click the **Change Branch** option, which displays the *Change Branch* dialog (see Figure 33).

**Figure 33** Change Branch Dialog



- 3 Select the desired branch and click **OK**.

**Note:** Once you have changed branches, you should perform some operation (such as checking some component out, then back in) to establish your presence in the new branch.

## 4.4.2 Managing Component Versions

Version control allows you to maintain multiple versions of selected Project or Environment components within a particular branch of the Repository. The version history of each component is recorded to a log file, and can be viewed by means of a menu option (see [Viewing a Component's Version History](#) on page 84).

**Important:** *More than one person concurrently using the same user ID will circumvent this version control system, and one person's work can be overwritten by another. You should ensure that all personnel using Enterprise Designer use unique IDs.*

### Checked-In State

When a component is checked in to the version control system, it is locked against modification until being checked out, and a lock is displayed in the component's icon in the Enterprise Explorer. Figure 34 shows the lock, using the OTD icon as an example. See [Checking a Component In](#) on page 80, [Checking a Component In Without Revisions](#) on page 89, and [Checking In a Previous Version as the Latest Version](#) on page 90.

**Figure 34** Checked In Icon (OTD Example)



### Checked-Out State

When the latest version of a component is checked out from the version control system, it is locked against another user checking it out. A writing pad icon (see Figure 35) is displayed next to the component's icon in the Enterprise Explorer, indicating that it is checked out. See [Checking a Component Out](#) on page 82.

**Figure 35** Checked Out Icon



### Retrieved State

When any version of a component is retrieved from the version history dialog, it is *not* locked against another user checking it out or retrieving it. A combined writing pad/warning icon (see Figure 36) is displayed next to the component's icon in the Enterprise Explorer, indicating that it is in your workspace — but warning you that it is not locked in any way. See [Retrieving a Component to Your Workspace](#) on page 87 and [Checking In a Previous Version as the Latest Version](#) on page 90.

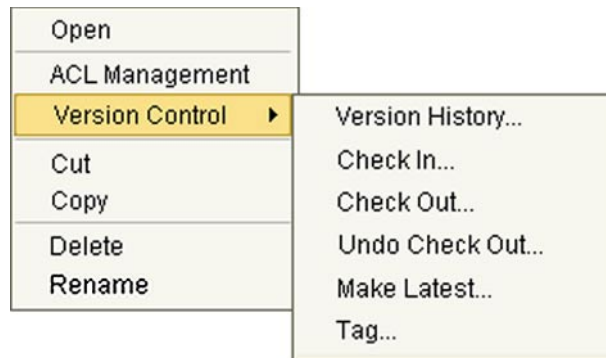
**Figure 36** Retrieved Icon



## Accessing Component Version Control Features

You can access the version control features for any component from the component's context menu, as shown in Figure 37.

**Figure 37** Connectivity Map Context Menu



**Table 14** Connectivity Map Menu Options - Version Control

Option	Function
Version History	Presents a dialog with which you can track the version history for the selected component. See <a href="#">Viewing a Component's Version History</a> on page 84 for more information.
Check In	Presents a dialog, with which you can check in a new version of the selected component. See <a href="#">Checking a Component In</a> on page 80 for more details.
Check Out	Presents a dialog with which you can check out the current version of the selected component. See <a href="#">Checking a Component Out</a> on page 82 for more information.
Undo Check Out	Presents a dialog with which you can undo the check-out of the selected component. See <a href="#">Checking a Component In Without Revisions</a> on page 89 for more information.
Make Latest	Allows you to check in the version of the selected component that was retrieved to your workspace, making it the latest version. See <a href="#">Checking In a Previous Version as the Latest Version</a> on page 90.
Tag	Presents a dialog with which you can specify a tag to attach to the selected component.

## Checking a Component In

Once you have created and configured a component for the first time, or created a revised version of an existing component, you must check that component in to save it to the common area of the Repository and release your lock on the object.

You can check in components either individually or collectively; for example, as grouped in a Project or Subproject. You have the option of including components in a Subproject with the components in the parent Project for checking in (they are excluded by default), and you can exclude individual components within a collection from being checked in.

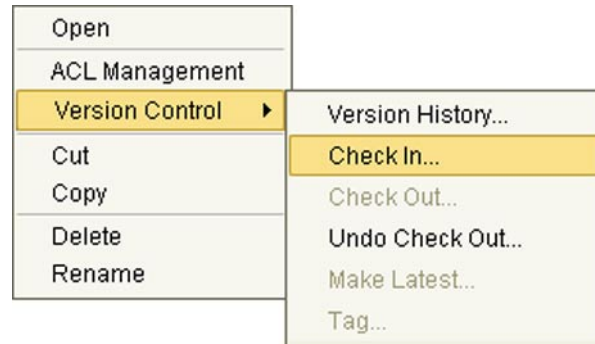
You can also check in multiple components by holding the **Ctrl** key while you select the components.

**Note:** Although you can select multiple Projects (and they will appear highlighted in Project Explorer), only the components in the first Project you select will be shown in the dialog box. To check in components from multiple Projects at the same time, you must select the components themselves, not the parent Projects.

### To check in a new version of a Project or Environment component

- 1 In the Enterprise Explorer, select the component(s) and right-click to display the context menu.

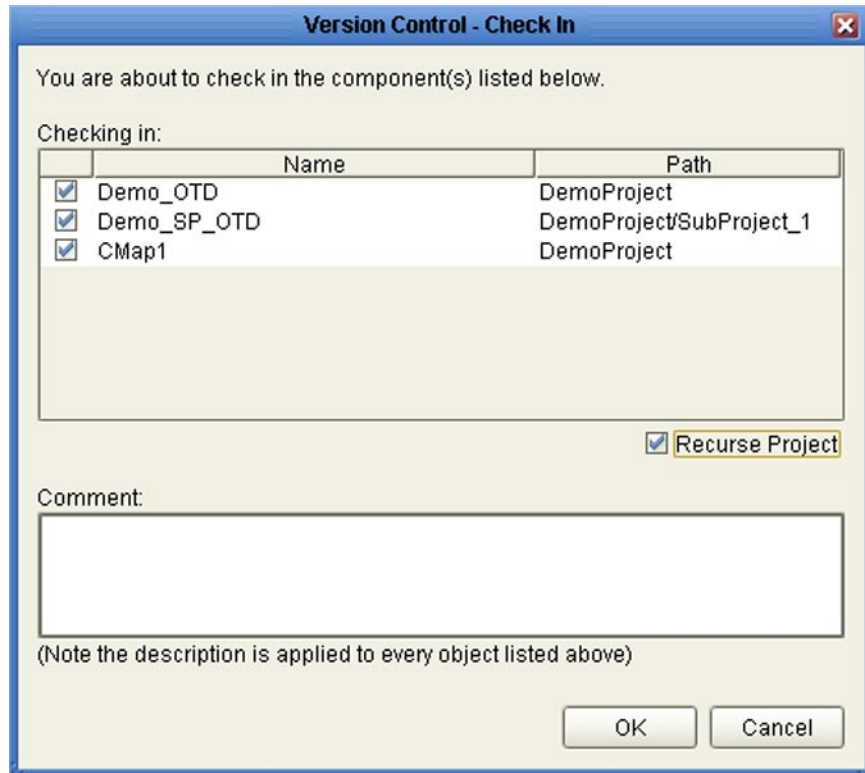
**Figure 38** Example Context Menu - Check In Option



- 2 Select **Version Control > Check In**, as shown in Figure 38, to display the dialog shown in Figure 39. All contained objects eligible for checking in are shown selected by default. Any objects that are not eligible for checking in are shown dimmed.

**Note:** If you have selected a Project, and sub-Projects are present, you must select **Recurse Project** to add the components contained in the sub-Projects to the list. Otherwise, those components will be excluded from check-in.

**Figure 39** Version Control - Check In Dialog



- 3 All components eligible for checking in are shown selected by default. Any components that are not eligible for checking out are shown dimmed.

**Note:** *If you have selected a Project, and Subprojects are present, you must select **Recurse Project** to add the components contained in the Subprojects to the list. Otherwise, those components will be excluded from being checked in.*

- 4 Deselect any components that you want to exclude from check-in.
- 5 Type in a description of the changes in the new version.
- 6 Click **OK** to check the new versions in. The version numbers are automatically incremented.

## Checking a Component Out

You can check components out to your workspace for editing by using the procedure described in this section. While the component is checked out from the version control system, it is locked against another user checking it out.

You can check out components either individually or collectively; for example, as grouped in a Project or Subproject. You have the option of including components in a Subproject with the components in the parent Project for checkout (they are excluded by default), and you can exclude individual components within a collection from being checked out.

You can also check out multiple components by holding the **Ctrl** key while you select the components.

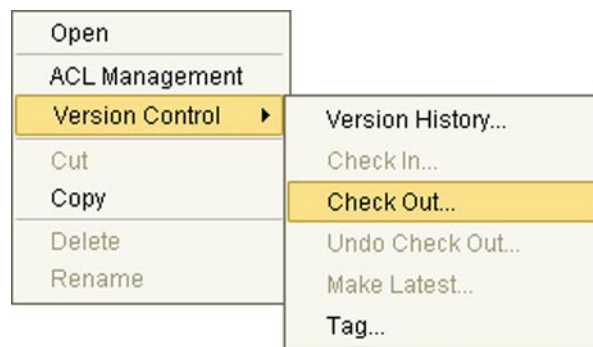
**Note:** Although you can select multiple Projects (and they will appear highlighted in Project Explorer), only the components in the first Project you select will be shown in the dialog box. To check out components from multiple Projects at the same time, you must select the components themselves, not the parent Projects.

**Note:** Only one user can have a file checked out for editing at a time. If another user attempts to check out the same file, they will receive a message indicating that the file is currently checked out (see the Sun SeeBeyond eGate™ Integrator System Administration Guide for additional information).

### To check out the latest version of a component or collection of components

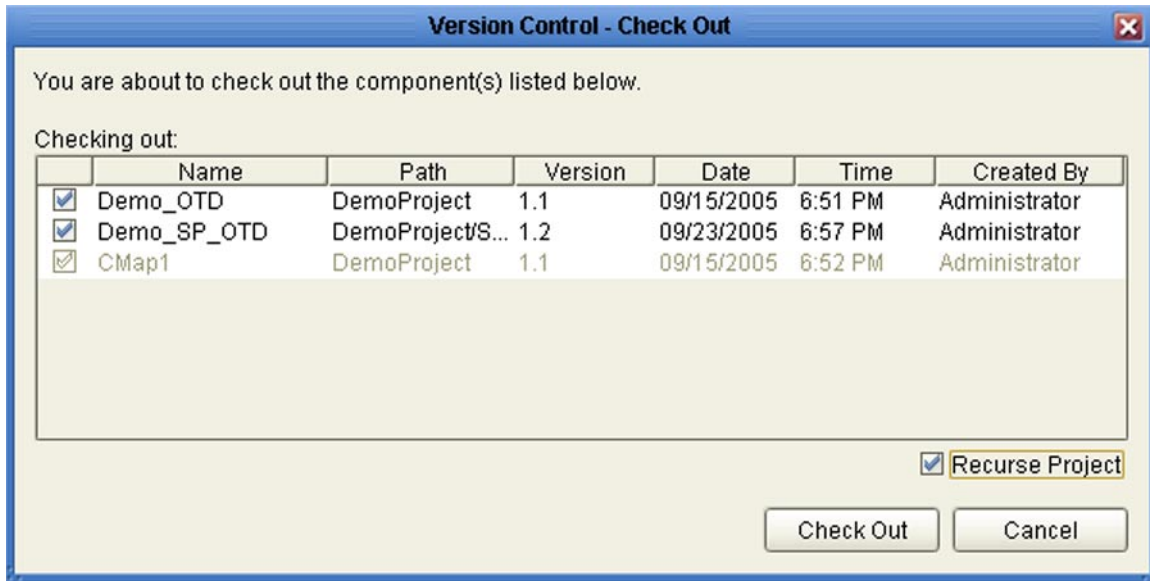
- 1 In the Enterprise Explorer, select the Project or component and right-click to display its context menu.

**Figure 40** Example Context Menu - Check Out Option



- 2 Select **Version Control > Check Out**, as shown in Figure 40, to display the dialog shown in Figure 41.

**Figure 41** Version Control - Check Out Dialog



- 3 All components eligible for checking out are shown selected by default. Any components that are not eligible for checking out are shown dimmed. (In the figure, CMap1 is already checked out, so is not eligible for checking out again.)

**Note:** If you have selected a Project, and Subprojects are present, you must select **Recurse Project** to add the components contained in the Subprojects to the list. Otherwise, those components will be excluded from being checked out.

- 4 Deselect any components that you want to exclude from checkout.
- 5 Click **Check Out** to check the component(s) out.
- 6 Click the **Save** or **Save All** icon to place them in your Repository workspace.

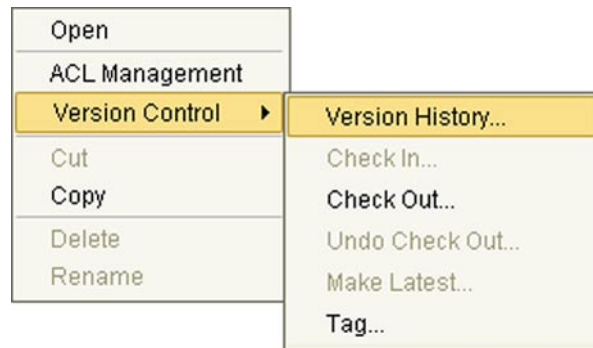
## Viewing a Component's Version History

**Note:** Although you can select multiple components (and they will appear highlighted in Project Explorer), only the first component you select will be shown in the dialog box. Version histories must be examined one component at a time.

To view the version history for a Project or Environment component

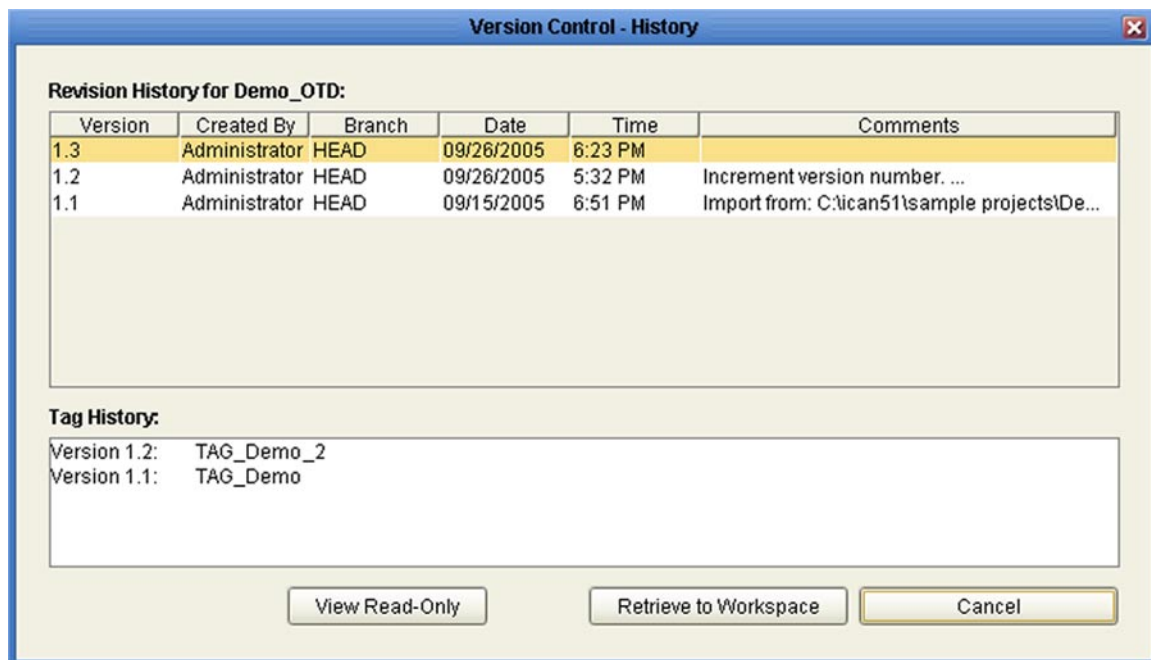
- 1 In the Enterprise Explorer, select the component and right-click to display its context menu.

**Figure 42** Example Context Menu - Version History Option



- 2 Select **Version Control > Version History** to display the dialog shown in Figure 43. If the component has been tagged, the tags are shown in a separate *Tag History* pane.

**Figure 43** Version Control - History Dialog





- 3 Double-click within the *Comments* column to display the full text of the comment.
- 4 Click **Cancel** to close the box.

**Note:** *If a version is checked out to any user's workspace, or retrieved to your workspace, the appropriate icon also appears in the Version column.*

**Note:** *The version history for a component that has been **cut** and pasted is preserved, since there can be only one instance of it. The version history for a component that has been **copied** and pasted is **not** preserved, since there can be many instances of it; the version number for each pasted instance is reset.*

## Viewing a Read-Only Version of a Component

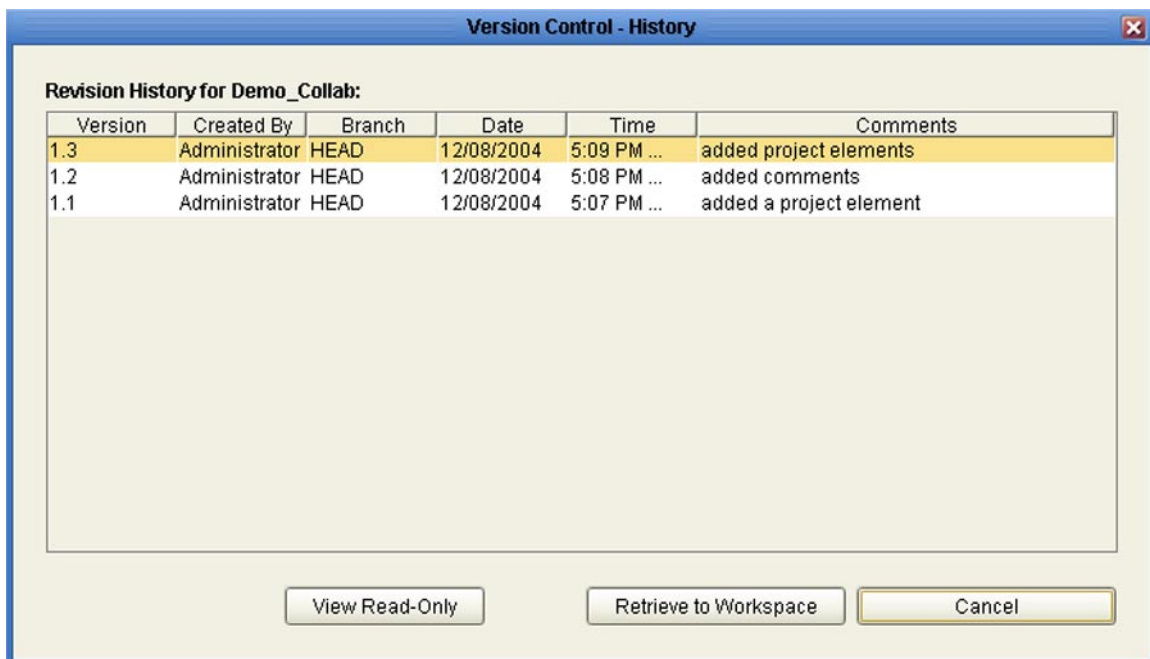
You can view a read-only instance of either the current or a previous version of a Java-based Collaboration or a User-Defined OTD by selecting the version from the Version History dialog box. The component must be checked in, and viewing as read-only does *not* lock the file from being checked out or retrieved by other users for editing. The version being viewed is opened as a separate instance in the appropriate editor, and can simply be closed when you are finished viewing it.

**Note:** *If you have a version of the same component in your workspace, it will not be affected by viewing another version in read-only mode.*

To view a read-only version of a Java-based Collaboration or User-Defined OTD

- 1 In the Enterprise Explorer, select the component and right-click to display its context menu.
- 2 Select **Version Control > Version History** (see [Viewing a Component's Version History](#) on page 84) to display the dialog shown in Figure 45.

**Figure 44** Version Control - History Dialog



- 3 Select the version you want to view and click **View Read-Only**.
- 4 The selected version will be displayed as a separate editor instance.
- 5 When you are finished viewing the read-only version, right-click the tab in the docking tray at the bottom of the editor and select **Close**.

## Retrieving a Component to Your Workspace

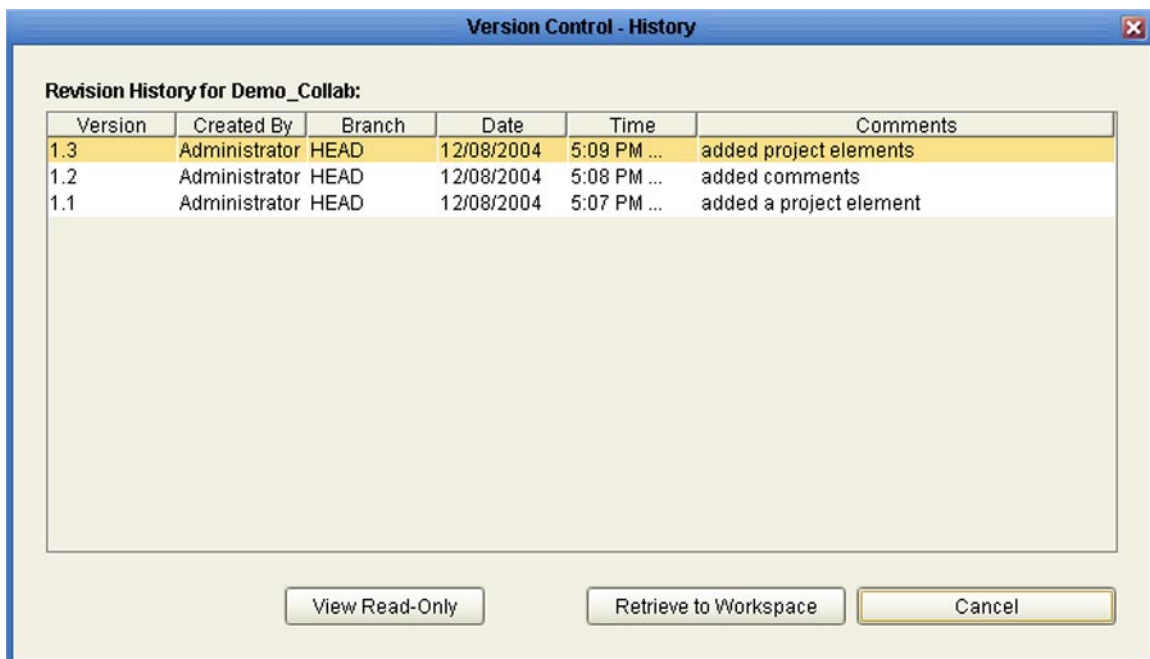
You can retrieve either the current or a previous version of any component by retrieving it from the Version History dialog box. Retrieving does *not* lock the file from being checked out or retrieved by other users for editing. To check a retrieved version back in as the latest version, you must use the **Make Latest** option described in [Checking In a Previous Version as the Latest Version](#) on page 90.

**Note:** *Although you can select multiple components (and they will appear highlighted in Project Explorer), only the first component you select will be shown in the dialog box. Version histories must be examined one component at a time.*

### To retrieve a previous version of a Project component

- 1 In the Enterprise Explorer, select a component and right-click to display its context menu.
- 2 Select **Version Control > Version History** (see [Viewing a Component's Version History](#) on page 84) to display the dialog shown in Figure 45.

**Figure 45** Version Control - History Dialog



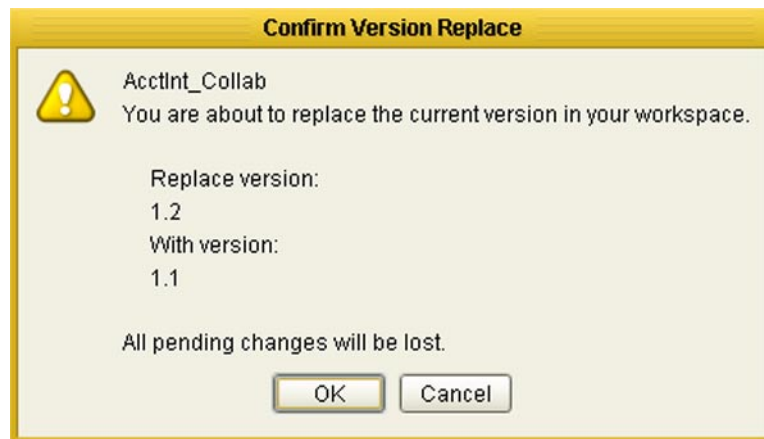
- 3 Select the version you want to retrieve and click **Retrieve to Workspace**.
  - A If you are attempting to retrieve the *latest* version of the component, you will be presented with the dialog shown in Figure 46.

**Figure 46** Access File Dialog



- ♦ **Check Out for Edit** copies the file to your workspace and locks it — the file becomes read-only to other users. This is the same mechanism as described in [Checking a Component Out](#) on page 82. To check the latest version out from the dialog, you must select this option and click **OK**. You will then be presented with the dialog shown previously (Figure 41).
  - ♦ **Retrieve to Workspace** copies the file to your workspace, but does not prevent it from being checked out or retrieved by other users. This is the default setting for the dialog; simply click **OK**.
- B** If you are attempting to retrieve a *previous* version of the component, you will be presented with the dialog shown in Figure 47. Clicking **OK** will overwrite any other version you have retrieved to your workspace, or replace the currently checked-in version in your workspace only — other users will be unaffected.

**Figure 47** Confirm Version Replace Dialog



**Note:** If you have the latest version of the component checked out to your workspace, the **Checked Out** icon will appear in the Version column of the Version History dialog and the **Retrieve to Workspace** button will be disabled. You must check the latest version back in to version control before you can retrieve any version.

## Checking a Component In Without Revisions

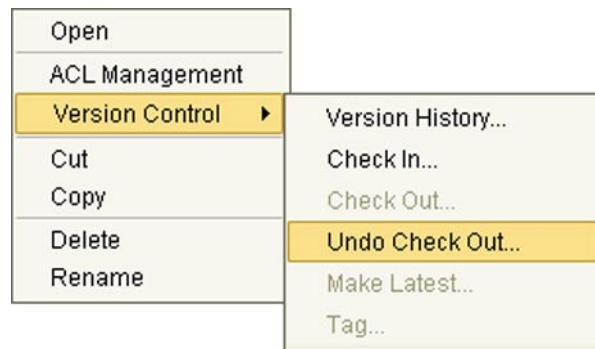
When you have checked out any version of a component and want to check it back in without revisions or incrementing the version number, you can simply cancel the check-out by using the following procedure. You can also check in multiple components by holding the **Ctrl** key while you select the components.

**Note:** This option is not available at the Project or Subproject level. You must perform the Undo operation on the individual components.

### To check in a Project or Environment component without revisions

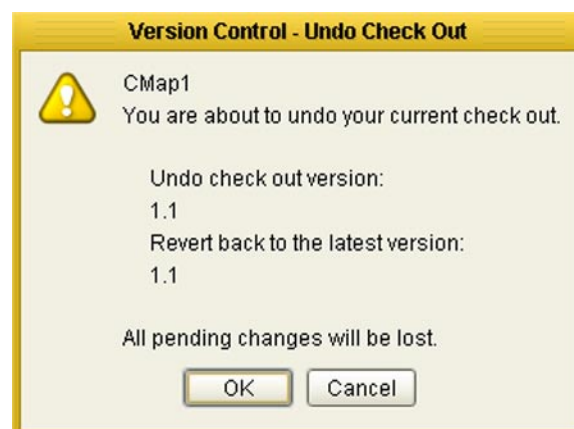
- 1 In the Enterprise Explorer, select the component and right-click to display its context menu.

**Figure 48** Example Context Menu - Undo Check Out Option



- 2 Select **Version Control > Undo Check Out**, as shown in Figure 48, to display the *Version Control - Undo Check Out* dialog shown in Figure 49.

**Figure 49** Version Control - Undo Check Out Dialog



- 3 Click **OK** to check the currently checked-out version back in.

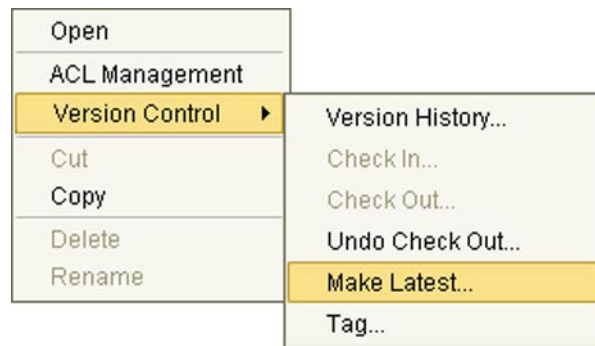
## Checking In a Previous Version as the Latest Version

If you have retrieved a previous version of a component to your workspace, you can check it in to the version control system as the latest version by selecting the *Make Latest* option.

To make a previous version of a component become the latest version

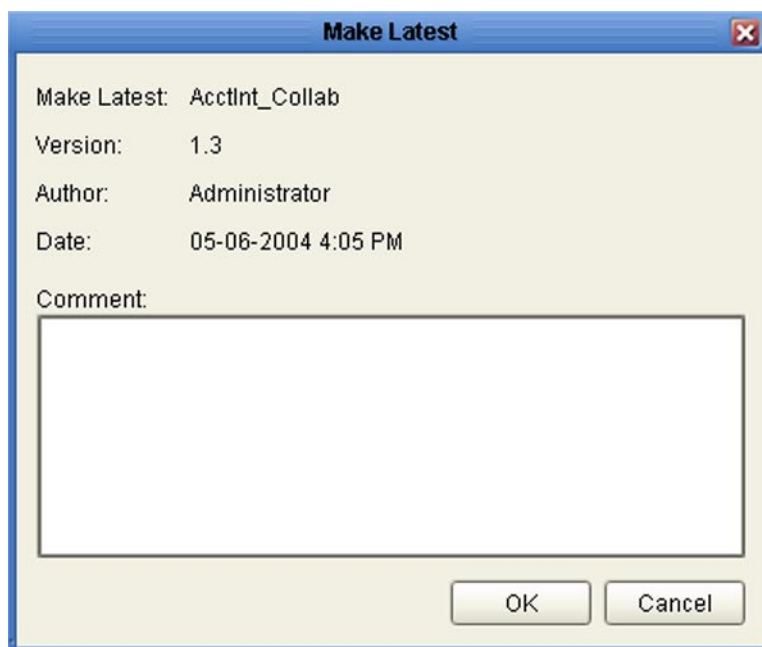
- 1 In the Enterprise Explorer, select the component and right-click to display its context menu.

**Figure 50** Example Context Menu - Make Latest Option



- 2 Click **Version Control > Make Latest** to display a confirmation dialog.
  - A If the latest checked-in version of the component has not changed since you retrieved the previous version, you will see the dialog shown in Figure 47.

**Figure 51** Make Latest Dialog



Type in a description of the changes in this version and click **OK**. The version in your workspace will be checked in as the latest version of the component.

- B** If the latest checked-in version of the component is different from the one that was current when you retrieved the previous version, you will first see the dialog shown in Figure 52.

**Figure 52** Confirm Latest Version Override Dialog



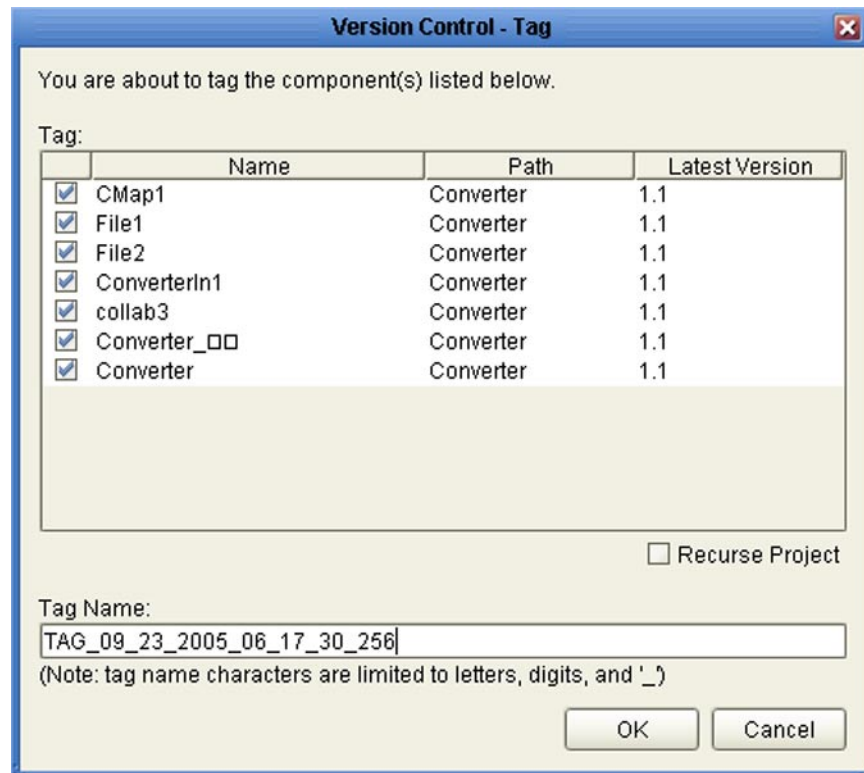
If you are sure you want to replace the current latest version, click **OK** to display the dialog shown previously (Figure 51). Type in a description of the changes in this version and click **OK**. The version in your workspace will be checked in as the latest version of the component.

**Important:** *This situation can occur if another user has made changes to the latest version — you must use caution when checking in your version, since the other user's changes will be superseded.*

## Tagging Components

You can tag components either individually or as collections so that you may retrieve them for deployment by specifying the tag name. When you select a component such as a Project for tagging, all components contained in the Project that are eligible for tagging are automatically included in a list (see Figure 53). You can then deselect individual components from the list.

**Figure 53** Version Control Tag Dialog



If a Project contains sub-Projects, you can add the components contained in them to the list by selecting the **Recurse Project** box (the sub-Projects themselves are not shown in the list). Only components that are *checked in* can be tagged; components that are *checked out* will be shown in the list, but will be dimmed and deselected.

**Note:** Previous versions of components can be tagged by retrieving the previous version to your workspace (see [Retrieving a Component to Your Workspace](#) on page 87).

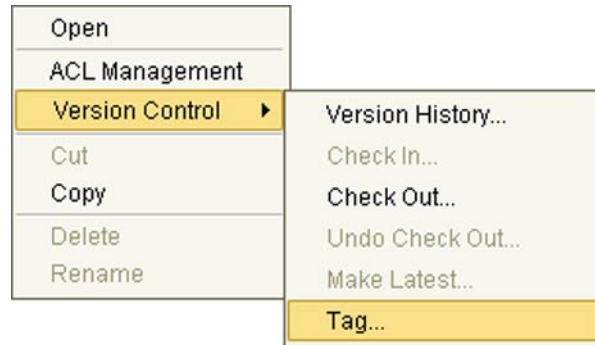
The default tag name is based on the date and time. Generally, you will want to create your own name for the tag.

### To tag a component or collection of components

- 1 In the Enterprise Explorer, select the component and right-click to display its context menu.



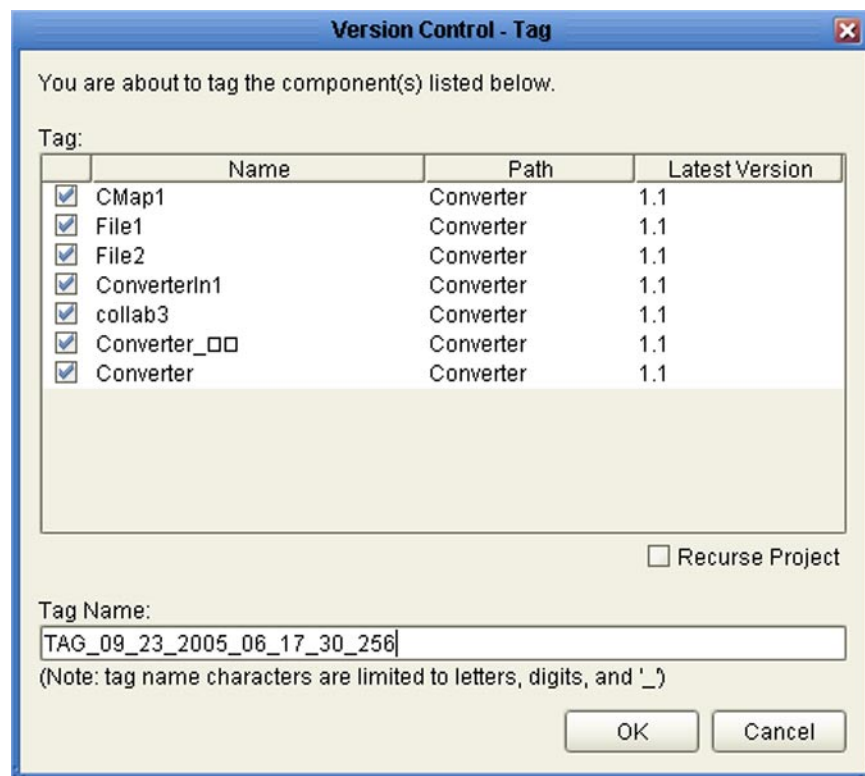
**Figure 54** Example Context Menu - Tag Option



- 2 Click **Version Control > Tag** to display the dialog shown in Figure 55.

**Note:** *If sub-Projects are present, you must select **Recurse Project** to add the components contained in the sub-Projects to the list. Otherwise, those components will be excluded from tagging.*

**Figure 55** Version Control Tag Dialog



- 3 Deselect any components that you want to exclude from tagging with this collection.
- 4 Create a tag name for the collection; otherwise, the default name will be used.
- 5 Click **OK** to create the tag.

The tag will now be shown in a *Tag History* list in the respective Version History dialogs for the selected components. Tagged components can be selected for deployment when you create a Deployment Profile (see [Creating a Deployment Profile](#) on page 132).

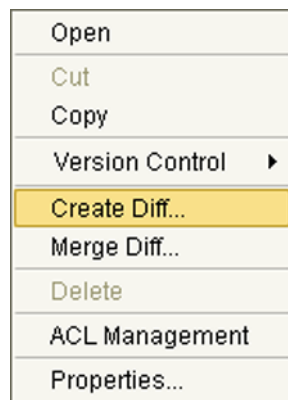
## Merging Changes from Another Version

A given version of a Collaboration Definition can be updated to a different version of the same Collaboration Definition by using the *Automerge* feature. Two versions of any Collaboration Definition can be saved to a difference (.sdf) file, which is an archive file containing the source code for the two versions. This file can be merged with a copy of the *Previous Version* of a given Collaboration Definition to produce a copy of the *Current Version* (refer to Figure 57).

To create a Difference (Diff) file

- 1 In Project Explorer, right-click the desired Collaboration Definition to display its context menu.
- 2 Select **Create Diff**, as shown in Figure 56, to display the dialog shown in Figure 57.

**Figure 56** Collaboration Definition Context Menu - Create Diff Option



**Figure 57** Version Control - Create Diff Dialog



- 3 Select the two versions of the Collaboration Definition that you want to differentiate.
- 4 Click **Generate Diff** to display the *Specify Name* dialog.
- 5 Enter a name for the difference file in the *File Name* box and select a directory in which to save the file (or use the defaults).

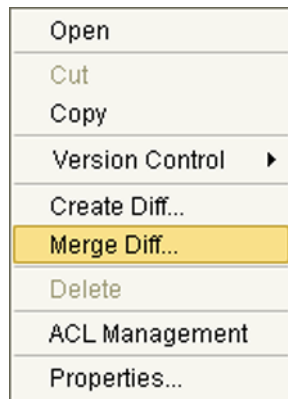
- 6 Click **Save** to save the file.

**Note:** You can open the *.sdf* file with a program such as WinZip to examine its contents, which consists of text files containing the Java code of the two versions. The code for the “Previous Version” is saved as *source*, and the code for the “Current Version” is saved as *modified*. Changes made to the *source* code are reflected in the *modified* code.

#### To merge a Difference (Diff) file into a Collaboration Definition

- 1 In Project Explorer, right-click the desired Collaboration Definition to display its context menu.

**Figure 58** Collaboration Definition Context Menu - Merge Diff Option



- 2 Select **Merge Diff** (see Figure 58), to display the dialog shown in Figure 59.

**Figure 59** Version Control - Merge Changes Dialog



- 3 Click the **Ellipsis (...)** button to display the *Specify Name* dialog.
- 4 Locate and select the difference (*.sdf*) file you want to merge into the selected Collaboration Definition.
- 5 Click **Open** add the file to the dialog shown in Figure 59.
- 6 Click **Merge** to execute the merger.
- 7 Check the resulting code and resolve any conflicts resulting from the merger. The differences between the *source* and *modified* code appear between markers shown as <<<<< and >>>>>. You must choose which code you want to keep, and remove the markers.

## 8 Commit and Save the merged Collaboration Definition.

**Important:** *If you are merging code that contains references to third-party classes or methods, you must import these files into the destination Project before merging the code.*

## Command-line Utilities

If you encounter problems with the version control system, there are two command-line utilities — a Repository version control utility and a workspace cleanup script — that can be run by personnel with Administrator privileges. These utilities should be used as a last resort, and with the utmost caution. See the *Sun SeeBeyond eGate™ Integrator System Administration Guide* for information.

## 4.5 Moving and Replicating Repository Objects

Project Explorer contains features that allow you to move and replicate Project-related Repository objects using commands from their respective context menus. These context menus are described in detail in their related chapters and sections.

### 4.5.1 Copying and Pasting

#### Projects

You can copy individual Projects from a Repository or parent Project using the **Copy** command from the Project's context menu. The components in the Project being copied may be either *checked in* or *checked out*; however, they are always pasted into the target location as being *checked in*. The original Project remains unchanged in the source location when you paste instances of it to other locations.

You can paste the copied Project into any Project (including itself) as a Subproject using the **Paste** command from its context menu. If a Project having the same name already exists in the target location, the paste proceeds normally and the name of the Project is appended with “\_#” (where # is a number, beginning with 1 and incrementing with each paste). You can paste a copied Project multiple times, to multiple locations.

All objects originally contained in the copied Project, including Subprojects and their components, are included when you paste it to a new location. All objects within the pasted instance are treated as being newly created, and all previously-existing version histories and ACL settings are discarded. All references from within the Project to objects external to the Project, however, are retained.

All references from objects external to the Project to objects within the Project remain unchanged, still referencing the objects in their original location.

#### Components

You can copy individual components from a Project tree using the **Copy** command from the component's context menu, and then paste them into other Projects using the **Paste** command. You can also select multiple components from multiple Projects. The components may be either *checked in* or *checked out*, and you can copy a component even when another user has the component checked out. Copying leaves the component in the source location unchanged when you paste instances of it to other locations.

If a component having the same name already exists in the target location, the paste proceeds normally and the name of the component is appended with “\_#” (where # is a number, beginning with 1 and incrementing with each paste). You can paste a copied component multiple times, to multiple locations.

When you paste a copied component, the pasted instance is treated as being newly created, and the previously-existing version history and ACL settings are discarded. All references to other objects, however, are retained.

All references from external objects to a copied component remain unchanged, still referencing the source component in its original location.

## 4.5.2 Cutting and Pasting

### Projects

You can cut individual Projects from a Repository or parent Project using the **Cut** command from the Project's context menu. All components in the Project must be *checked in* before the Project can be cut. The Project is removed from the source location when you paste it to another location.

The cut Project can be pasted into another Project as a Subproject using the **Paste** command from the Project's context menu. If a Project having the same name already exists in the target location, the paste operation is suspended and an error dialog appears. You can paste a cut Project only once.

Pasting a cut Project includes all objects originally contained in the Project, including Subprojects and their components. All references from within the Project to objects external to the Project are retained. All previously-existing version histories and ACL settings of the individual components are also retained. Retrieving a previous version of a component retrieves it to the new location.

All references from objects external to the Project to objects included in the paste are updated to refer to the new location, since the Project no longer exists in the original location.

### Components

You can cut individual components from a Project tree using the **Cut** command from the component's context menu. The component is removed from the source location when you paste it to another location. If a component having the same name already exists in the target location, the paste operation will be suspended and an error dialog will appear. You can paste a cut component only once.

You must have the component *checked out* before you can cut the component, and all changes you have made to the component must be committed. Cutting is disabled for other users when you have the component checked out.

When pasting a component, only the component itself is pasted. Any other objects referenced by the component are excluded, but all references to those objects are retained.

References to the pasted component are updated to refer to the new location only if the referencing objects are checked out during the paste. If the referencing object is not checked out at the time of the paste, a warning message is displayed. You can override the warning and continue with the paste, if you prefer.

The component's pre-existing version history and ACL settings are retained when cutting and pasting. Retrieving a previous version of a component retrieves it to the new location and also to the location where the component was located when that version was current.

## 4.6 Analyzing Component Dependencies

The Impact Analyzer helps you determine how a change to one component of a Project or Environment might affect other components in that Project or Environment. It provides the following information regarding the selected component:

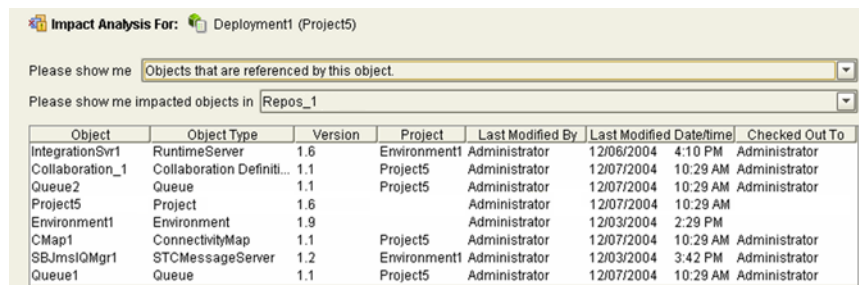
- Objects that have references to the selected component.
- Objects that are referenced by the selected component.

### 4.6.1 About Component Dependencies

All components reference the parent Project, and some components may reference objects other than Project or Environment components. Examples of typical dependencies are:

- **Deployment Profiles** generally reference all Project and Environment components, but are referenced only by the Project (see Figure 60 and Figure 61).

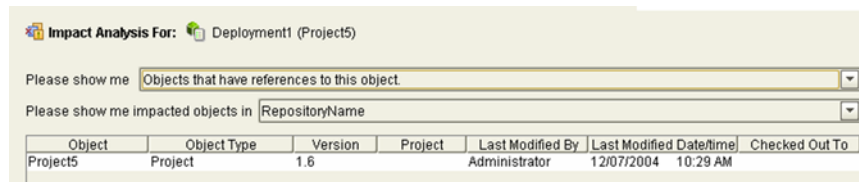
**Figure 60** Objects Referenced by Deployment Profile



The screenshot shows the 'Impact Analysis For: Deployment1 (Project5)' window. The 'Please show me' dropdown is set to 'Objects that are referenced by this object.' and the 'Please show me impacted objects in' dropdown is set to 'Repos\_1'. A table lists the following objects:

Object	Object Type	Version	Project	Last Modified By	Last Modified Date/time	Checked Out To
IntegrationSvr1	RuntimeServer	1.6	Environment1	Administrator	12/06/2004 4:10 PM	Administrator
Collaboration_1	Collaboration Definiti...	1.1	Project5	Administrator	12/07/2004 10:29 AM	Administrator
Queue2	Queue	1.1	Project5	Administrator	12/07/2004 10:29 AM	Administrator
Project5	Project	1.6		Administrator	12/07/2004 10:29 AM	
Environment1	Environment	1.9		Administrator	12/03/2004 2:29 PM	
CMap1	ConnectivityMap	1.1	Project5	Administrator	12/07/2004 10:29 AM	Administrator
SBJmslQMGr1	STCMessageServer	1.2	Environment1	Administrator	12/03/2004 3:42 PM	Administrator
Queue1	Queue	1.1	Project5	Administrator	12/07/2004 10:29 AM	Administrator

**Figure 61** Objects Referencing Deployment Profile



The screenshot shows the 'Impact Analysis For: Deployment1 (Project5)' window. The 'Please show me' dropdown is set to 'Objects that have references to this object.' and the 'Please show me impacted objects in' dropdown is set to 'RepositoryName'. A table lists the following objects:

Object	Object Type	Version	Project	Last Modified By	Last Modified Date/time	Checked Out To
Project5	Project	1.6		Administrator	12/07/2004 10:29 AM	

- **Connectivity Maps** generally reference all Project components, and are referenced by the Project and the Deployment Profile (see Figure 62 and Figure 63).

**Figure 62** Objects Referenced by Connectivity Map

Impact Analysis For: CMap1

Please show me

Please show me impacted objects in

Object	Object Type	Version	Project	Last Modified By	Last Modified Date/time	Checked Out To
Project5	Project	1.6		Administrator	12/07/2004 10:29 AM	
Queue2	Queue	1.1	Project5	Administrator	12/07/2004 10:29 AM	Administrator
Queue1	Queue	1.1	Project5	Administrator	12/07/2004 10:29 AM	Administrator
Collaboration_1	Collaboration Definiti...	1.1	Project5	Administrator	12/07/2004 10:29 AM	Administrator

**Figure 63** Objects Referencing Connectivity Map

Impact Analysis For: CMap1

Please show me

Please show me impacted objects in

Object	Object Type	Version	Project	Last Modified By	Last Modified Date/time	Checked Out To
Project5	Project	1.6		Administrator	12/07/2004 10:29 AM	
Deployment1	ProjectDeployment	1.2	Project5	Administrator	12/07/2004 10:29 AM	Administrator

- **Collaboration Definitions** generally reference an OTD and one or more files, and are referenced by the Project and Connectivity Map (see Figure 64 and Figure 65).

**Figure 64** Objects Referenced by Collaboration Definition

Impact Analysis For: Collaboration\_1

Please show me

Please show me impacted objects in

Object	Object Type	Version	Project	Last Modified By	Last Modified Date/time	Checked Out To
JMS	JMSObjectTypeDefini...	1.1	eGate	Administrator	12/03/2004 3:42 PM	Administrator
Project5	Project	1.6		Administrator	12/07/2004 10:29 AM	
jms.otd.api.jar	ProjectFile	1.1	eGate	Administrator	12/02/2004 11:29 AM	Administrator
jms.jar	ProjectFile	1.1	eGate	Administrator	12/02/2004 11:29 AM	Administrator

**Figure 65** Objects Referencing Collaboration Definition

Impact Analysis For: Collaboration\_1

Please show me

Please show me impacted objects in

Object	Object Type	Version	Project	Last Modified By	Last Modified Date/time	Checked Out To
Project5	Project	1.6		Administrator	12/07/2004 10:29 AM	
CMap1	ConnectivityMap	1.1	Project5	Administrator	12/07/2004 10:29 AM	Administrator

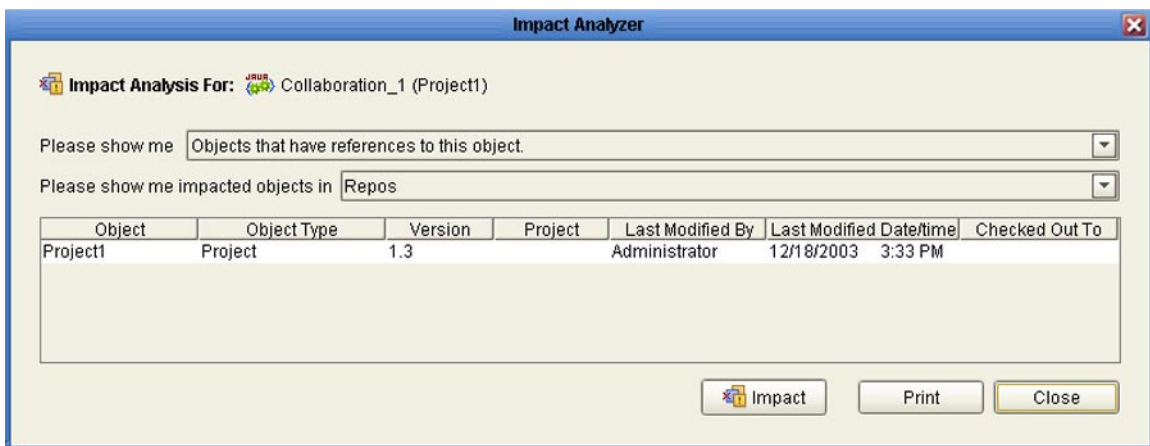


## 4.6.2 Using the Impact Analyzer



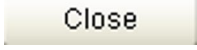
To perform an Impact Analysis

- 1 Select a component in either the Project Explorer or Environment Explorer.
- 2 Click the **Impact Analyzer** button, or select **Impact Analyzer** from the Tools menu, to display the *Impact Analyzer* dialog shown in Figure 66.
- 3 In the *Please show me* drop-down list, select objects you would like to view. You have the option of viewing either:
  - ♦ Objects that have references to this object.
  - ♦ Objects that are referenced by this object.
- 4 You can filter the number of listed objects using the *Please show me impacted objects in* drop-down list; by default, the entire Repository is selected.
- 5 Click **Impact** to display the list, according to your selections in the previous steps.
- 6 You can print the object list by clicking **Print** to display the Windows *Print* dialog.

**Figure 66** Impact Analyzer Dialog



**Table 15** Impact Analyzer Command Buttons

Button	Function
	Displays the list of objects you have selected, relative to the selected Project or Environment component.
	Presents the Windows <b>Print</b> dialog, which you can use to print the object list.
	Closes the <b>Impact Analyzer</b> dialog.

# Projects

This chapter describes the process of defining integration Projects, and the various components of a Project.

## What's in This Chapter

- [Overview](#) on page 102
- [About Project Explorer](#) on page 103
- [Project Components](#) on page 105
- [Creating a Project](#) on page 111
- [Services](#) on page 113
- [Message Destinations](#) on page 115
- [External Applications](#) on page 117
- [Web Service External Applications](#) on page 118
- [Schedulers](#) on page 119
- [Constants and Variables](#) on page 121
- [Connectivity Maps](#) on page 125
- [Deployment Profiles](#) on page 130

---

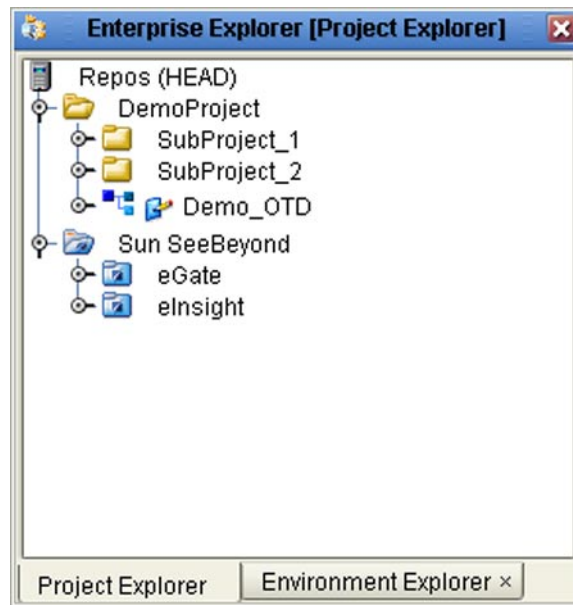
## 5.1 Overview

A Project represents a logical system designed to implement either all or part of a business integration task. The basic functionality of a Project is processing and routing messages, which it accomplishes by means of Collaborations. Projects are created using tools contained within Enterprise Designer, and are deployed to specific run-time Environments that represent the physical system supporting the software. Components developed for use in one Project can be used in another, and a Project can internally reference another Project.

## 5.2 About Project Explorer

Project Explorer deals with logical components, and includes folders and icons that represent the names and contents of Projects (see Figure 67). It is used in conjunction with the various editors to create and configure the components of a Project.

**Figure 67** Project Explorer














Each component in Project Explorer has an icon to identify the component type (see [Project Explorer Icons](#) on page 104). Right-clicking a component in the Project Explorer displays a context menu for that component, from which you can select various options. Only those menu options that are allowed for the component in its current state are activated. See the following sections for more information.

- [Using the Project Context Menu](#) on page 105
- [Using Project Component Context Menus](#) on page 109

## 5.2.1 Project Explorer Icons

The icons described in Table 16 appear in the Project Explorer. A lock displayed in the lower-left corner of an icon indicates that the component is currently checked into the version control system (see the OTD icon example).

**Table 16** Project Icons

Icon	Description
	Represents the <b>Repository</b> , which is the central database where all Project information is saved. Binary files required at run time are also stored here.
	Represents a <b>Project</b> or Subproject.
	Represents a set of templates you can import to your Project.
	Represents a <b>Connectivity Map</b> , which contains the business logic and information about the data transmission.
	Represents a <b>Project variable</b> or <b>constant</b> .
	Represents an <b>Object Type Definition</b> (OTD) file.
	Represents a <b>Collaboration Definition (Java)</b> file.
	Represents a <b>Collaboration Definition (XSLT)</b> file.
	Represents a <b>Deployment Profile</b> , which specifies how Project components are deployed to a run-time Environment.
	When displayed along side one of the above icons, indicates that the current latest version of the component has been checked out for editing. See <a href="#">Managing Component Versions</a> on page 78 for additional information.
	When displayed along side one of the above icons, indicates that some version of the component — either the latest version or a previous version — has been retrieved to the local workspace. See <a href="#">Managing Component Versions</a> on page 78 for additional information.

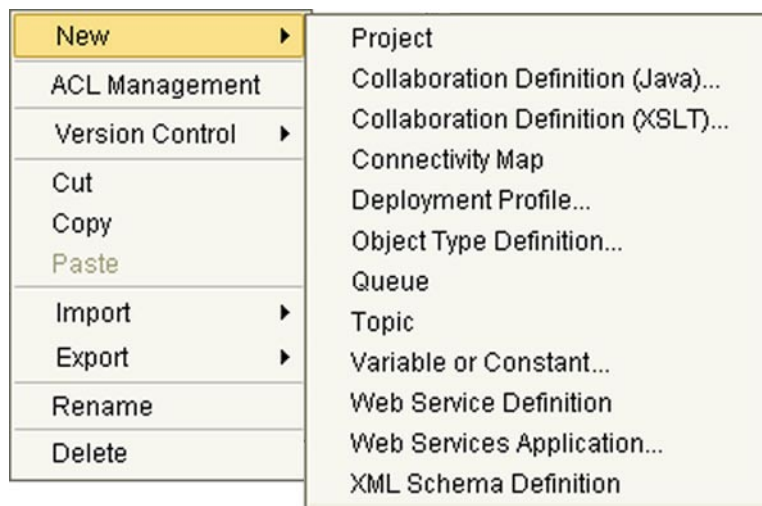
## 5.3 Project Components

You create an Project by selecting the **New Project** option from the *Repository* context menu in Project Explorer (see [Using the Repository Context Menu](#) on page 59). The different components available to you to incorporate into a Project are represented in the context menu.

### 5.3.1 Using the Project Context Menu

All Projects and Subprojects have context menus similar to that shown in Figure 68. Additional menu options may be shown, depending upon what additional products you have installed.

**Figure 68** Project Context Menu



**Table 17** Project Context Menu Options

Option		Function
New	Project	Adds a Subproject folder to the selected Project.
	Collaboration Definition (Java)	Presents the Collaboration Definition Wizard (Java), with which you can create a Java-based Collaboration Definition. See <a href="#">Creating a Java-based Collaboration Definition</a> on page 316.
	Collaboration Definition (XSLT)	Presents the Collaboration Definition Wizard (XSLT), with which you can create an XSLT-based Collaboration Definition. See <a href="#">Creating an XSLT-based Collaboration Definition</a> on page 381.
	Connectivity Map	Adds a Connectivity Map to the Project. See <a href="#">Connectivity Maps</a> on page 125.

**Table 17** Project Context Menu Options

Option		Function
New (continued)	Deployment Profile	Presents a dialog with which you can create a Deployment Profile for the selected Project. See <a href="#">Using the Deployment Editor</a> on page 130.
	Object Type Definition	Presents the OTD Wizard, with which you can create an Object Type Definition (OTD) file. See <a href="#">Object Type Definitions II</a> on page 172 for additional information.
	Queue	Adds a queue to your Project.
	Topic	Adds a topic to your Project.
	Variable or Constant	Presents a dialog with which you can add a constant or variable icon to your Project.
	Web Service Definition	Adds a WSDL document to your Project and opens the Web Services Designer. (see <a href="#">Creating Web Service Definitions</a> on page 466)
	Web Services Application	Adds a Web Services Application to the selected Project. See <a href="#">Web Service External Applications</a> on page 118.
	XML Schema Definition	Adds an XML Schema Definition (XSD) to your Project and opens the XSD Editor. (see <a href="#">Creating XML Schema Definitions</a> on page 418)
ACL Management		Presents the ACL Properties dialog, with which an Administrator can assign read/write privileges to users for the selected Project. See the <i>Sun SeeBeyond eGate™ Integrator System Administration Guide</i> .
Version Control	Check In	Presents a dialog with which you can check in a new version of the selected Project. Refer to <a href="#">Checking a Component In</a> on page 80 for more details.
	Check Out	Presents a dialog with which you can check out the current version of the selected Project. See <a href="#">Checking a Component Out</a> on page 82 for more information.
	Tag	Presents a dialog with which you can specify a tag to attach to the selected Project.
Cut		Copies the selected Project and removes it from the current Project, after which you can paste it to another Project as a Subproject (once only). All changes must be committed before you can cut the Project. Cut and paste is disabled for other users when you have the Project checked out. See the <i>Note</i> following this table, and <a href="#">Moving and Replicating Repository Objects</a> on page 97 for additional information.

**Table 17** Project Context Menu Options

Option		Function
Copy		Copies the selected Project, after which you can paste it to other Projects as a Subproject (multiple times). All changes must be committed before you can copy the Project. You can copy a Project even when another user has the Project checked out. See the <i>Note</i> following this table, and <a href="#">Moving and Replicating Repository Objects</a> on page 97 for additional information.
Paste		Pastes a cut or copied component into the selected Project (Projects are pasted as Subprojects). Pasting is disabled for other users when you have the Project checked out. See the <i>Note</i> following this table, and <a href="#">Moving and Replicating Repository Objects</a> on page 97 for additional information.
Import	File	Presents a dialog with which you can import an external file into the Project.
	Import Project	Presents a dialog with which you can import a Project as a Subproject under the selected Project. See <a href="#">Importing a Project or Environment</a> on page 67.
	Web Service Definition	Presents a dialog with which you can import existing Web Service Definitions for defining a web service. See <a href="#">Importing Existing Web Service Definitions</a> on page 491.
	XML Schema Definition	Presents a dialog with which you can import existing XML Schema Definitions for defining a web service.
Export	Export Project	Presents a dialog with which you can export the selected Project. See <a href="#">Exporting a Project or Environment</a> on page 62.
Rename		Activates the field, allowing you to rename the selected Project.

**Table 17** Project Context Menu Options

Option	Function
Delete	<p>Deletes the selected Project, subject to the following conditions:</p> <ul style="list-style-type: none"><li>▪ You have <i>write</i> privileges for the Project (see <i>ACL Management</i>, above).</li><li>▪ The Project is not checked out by anyone other than yourself.</li></ul> <p>If these conditions are true, a dialog is displayed in which you confirm that you want to delete the selected Project. Clicking <b>Yes</b> then deletes the Project.</p> <p>If the selected Project contains active Deployment Profiles, a dialog is displayed listing those deployments and requesting that you deactivate them. You must select deactivation to enable the <b>OK</b> button.</p>

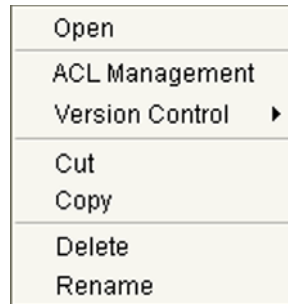
**Important:** All Project names should contain only alphanumeric characters (letters and numbers), dashes, and underscores.



### 5.3.2 Using Project Component Context Menus

Most Project components have context menus similar to that shown in Figure 69. Those context menus that are different are described in the sections for the specific components.

**Figure 69** Project Component Context Menu



**Table 18** Project Component Context Menu Options

Option	Function
Open	If shown, opens the selected component in the appropriate Enterprise Designer editor.
ACL Management	Presents the ACL Properties dialog, with which an Administrator can assign read/write privileges to users for the selected component. See the <i>Sun SeeBeyond eGate™ Integrator System Administration Guide</i> .
Version Control	Accesses the version control features for the selected component. See <a href="#">Accessing Component Version Control Features</a> on page 79 for additional information.
Cut	Copies the selected component and removes it from the current Project, after which you can paste it to another Project (once only). All changes must be committed before you can cut the component. Cut and paste is disabled for other users when you have the component checked out. See <a href="#">Moving and Replicating Repository Objects</a> on page 97 for additional information.
Copy	Copies the selected component, after which you can paste it to other Projects (multiple times). All changes must be committed before you can copy the component. You can copy and paste a component even when another user has the component checked out. See <a href="#">Moving and Replicating Repository Objects</a> on page 97 for additional information.

**Table 18** Project Component Context Menu Options

Option	Function
Delete	<p>Deletes the selected component, subject to the following conditions:</p> <ul style="list-style-type: none"><li>▪ You have <i>write</i> privileges for the component (see ACL Management, above).</li><li>▪ You have the component checked out to your workspace.</li><li>▪ The component is not checked out by anyone other than yourself.</li></ul> <p>If these conditions are met, a dialog is presented in which you confirm that you want to delete the selected component. Clicking <b>Yes</b> then deletes the component.</p>
Rename	<p>Activates the field, allowing you to rename the selected component.</p>

**Note:** Select **Refresh All from Repository** before you open any Project component (such as a Collaboration) to ensure that you open the latest version of the component.

**Note:** The version history for a component that has been **cut** and pasted is preserved, since there can be only one instance of it. The version history for a component that has been **copied** and pasted is **not** preserved, since there can be multiple instances of it; the version number for each pasted instance is reset.

**Note:** If a component is copied and pasted back into the original Project, the name is automatically modified with a suffix (**\_1**); in the case of multiple pastes, the suffix is incremented by 1 for each subsequent paste.

**Important:** All Project component names should contain only alphanumeric characters (letters and numbers), dashes, and underscores.

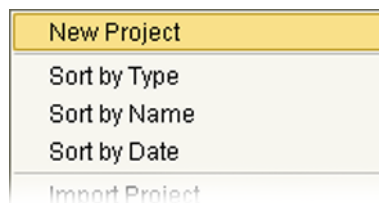
### 5.3.3 Creating a Project

You can create a Project by using the context menu system in Project Explorer together with the Connectivity Map Editor. This section contains a brief outline of the procedure — an end-to-end description can be found in the *Sun SeeBeyond eGate™ Integrator Tutorial*.

To create a Project

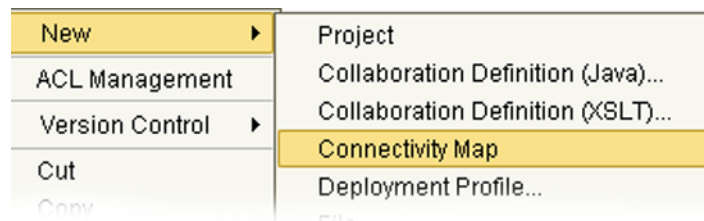
- 1 Right-click the **Repository** in the Project Explorer to display its context menu (see Figure 70).

**Figure 70** Repository Context Menu - New Project



- 2 Select a **New Project**, which initiates a hierarchical structure under the Repository.
- 3 Rename the Project as desired.
- 4 Right-click the **Project** to display its context menu (see Figure 71).

**Figure 71** Project Context Menu - New Connectivity Map



- 5 Select a **New Connectivity Map**, which displays the Connectivity Map Editor.
- 6 In the Connectivity Map Editor, populate the canvas with the various components needed for your Project (see [Connectivity Maps](#) on page 125), but leave the connectivity for later.
- 7 After you have developed the basic architecture of your Project, return to the Project context menu in Project Explorer.
- 8 Select a **New Object Type Definition**, and build your OTD as described in [Object Type Definitions II](#) on page 172.
- 9 Select a **New Collaboration Definition (Java or XSLT)**, and build your Collaboration Definition(s) as described either in [Collaboration Definitions \(Java\) I](#) on page 242 or [Collaboration Definitions \(XSLT\)](#) on page 377.

- 10 Return to the Connectivity Map and connect the various components (see [Connectivity Maps](#) on page 125).
- 11 Save and test your Project.

Once you have created your Project, you need to create a run-time Environment to represent your physical system, and map your Project to the Environment. See [Deployment Profiles](#) on page 130.

## 5.4 Services

A service provides a framework for a business process or a Collaboration, and contains the information required to execute a set of business rules.

### 5.4.1 Business Processes

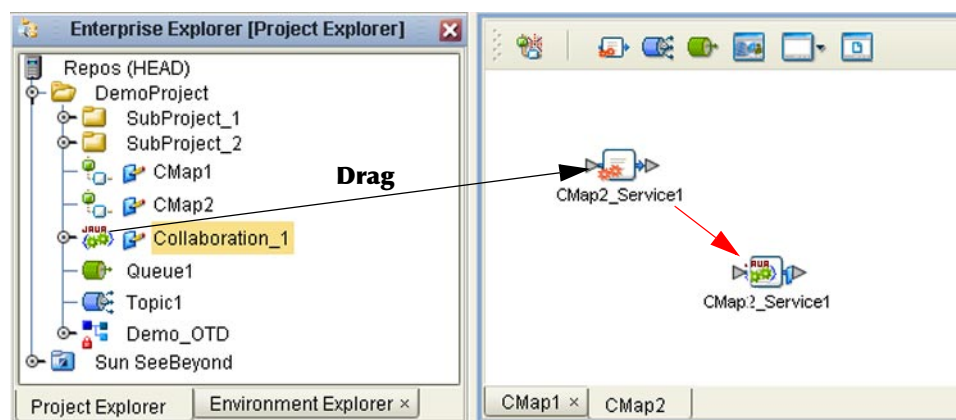
A business process is a collection of actions that take place in your company, revolving around a specific business practice. These business processes are modeled in eInsight Business Process Manager, and then embodied as a Project service in eGate Integrator. See the *Sun SeeBeyond eInsight™ Business Process Manager User's Guide* for information on business process modeling.

### 5.4.2 Collaborations

A Collaboration is a logical operation performed at run time between some combination of message destinations and external applications. The operation is specified by a Collaboration Definition, which can be written using either Java or XSLT.

The Collaboration acts as a service having a publication or subscription relationship with each linked entity. The link is provided by a JMS Client connection (see [Connecting Components](#) on page 128). Dragging a Collaboration Definition from the Project Explorer and dropping it onto the Service icon in the Connectivity Map defines the service as a Collaboration (see Figure 72). The icon changes as shown in the figure to indicate its new identity.

**Figure 72** Defining the Service Component



Any changes made to the names of the Collaborations should be done when the Collaboration is created. If the name is changed later, the Collaboration should be opened, regenerated if applicable, and saved again. For safe measure, this should also be done before creating the Connectivity Map and Deployment Profile.

Connection-related properties for the Collaboration (or other service) are configured in the adjoining JMS Client. These properties include:

- Concurrent or serial processing
- Transaction mode (transacted or XA)
- Security

All properties, and the procedures for configuring them, are detailed in the *Sun SeeBeyond eGate™ Integrator JMS Reference Guide*.

See also:

- **Collaboration Definitions (Java) I** on page 242
- **Collaboration Definitions (XSLT)** on page 377
- **Object Type Definitions I** on page 146

## 5.5 Message Destinations

A message destination is a container for stored data, and can follow either the topic or queue JMS model.

- A *topic* is a message destination that conforms to the publish-and-subscribe messaging paradigm.

**Figure 73** Topic Icon



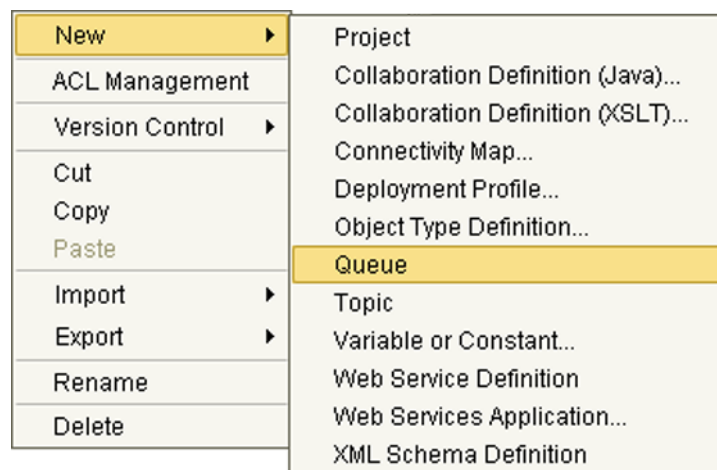
- A *queue* is a message destination that conforms to the point-to-point messaging paradigm.

**Figure 74** Queue Icon



You can add a message destination to a Project by selecting the **New > Queue** or **New > Topic** option from the *Project* context menu in Project Explorer (see Figure 75). You can also drag a Queue or Topic icon (see Figure 73 and Figure 74) from the Connectivity Map toolbar onto the canvas. Clicking the **Generate Connectivity Map** icon then adds the component to the Project in the Repository (see [Connectivity Maps](#) on page 125).

**Figure 75** Project Context Menu - New Queue



Right-clicking the queue or topic in Project Explorer displays its context menu (see [Using Project Component Context Menus](#) on page 109).

See the *Sun SeeBeyond eGate™ Integrator JMS Reference Guide* for additional information on message destinations.

## 5.5.1 Naming a Message Destination

### Standard Characters

As with other Project components, message destination names can contain alphanumeric characters (letters and numbers), dashes, and underscores.

### Special Characters

The following special characters can be used in message destination names:

<                      >                      &                      \$                      %

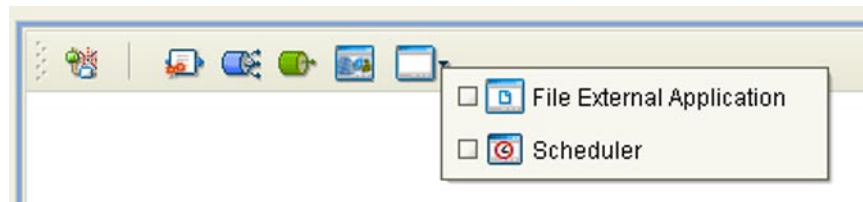


## 5.6 External Applications

The basic purpose of eGate Integrator is to facilitate the interchange of data between external business applications. These business applications are collectively referred to as external applications, and are represented in the Project by logical proxies for the specific applications involved. An external application can be identified with an ERP application such as SAP or PeopleSoft, a DBMS such as Oracle or SQL, or with a particular communications protocol, such as TCP/IP or HTTPS.

External applications are logical representations of external software applications that are being integrated by the eGate Integrator system. These are linked to a Service by means of an eWay. Clicking the drop-down arrow beside the external application icon in the Connectivity Map Editor presents a menu showing those applications corresponding to eWays that you have installed, plus the Scheduler. An example is shown in Figure 76.

**Figure 76** External Application Drop-Down Menu



Selecting the check box beside an individual external application adds that icon to the toolbar; clearing the check box removes it from the toolbar.

Right-clicking the external application in Project Explorer displays its context menu (see [Using Project Component Context Menus](#) on page 109). See the individual eWay user guides for additional information.

## 5.7 Web Service External Applications

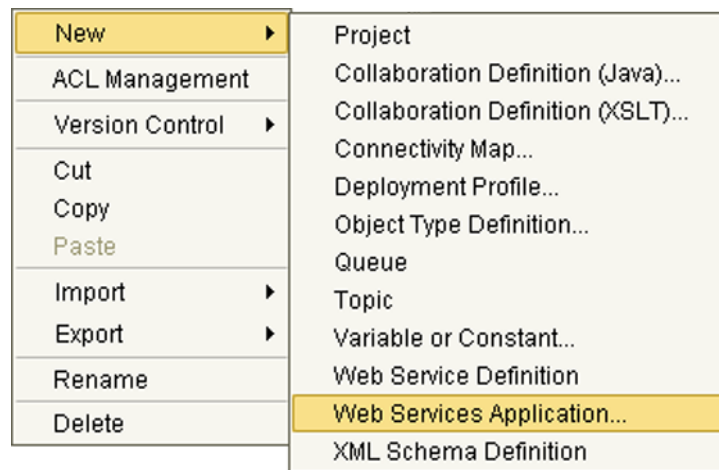
A *Web Service External Application* is the logical representation of a web service that appears in a Project (see [Developing Web Services](#) on page 413).

**Figure 77** Web Service External Application Icon



You can add a Web Service External Application to a Project by selecting the **New > Web Services Application** option from the *Project* context menu in Project Explorer (see Figure 78). You can also drag a Web Service External Application icon (see Figure 77) from the toolbar onto the canvas in the Connectivity Map Editor; clicking the **Generate Connectivity Map** icon then adds the component to the Project in the Repository (see [Connectivity Maps](#) on page 125).

**Figure 78** Project Context Menu - New Web Services Application



If you are creating more than one Web Service External Application for deployment in any given Environment, you must ensure that the respective *servlet context* properties have different values so that each application will have a different URL. See [SOAP/HTTP Web Service External Systems](#) on page 524.

Right-clicking the Web Service External Application in Project Explorer displays its context menu (see [Using Project Component Context Menus](#) on page 109).

## 5.8 Schedulers

A *Scheduler*, or *Scheduler Adapter*, allows a service to be performed at a prescribed interval. You create this component by dragging the **Scheduler** icon (see Figure 79) from the Connectivity Map toolbar onto the canvas; clicking the **Generate Connectivity Map** icon then adds the component to the Project in the Repository (see [Connectivity Maps](#) on page 125).s.

**Figure 79** Scheduler Icon



Right-clicking the Scheduler in Project Explorer displays its context menu (see [Using Project Component Context Menus](#) on page 109).

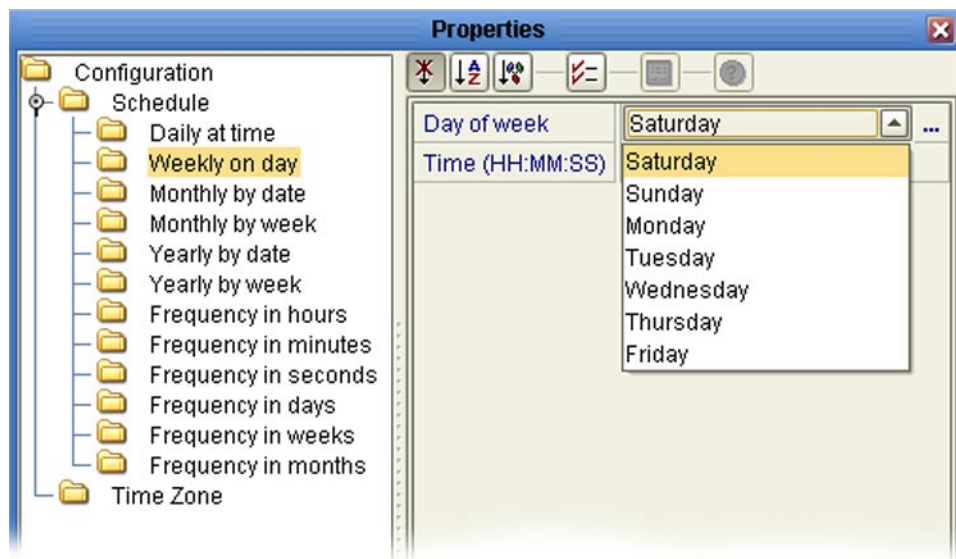
### 5.8.1 Configuring a Scheduler

Once the scheduler is connected to a service in the Connectivity Map, double-clicking the JMS Client (the connector icon) displays the *Properties* dialog for that scheduler.

#### Schedule

Expanding the **Schedule** node displays the list of schedule types from which you can select the one most appropriate for your use. Selecting the type displays the properties for that type; for example, **Weekly on day** displays entry fields for the day of the week and the time (see Figure 80).

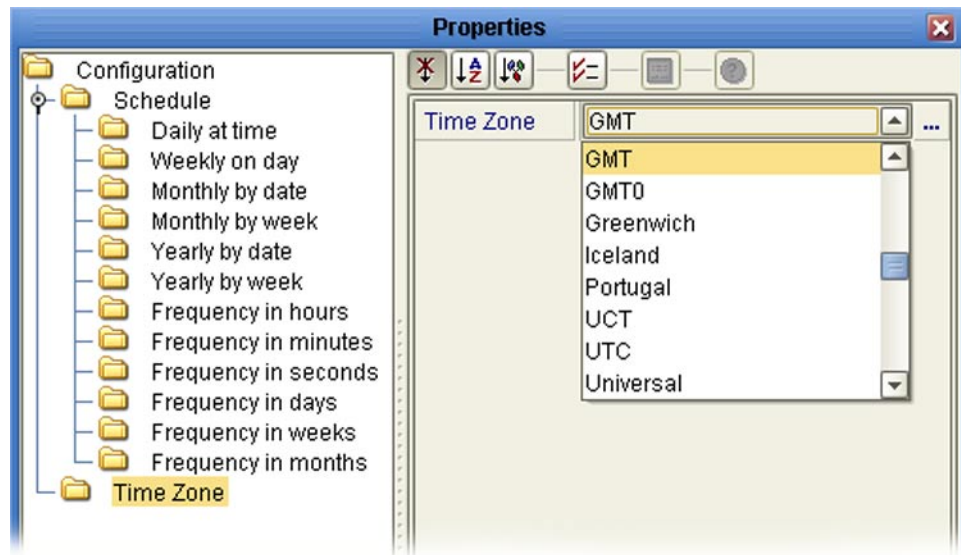
**Figure 80** Scheduler Properties Dialog - Schedule



## Time Zone

Selecting **Time Zone** displays the *Time Zone* entry field in which you specify the time zone that you want the scheduler to be synchronized with (see Figure 81).

**Figure 81** Scheduler Properties Dialog - Time Zone



### 5.8.2 Naming a Scheduler

#### Standard Characters

As with other Project components, scheduler names can contain alphanumeric characters (letters and numbers), dashes, and underscores.

#### Special Characters

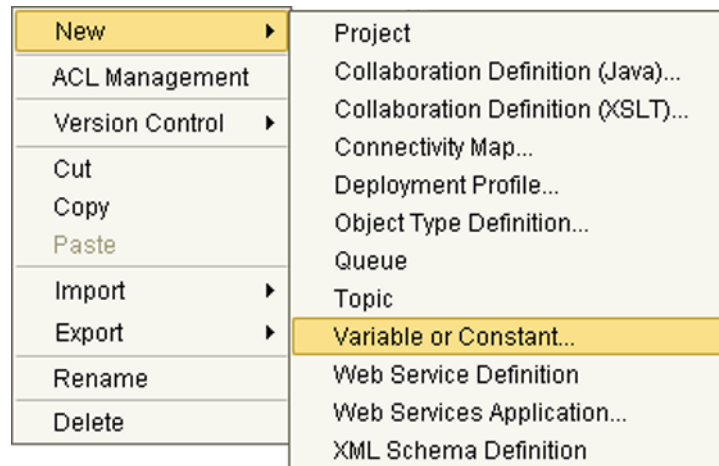
The following special characters can be used in scheduler names:

<                      >                      &

## 5.9 Constants and Variables

You can add a constant or a variable to a Project by selecting the **New > Variable or Constant** option from the *Project* context menu in Project Explorer (see Figure 82).

**Figure 82** Project Context Menu - New Variable or Constant



Right-clicking the constant or variable in Project Explorer displays its context menu (see [Using Project Component Context Menus](#) on page 109).

### 5.9.1 Project Constants

Constants are name-value pairs that are visible across the Project. For example, Figure 83 shows a standard currency defined to be used globally throughout the system.

**Figure 83** Creating a Project Constant

The screenshot shows a Windows-style dialog box titled "New Variable or Constant for SBYN\_Warehouse". It contains the following fields and controls:

- Name:** A text box containing "STANDARD\_CURRENCY".
- Category:** A text box containing "Project Constant".
- Description:** A text box containing "This represents a standard currency that is used throughout the entire system.".
- Is a Constant:** A checkbox that is checked.
- Value Type:** A dropdown menu currently set to "String".
- Value:** A text box containing "USD".
- Buttons:** "OK" and "Cancel" buttons at the bottom.

**Table 19** Project Variable Options

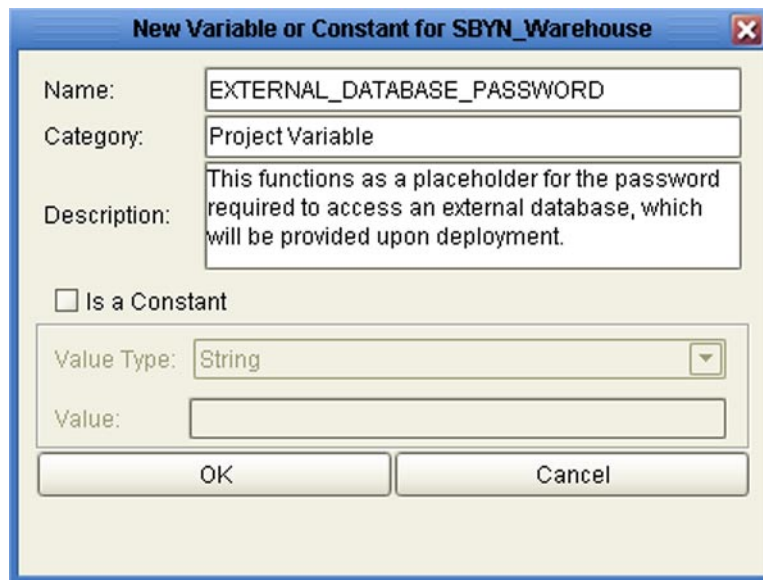
Property	Option	Description/Usage
Name		Your name for the Project constant.
Category		You may assign a category name, if desired.
Description		Your description for the constant.
Is a Constant		Select to enable the constant properties fields.
Value Type	String	Allows the string value to be displayed explicitly.
	Password	Encrypts the value, displaying asterisks (*) in the field.
Value		The value for the constant.

## 5.9.2 Project Variables

Variables function as placeholders, having values that are determined when you create a specific Deployment Profile (see [Mapping Variables](#) on page 136). Project variables can be literals or Environmental constants (see [Environmental Constants](#) on page 512).

As an example, Figure 84 shows a Project variable defined to represent a password of a database user in a target Environment. A system manager assigns an actual value to this variable in the Deployment Profile editor. The value of the assigned Project variable (an Environmental constant) is then used to connect to the database in the target Environment.

**Figure 84** Creating a Project Variable



The screenshot shows a dialog box titled "New Variable or Constant for SBYN\_Warehouse". It contains the following fields and options:

- Name:** A text box containing "EXTERNAL\_DATABASE\_PASSWORD".
- Category:** A dropdown menu showing "Project Variable".
- Description:** A text box containing "This functions as a placeholder for the password required to access an external database, which will be provided upon deployment."
- Is a Constant:** An unchecked checkbox.
- Value Type:** A dropdown menu showing "String".
- Value:** An empty text box.
- Buttons:** "OK" and "Cancel" buttons at the bottom.

**Table 20** Project Variable Options

Property	Description/Usage
Name	Your name for the Project variable.
Category	You may assign a category name, if desired.
Description	Your description for the variable.
Is a Constant	Does not apply to variables; leave unchecked.
Value Type	Does not apply to variables.
Value	Does not apply to variables.

### 5.9.3 Variables & Constants Object Group

Constants and variables are automatically added to a Variables and Constants object group within the Project (see Figure 85). Selecting an entry displays it in the lower panel, where it can be modified. Clicking **OK** overwrites the previous definition with the modified version.

**Figure 85** Variables and Constants Object Group

Name	Constant	Value	Category	Description
STANDARD_CURR...	<input checked="" type="checkbox"/>	USD	Project Constant	This represents a standard curren...
EXTERNAL_DATAB...	<input type="checkbox"/>		Project Variable	This functions as a placeholder fo...

  
**Modify a Variable or Constant**  

Name:	STANDARD_CURRENCY	<input checked="" type="checkbox"/> Is a Constant
Category:	Project Constant	Value Type: String
Description:	This represents a standard currency that is used throughout the entire system.	Value: USD

OK

SBYN\_Warehouse\_Variables

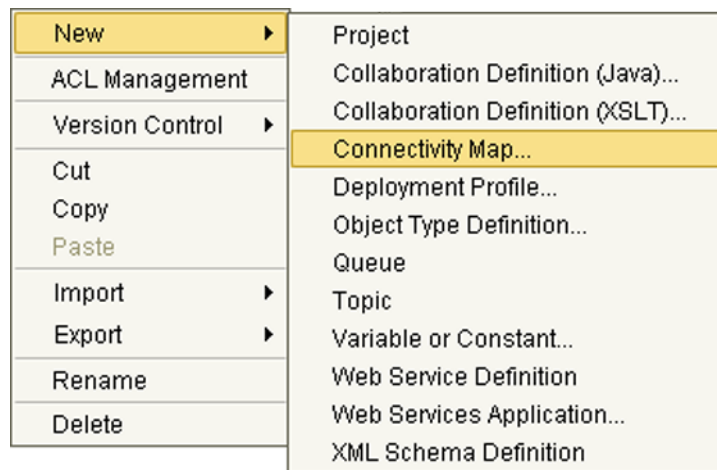


## 5.10 Connectivity Maps

A *connectivity map* is a graphical representation of your Project, containing the various logical components comprising the Project and the links between them. The Connectivity Map Editor allows you to create your Project by simply dragging and dropping icons onto a Project canvas and then connecting them to form data paths. You then can configure the components by means of dialogs that are displayed by clicking on the component icons.

You can add a connectivity map to a Project by selecting the **New > Connectivity Map** option from the *Project* context menu in Project Explorer (see Figure 86).

**Figure 86** Project Context Menu - New Connectivity Map



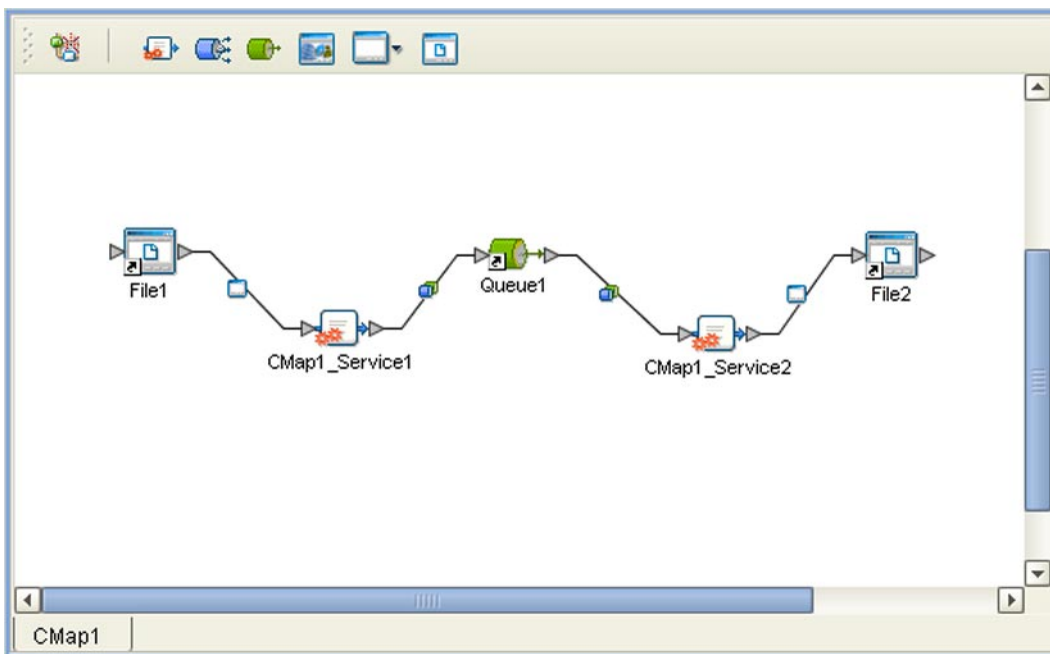
Right-clicking the connectivity map in Project Explorer displays its context menu (see [Using Project Component Context Menus](#) on page 109).

### 5.10.1 Using the Connectivity Map Editor

When you create a new Connectivity Map in the Enterprise Explorer, the editor panel displays the Connectivity Map Editor (see Figure 87). To define your Project, you simply drag icons from the toolbar to the workspace, or canvas, to populate the Connectivity Map with the necessary components. (You can also create objects in Project Explorer using the Project context menu, and drag them onto the Connectivity Map canvas.)

You subsequently link the components by dragging the cursor from the output arrowhead of one to the input arrowhead of the other (the cursor turns into a hand).

**Figure 87** Connectivity Map Editor










The drag-and-drop components include services, queues, topics, schedulers, and external applications. Additional components, such as *eWays* and *JMS Clients*, are placed automatically when you link the components you have placed manually (see [Connecting Components](#) on page 128).

### 5.10.2 Connectivity Map Editor Toolbar

The Connectivity Map Editor toolbar contains the icons listed in Table 21, plus additional icons representing eGate Integrator add-ons and other Java™ Composite Application Platform Suite components that you may have installed.

**Table 21** Connectivity Map Editor Toolbar Icons

Icon	Component	Function
	Generate Connectivity Map	Generates an object in the Repository corresponding to the graphical construct on the Connectivity Map canvas.
	Service	A logical component that provides the framework for a process or Collaboration. See <a href="#">Services</a> on page 113.
	Queue	A Message Destination that conforms to the point-to-point messaging paradigm, having one sender and one receiver. See <a href="#">Message Destinations</a> on page 115.
	Topic	A Message Destination that conforms to the publish/subscribe messaging paradigm, having one sender (publisher) and multiple receivers (subscribers). See <a href="#">Message Destinations</a> on page 115.
	Web Service External Application	Represents a Web Service External Application (see <a href="#">Web Service External Applications</a> on page 118).
	External Applications	Represents an application external to eGate Integrator. Click the arrow beside the icon to view a list of specific applications to which you can connect. See <a href="#">External Applications</a> on page 117.
	Scheduler	Represents a scheduling component, which appears on the list of External Applications. Use this component to set data transfer to occur at set intervals. See <a href="#">Schedulers</a> on page 119.

It is important to understand that the logical components appearing in the Connectivity Map are essentially *placeholders* that refer to the “actual” components that exist in the Repository and appear in the Project Explorer. Renaming or deleting a queue or topic in the Connectivity Map only affects the placeholder, not the object in the Repository.

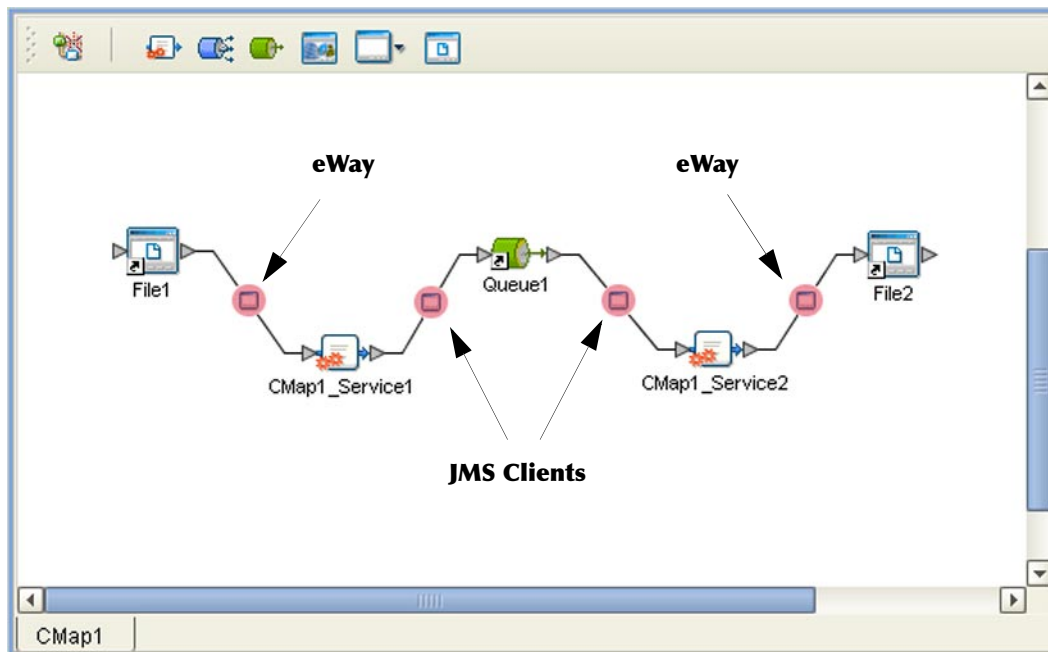
Also, renaming or deleting a queue or topic in the Repository will not affect the existence or name of the associated placeholder in the Connectivity Map. The change will, however, be reflected in the *tooltips* for the placeholder. This allows you to re-assign the placeholder without disrupting the continuity of the Connectivity Map.

### 5.10.3 Connecting Components

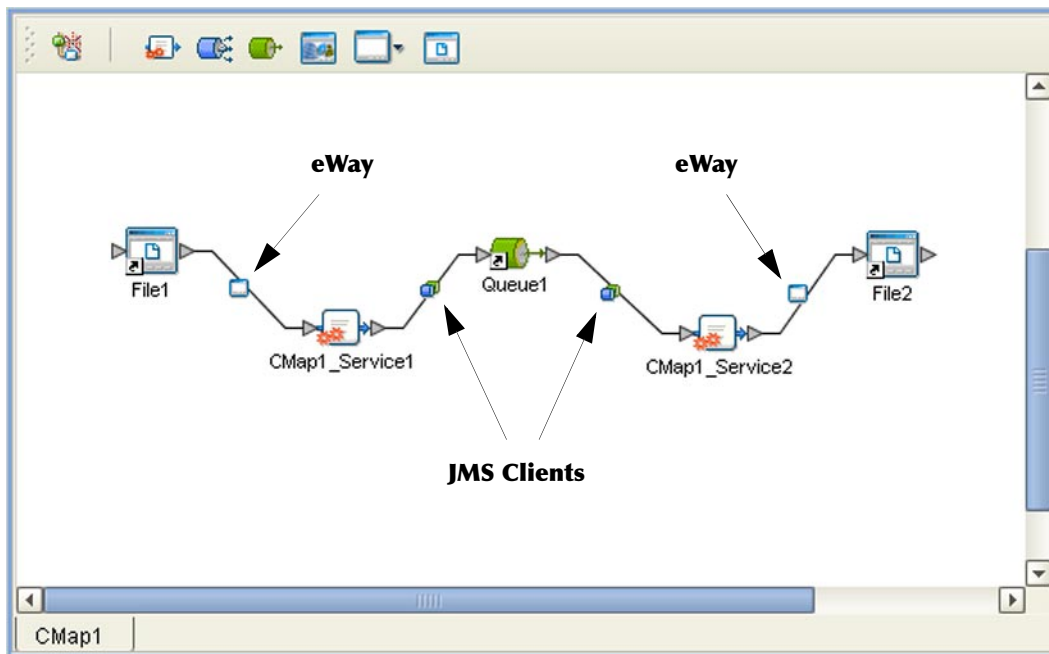
When you link two components on a Connectivity Map, Enterprise Designer places a connection icon on the link; the entity represented by the icon depends upon the type of components you are linking.

- When you link an external application with a Collaboration, Enterprise Designer automatically adds an eWay adapter to the link. The eWay enables communication and movement of data between the external application and the eGate Integrator system. The eWay configuration specifies the logical connection properties for the link. See the individual eWay adapter user guides for specific information.
- When you link a Web Services External Application with a Collaboration, Enterprise Designer automatically adds a Web Service External Application Connector to the link. The connector enables communication and movement of data between the web services application and the eGate Integrator system. The binding configuration specifies the logical connection properties for the link. See [Configuring WSDL Binding Properties](#) on page 496 for specific information.
- When you link a Service with a Message Destination (queue or topic), Enterprise Designer adds a JMS Client connection to the link. The JMS Client configuration specifies the logical connection properties for the linked Service. See the *Sun SeeBeyond eGate™ Integrator JMS Reference Guide* for information.

**Figure 88** Connection Icons in a Connectivity Map - Not Configured

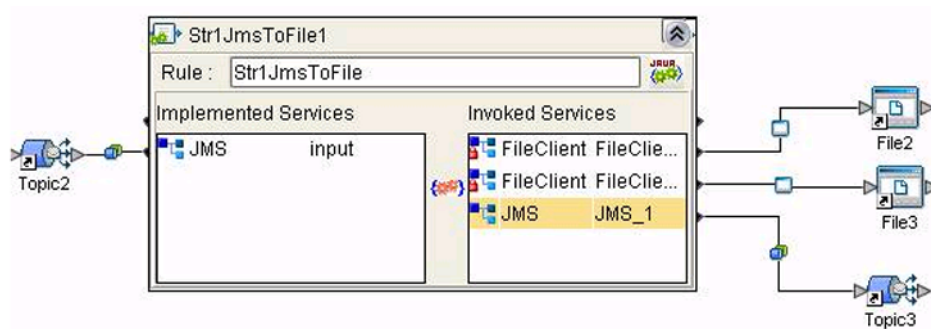


Initially, these connectors are shown as red disks, as illustrated in Figure 88. Double-clicking the connection icon displays a Properties dialog, in which you can configure the connector. Clicking **Yes** in the dialog then sets the configuration and changes the icons in the Connectivity Map, as shown in Figure 89.

**Figure 89** Connection Icons in a Connectivity Map - Configured

## Linking Multiple Destinations

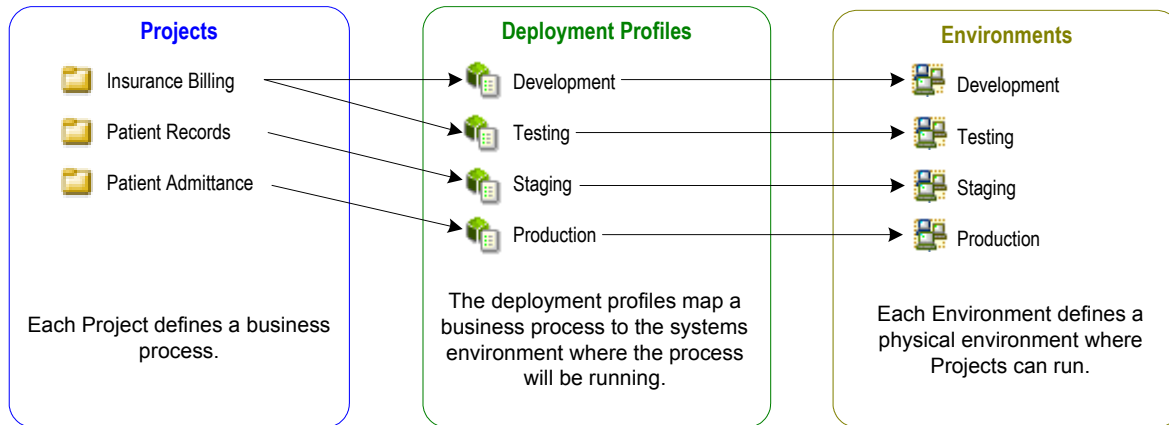
When there are multiple destinations, as with a JMS topic, the Connectivity Map Editor cannot resolve which output port connects to which destination. Because of this, you must first create the Collaboration Definition, and then make the connections by opening the Collaboration binding box and connecting the output ports for the individual services to their intended destinations (see Figure 90).

**Figure 90** Linking Multiple Destinations

## 5.11 Deployment Profiles

Deployment profiles define how specific instances of a Project are deployed to a particular Environment (see Figure 91). A deployment profile contains information about the assignment of services and message destinations to integration and message servers. It also contains version information for all relevant objects in the Project.

**Figure 91** eGate Integrator Implementation Model

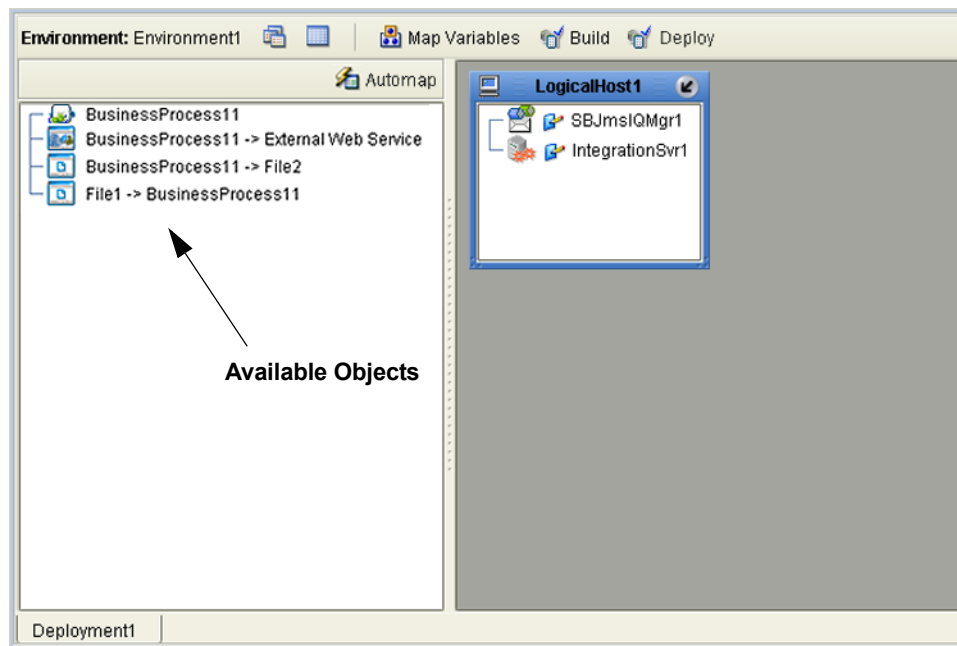








To create a new deployment profile, right-click on a Project in the Project Explorer to display its context menu. From the menu, select **New > Deployment Profile** (see [Creating a Deployment Profile](#) on page 132). To edit an existing deployment profile, select **Open** from its context menu (see [Using Project Component Context Menus](#) on page 109). Either of these actions opens the Deployment Editor, where you can map Project components to the appropriate Environment components.

### 5.11.1 Using the Deployment Editor

The Deployment Editor (see Figure 92) provides a canvas in which you define how Project components will be deployed in a run-time Environment. The Deployment Editor contains controls that allow you to:

- Map Project components to the appropriate Environment components.
- Build an enterprise archive (EAR) file.
- Deploy the EAR file to a run-time Environment.

**Figure 92** Deployment Editor**Table 22** Deployment Editor Toolbar Commands

Button/Icon	Function
	Selects the Map view of the Deployment Editor (default).
	Selects the Spreadsheet view of the Deployment Editor (see <a href="#">Figure 105 on page 137</a> ).
	Automatically deploys components to their matching containers, when there is a one-to-one correspondence between them. See <a href="#">Automapping</a> on page 135.
 Map Variables	Allows you to assign names and values to Project variables for the specific Deployment Profile. See <a href="#">Mapping Variables</a> on page 136.
 Build	Builds the enterprise archive (EAR) file, based on the Project and its Deployment Profile (see <a href="#">Building an Application File</a> on page 139).
 Deploy	Deploys the enterprise archive (EAR) file built in the previous step, following the Deployment Profile you have defined (see <a href="#">Deploying the Project</a> on page 145).

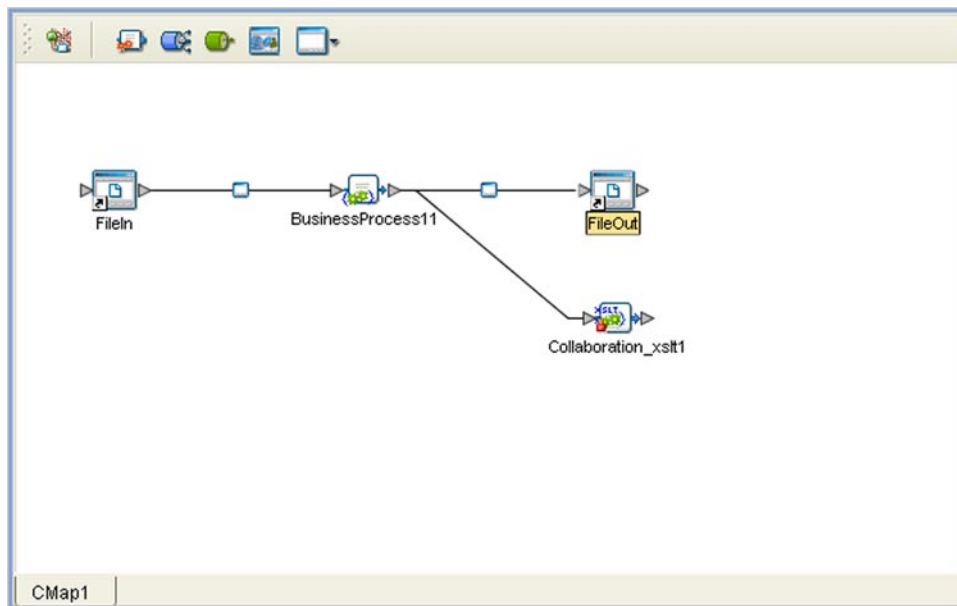
### 5.11.2 Creating a Deployment Profile

A Project having the Connectivity Map shown in Figure 93 will be used as a deployment example in the following procedure.

**Note:** Each of a Project's active Deployment Profiles must target a separate Environment; conversely, any given Environment can have only one Deployment Profile assigned to it from a given Project.

**Important:** Deployment Profile names should contain only alphanumeric characters (letters and numbers), dashes, and underscores.

**Figure 93** Example Project Connectivity Map

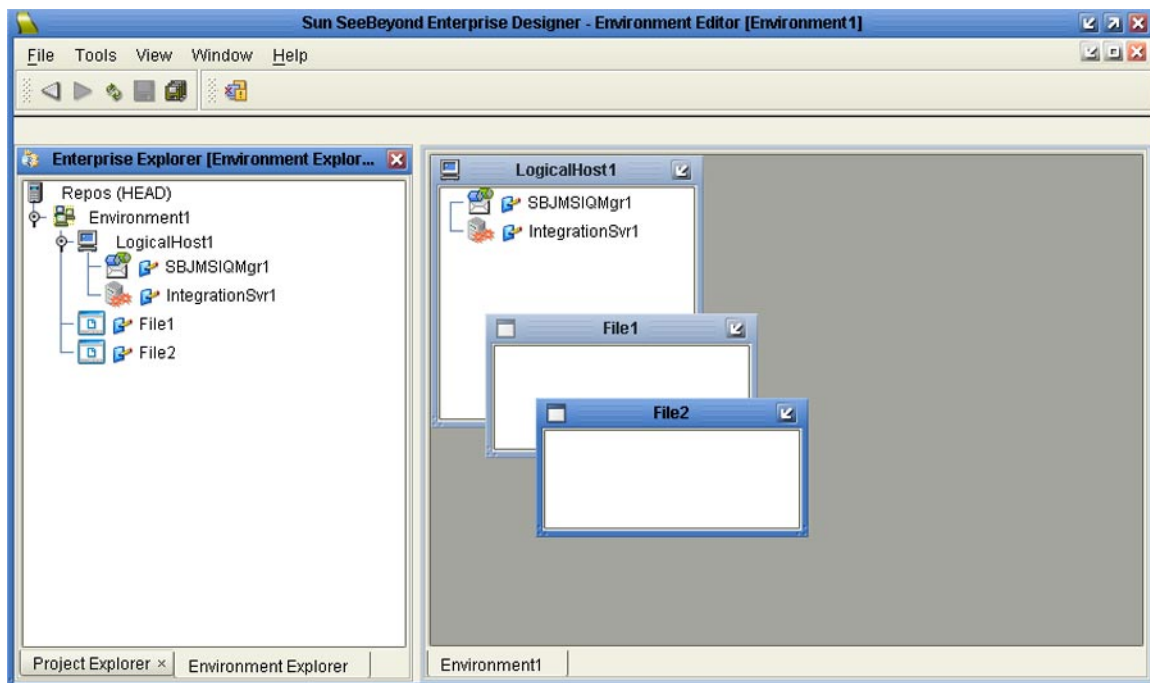


#### To create a Deployment Profile

- 1 In the Environment Explorer, right-click on the Repository to display its context menu.
- 2 Select **New Environment**, and assign an appropriate name.
- 3 Right-click on the Environment to display its context menu, and create the components you need, for example:
  - A A new Logical Host
  - B A new Integration Server
  - C A new JMS IQ Manager
  - D Two new File External Systems, one Inbound and one Outbound.
- 4 Name the Environmental components appropriately. They will appear as shown in Figure 94.

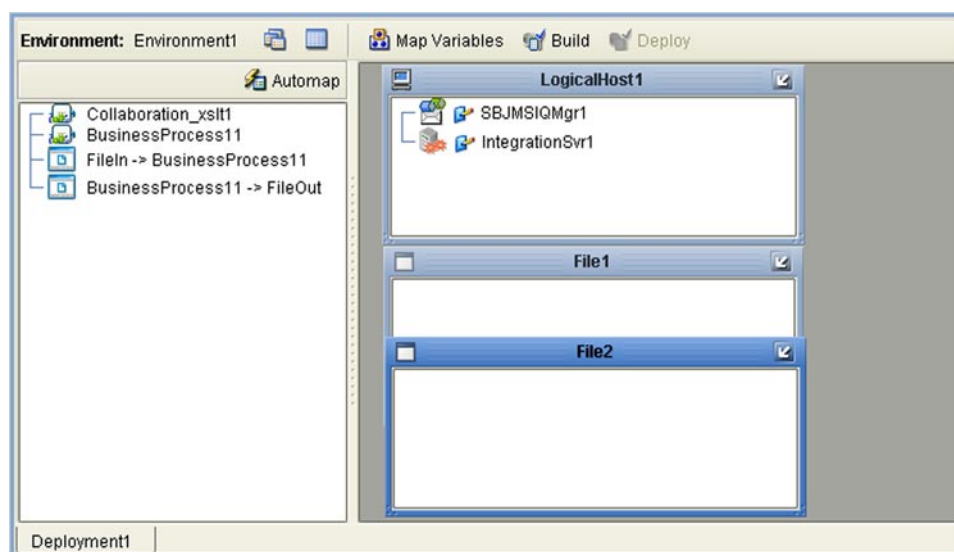


**Figure 94** Web Client Example Environment



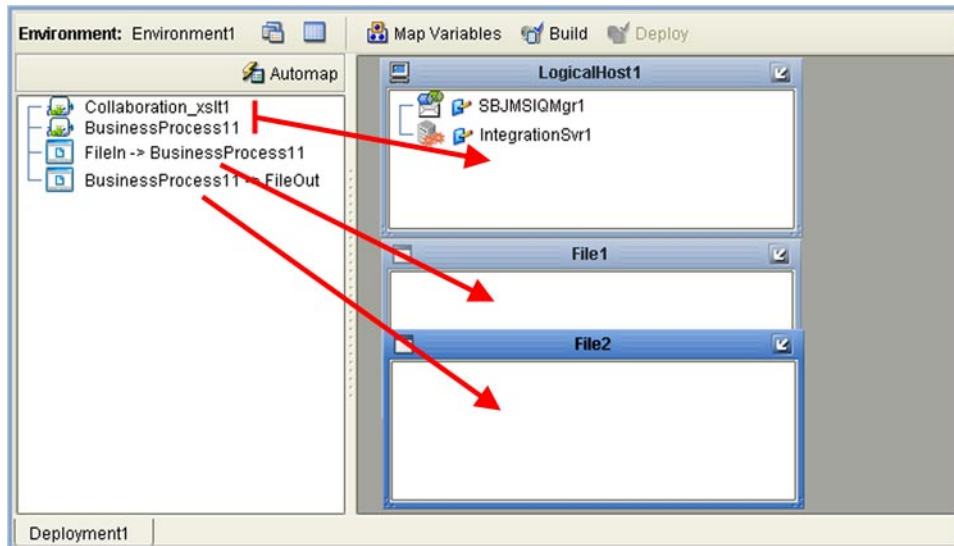
- 5 In the Project Explorer, right-click on the Project to display its context menu.
- 6 From the menu, select **New > Deployment Profile**. The Deployment Profile Editor appears, displaying the Environment you created previously along with all deployable Project components (see Figure 95).

**Figure 95** Example Deployment Profile (1)



- 7 Drag the Project components from the left panel and drop them into the appropriate Environment components in the right panel, as illustrated in Figure 96.

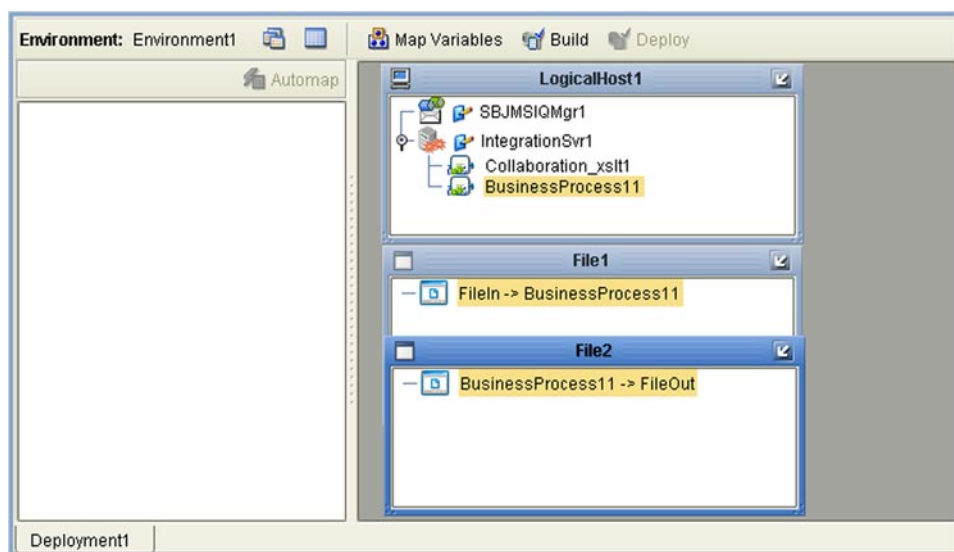
**Figure 96** Example Deployment Profile (2)



**Note:** You can select multiple components of the same type that are destined for a single external system, dragging them to that external system in one operation.

- 8 When the Environment components are fully populated, the left panel will be blank, as shown in Figure 97. You should now **Save** the profile.

**Figure 97** Example Deployment Profile (3)



## Automapping

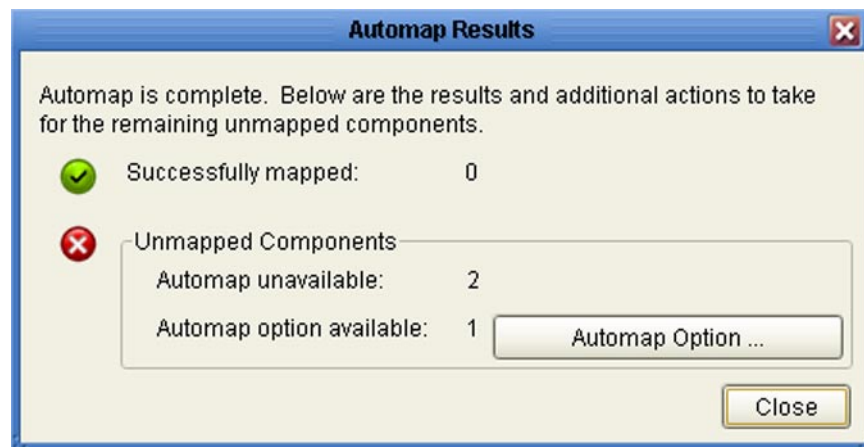
When a one-to-one correspondence exists between the available objects and the containers in the Environment to which you want to deploy them, clicking the **Automap** icon (see Figure 98) automatically deploys the components to their matching containers. (This feature only works with external systems for which it is enabled.)

**Figure 98** Automap Icon



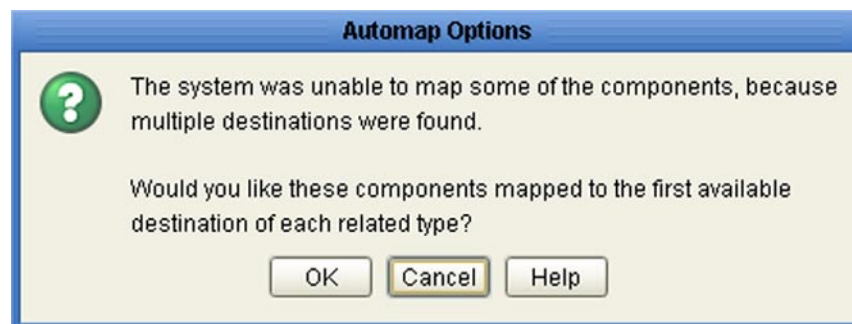
After the Automap feature executes, a dialog box is displayed showing the results (see Figure 99). The lack of an appropriate container, or ambiguity between potential containers, will cause a component to remain unmapped.

**Figure 99** Automap Results Dialog Box



In the case of an ambiguity, for example, mapping options are presented in an Automap Options dialog box (see Figure 100) that is displayed when you click **Automap Option**. You may click **OK** to accept the Automapping option, or click **Cancel** and map the deployable components manually.

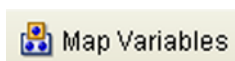
**Figure 100** Automap Options Dialog Box



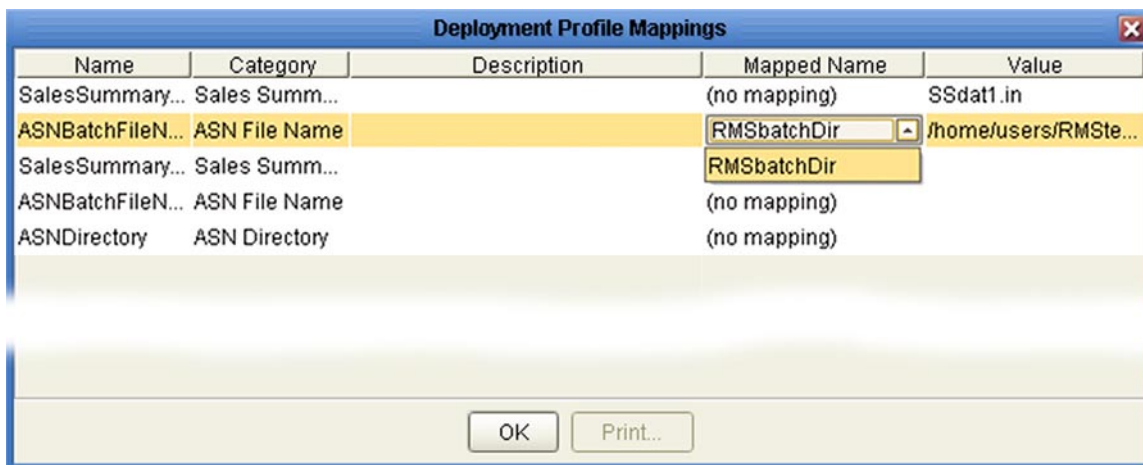
## Mapping Variables

Project variables function as placeholders, having values that are determined when you create a specific Deployment Profile. These values can be literals or Environmental constants (see [Environmental Constants](#) on page 512). Clicking the **Map Variables** button displays the Deployment Profile Mappings panel, where you can assign names (see Figure 102) and values (see Figure 103).

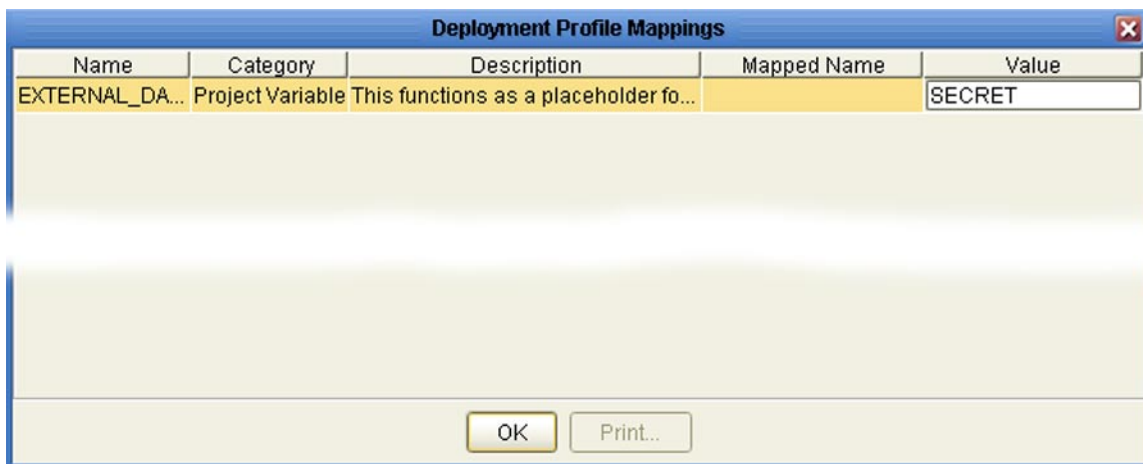
**Figure 101** Map Variables Button



**Figure 102** Deployment Profile Mappings



**Figure 103** Project Variable Value Entry



## Version Control

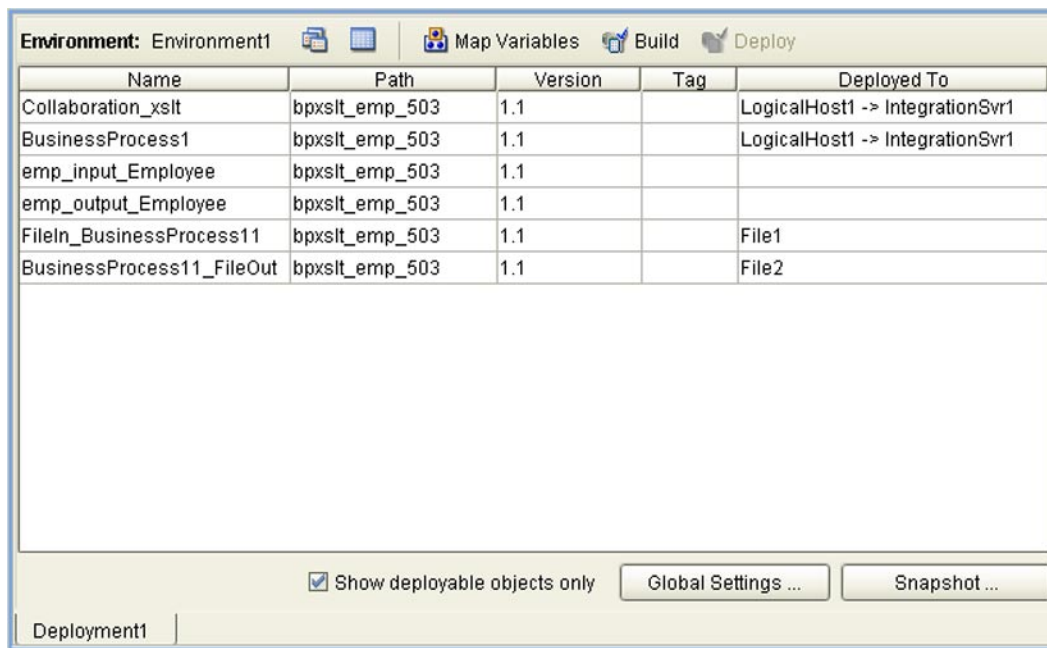
The Deployment Profile Editor allows you to control which version of the various Project components get mapped in the Deployment Profile. The editor provides a spreadsheet-like view to display the Project components and their versions. This view includes the component name, project path, current version in the users' workspace, tag on the version (if there is one), and to what external system the component is mapped. This view is enabled by clicking the Spreadsheet View icon in the Deployment Profile Editor toolbar (see Figure 104).

**Figure 104** Spreadsheet View Icon



The spreadsheet view (see Figure 105) has an option that is checked by default to show only those components that are displayed when mapping components to external systems. If not checked, those components not ordinarily seen in the Deployment Profile (for example, Connectivity Maps and OTDs) will be listed as well. Listing them allows you to also specify the version of those components.

**Figure 105** Spreadsheet View



Name	Path	Version	Tag	Deployed To
Collaboration_xslt	bpxslt_emp_503	1.1		LogicalHost1 -> IntegrationSvr1
BusinessProcess1	bpxslt_emp_503	1.1		LogicalHost1 -> IntegrationSvr1
emp_input_Employee	bpxslt_emp_503	1.1		
emp_output_Employee	bpxslt_emp_503	1.1		
FileIn_BusinessProcess11	bpxslt_emp_503	1.1		File1
BusinessProcess11_FileOut	bpxslt_emp_503	1.1		File2

☒ Show deployable objects only    Global Settings ...    Snapshot ...

Deployment1

The spreadsheet view allows you to specify the version for each component by selecting a version or tag from a drop-down list. Selecting versions or tags does not affect your workspace, but is only for setting up the versions to appear in a snapshot; only when the snapshot is retrieved will your workspace be modified.

## Creating Snapshots

The Deployment Profile Editor allows you to take a snapshot of the configuration currently in your workspace, as shown in the spreadsheet view. Clicking the **Snapshot** button presents a dialog in which you enter a name for the snapshot (see Figure 106). Clicking **OK** then saves the snapshot.

**Figure 106** Snapshot Dialog



## Retrieving Snapshots

The Deployment Profile Editor gives you the option of deploying the latest component versions from the Repository or those versions recorded in a snapshot of your workspace at a previous time. Clicking the **Global Settings** button presents a dialog in which you can specify your choice and, if appropriate, name the desired snapshot (see Figure 107). Clicking **OK** then carries out the operation.

**Figure 107** Global Settings Dialog



In either case, if your chosen configuration is different from what is currently in your workspace, you will receive another dialog informing you that the action will modify what is in your workspace and offering you the opportunity to not perform the operation.

- If you select a specific snapshot which is different from what is currently in your workspace, the versions of those components used by the Deployment Profile that are part of the specified snapshot will be retrieved into your workspace as read-only versions.
- If you select to use the latest Repository versions while another configuration is currently in your workspace, those components used by the Deployment Profile that are currently in your workspace will be removed and replaced by the latest versions from the Repository.

### 5.11.3 Building an Application File

The end product of the Project design process is an *application file* that can be deployed to an application/integration server. This enterprise archive (EAR) file contains a collection of **.jar** files, classes, and resources. In the case of a web application, the corresponding web archive (WAR) file contains a group of **.jar** files, classes, and resources that can be packaged and accessed as a single servlet context.

Once a Deployment Profile has been defined for your Project, you can build the EAR or WAR file by clicking the **Build** button in the Deployment Editor toolbar (see Figure 108). You can also generate an EAR file from the command line, without having Enterprise Designer running (see [Building an Application File From the Command Line](#) on page 140).

**Figure 108** Build Button



The resulting EAR or WAR file is stored in the following directory.:

```
<Sun_JavaCAPS_install_dir>\edesigner\builds
```

**Note:** *By default, the EAR file does not contain the associated Java source files. For the Java source files to appear in the EAR file, you need to change the value of the **run.mode** property in the **runed.bat** file from **run** to **debug** before building the file. Note that the resulting EAR file can be quite large.*



## Building an Application File From the Command Line

You can generate an EAR file from the command line, without having Enterprise Designer running, by using the **commandline codegen** tool. When using this tool, you have three options for specifying the build properties for the EAR file:

- 1 You can specify the properties in the **build.properties** file.
- 2 You can specify the properties explicitly when issuing the command.
- 3 You can specify properties globally in the **build.properties** file, and override them selectively by specifying those to be overridden when issuing the command.

In all cases, the resulting **.ear** file will be located in the following directory:

```
commandlinecodegen\localrepository\DEST
```

**Note:** All relevant components must have been saved in Enterprise Designer prior to running **commandline codegen**.

**Table 23** Commandline Codegen Properties

Property Name	Description
commandline.rep.url	Repository URL (required). The format is: <i>http://&lt;host&gt;:&lt;port&gt;/&lt;repositoryname&gt;</i>
commandline.rep.user	Repository user name (required).
commandline.rep.pass	Repository user password (required).
commandline.rep.dir	Repository root directory (required). The default value is <i>localrepository</i> .
commandline.rep.projectName	Project name (required).
commandline.rep.projectDeployName	Deployment Profile name (required).
commandline.rep.projectDeploymentTag	Project snapshot name and object tags (optional). The comma-delimited format is: <i>&lt;snapshot name&gt;, &lt;tag1&gt;, &lt;tag2&gt;, ... , &lt;tagN&gt;</i>
commandline.rep.projectBranchName	Repository Branch for the Project (optional).
commandline.esr.select	Specific ESR numbers to be downloaded and applied (optional). If not specified, <i>all</i> ESRs will be downloaded. The comma-delimited format is: <i>&lt;ESRnum1&gt;, &lt;ESRnum2&gt;, ... , &lt;ESRnumN&gt;</i>
commandline.esr.ignore	Specific ESR numbers to be ignored when downloading (optional). The comma-delimited format is: <i>&lt;ESRnum1&gt;, &lt;ESRnum2&gt;, ... , &lt;ESRnumN&gt;</i>

**Note:** The parameters **commandline.esr.select** and **commandline.esr.ignore** are mutually exclusive; only one or the other can be set.

**Important:** Ensure there are no leading or trailing spaces in the values you enter for variables.



### To set up your system to use commandline codegen

- 1 Set the **JAVA\_HOME** environmental variable on your computer to the location where JRE version 1.4 is installed (as part of the eGate Integrator installation), for example:

```
set JAVA_HOME=c:\JavaCAPS51\edesigner\jdk
```

- 2 Install Apache **Ant**, version 6 or later, on your computer. You can download this program from the following URL:

```
http://ant.apache.org/bindownload.cgi
```

- 3 Install **commandline codegen** following the instructions in the *Sun Java™ Composite Application Platform Suite Installation Guide*, if you have not already done so (this component is usually installed when you install the other eGate components).
- 4 At the command-line prompt in your *commandlinecodegen* directory, set the **ANT\_HOME** environmental variable, for example:

```
set ANT_HOME=c:\ant6.2
```

### To build an application file based on a property file only (Windows)

- 1 Locate the file **build.properties** in your *commandlinecodegen* directory.
- 2 Open the file using a text editor such as *Notepad*. The contents are as follows:

```
commandline.rep.url=http://<host>:<port>/<repositoryname>  
commandline.rep.user=  
commandline.rep.pass=  
commandline.rep.dir=localrepository  
commandline.rep.projectName=  
commandline.rep.projectDeployName=  
commandline.rep.projectDeploymentTag=  
commandline.rep.projectBranchName=  
commandline.esr.select=  
commandline.esr.ignore=
```

- 3 Fill in the properties as described in Table 23 (optional properties may be left blank).
- 4 Run **commandline codegen** by issuing the following command from the command-line prompt in your *commandlinecodegen* directory:

```
ant -propertyfile build.properties
```

### To build an application file by passing parameters in the command-line only (Windows)

Run **commandline codegen** by issuing the following command from the command prompt in your *commandlinecodegen* directory (see Table 23):

```
ant "-D{propertyname1}={propertyvalue1}" "-D{propertyname2}={  
propertyvalue2}" ... "-D{propertynameN}={propertyvalueN}"
```

**Note:** When using the command-line properties method, you must specify values for all required properties. You also may specify values for any optional properties as is appropriate.

To build an application file based on both a property file and by passing parameters in the command-line (Windows)

- 1 Locate the file **build.properties** in your *commandlinecodegen* directory.
- 2 Open the file using a text editor such as *Notepad*. The contents are as follows:

```
commandline.rep.url=http://<host>:<port>/<repositoryname>
commandline.rep.user=
commandline.rep.pass=
commandline.rep.dir=localrepository
commandline.rep.projectName=
commandline.rep.projectDeployName=
commandline.rep.projectDeploymentTag=
commandline.rep.projectBranchName=
commandline.esr.select=
commandline.esr.ignore=
```

- 3 Fill in the properties as described in Table 23.
- 4 Run **commandline codegen** by issuing the following command from the command-line prompt in your *commandlinecodegen* directory:

```
ant -propertyfile build.properties "-D{propertyname1}
={propertyvalue1} -D{propertyname2}={propertyvalue2}
... -D{propertynameN}={propertyvalueN}"
```

**Note:** When using a combination of a property file and command-line properties, the command-line properties override those specified in the property file. You need to list only those properties that you want to override.

To build an application file based on a property file only (Linux/Solaris)

- 1 In the command prompt, go to **Bash** mode.
- 2 Run the commands:

```
Export ANT_HOME
Export JAVA_HOME
dos2unix ant ant
dos2unix build.xml build.xml
dos2unix build.properties build.properties
```

- 3 Locate the file **build.properties** in your *commandlinecodegen* directory.
- 4 Open the file using a text editor. The contents are as follows:

```
commandline.rep.url=http://<host>:<port>/<repositoryname>
commandline.rep.user=
commandline.rep.pass=
commandline.rep.dir=localrepository
commandline.rep.projectName=
commandline.rep.projectDeployName=
commandline.rep.projectDeploymentTag=
commandline.rep.projectBranchName=
commandline.esr.select=
commandline.esr.ignore=
```

- 5 Fill in the properties as described in Table 23 (optional properties may be left blank).
- 6 Run **commandline codegen** by issuing the following command from the command-line prompt in your *commandlinecodegen* directory:

```
sh ant -propertyfile build.properties
```

To build an application file by passing parameters in the command-line only (Linux/Solaris)

- 1 In the command prompt, go to **Bash** mode.

- 2 Run the commands:

```
Export ANT_HOME
Export JAVA_HOME
dos2unix ant ant
dos2unix build.xml build.xml
dos2unix build.properties build.properties
```

- 3 Run **commandline codegen** by issuing the following command from the command prompt in your *commandlinecodegen* directory (see Table 23):

```
sh ant "-D{propertyname1}={propertyvalue1}" "-D{propertyname2}={propertyvalue2}" ... "-D{propertynameN}={propertyvalueN}"
```

**Note:** When using the command-line properties method, you must specify values for all required properties. You also may specify values for any optional properties as is appropriate.

To build an application file based on both a property file and by passing parameters in the command-line (Linux/Solaris)

- 1 In the command prompt, go to **Bash** mode.

- 2 Run the commands:

```
Export ANT_HOME
Export JAVA_HOME
dos2unix ant ant
dos2unix build.xml build.xml
dos2unix build.properties build.properties
```

- 3 Locate the file **build.properties** in your *commandlinecodegen* directory.

- 4 Open the file using a text editor such as *Notepad*. The contents are as follows:

```
commandline.rep.url=http://<host>:<port>/<repositoryname>
commandline.rep.user=
commandline.rep.pass=
commandline.rep.dir=localrepository
commandline.rep.projectName=
commandline.rep.projectDeployName=
commandline.rep.projectDeploymentTag=
commandline.rep.projectBranchName=
commandline.esr.select=
commandline.esr.ignore=
```

- 5 Fill in the properties as described in Table 23.

- 6 Run **commandline codegen** by issuing the following command from the command-line prompt in your *commandlinecodegen* directory:

```
sh ant -propertyfile build.properties "-D{propertyname1}={propertyvalue1} -D{propertyname2}={propertyvalue2} ... -D{propertynameN}={propertyvalueN}"
```

When using a combination of a property file and command-line properties, the command-line properties override those specified in the property file. You need to list only those properties that you want to override.

**To build an application file based on another build script (Linux/Solaris)**

Add the following lines to the build script, filling in the properties as described in Table 23:

```
<ant antfile="remotebuild.xml" dir="absolute path to commandline  
  codegen directory">  
  <property name="commandline.rep.url" value="repositoryurl"/>  
  <property name="commandline.rep.user" value="username"/>  
  <property name="commandline.rep.pass" value="password"/>  
  <property name="commandline.rep.dir" value="directory to download  
    the repository"/>  
  <property name="commandline.rep.projectName" value="projectname"/>  
  <property name="commandline.rep.projectDeployName"  
    value="deployment name"/>  
  <property name="commandline.rep.projectBranchName"  
    value="branchname"/>  
  <property name="commandline.rep.projectDeploymentTag"  
    value="snapshotname"/>  
  <property name="commandline.dir" value="absolute path to  
    commandline codegen directory "/>  
</ant>
```

### 5.11.4 Deploying the Project

Once a Deployment Profile has been defined for your Project and the application file has been built, you can deploy the Project to the selected Environment by clicking the **Deploy** button in the Deployment Editor toolbar (see Figure 109). You can also deploy the Project from Enterprise Manager or from the command line, without having Enterprise Designer running, as described in the *Sun SeeBeyond eGate™ Integrator System Administration Guide*.

**Figure 109** Deploy Button



**Note:** Before you can deploy a copy of an existing Project while the original Project is deployed, you must change both the Project name and the durable subscriber name(s) in the JMS Client configuration (see *Sun SeeBeyond eGate™ Integrator JMS Reference Guide*).

### Deploying to a Sun Java System Application Server

A complete procedure for deploying an application file to a Sun Java System Application Server is described in the *Sun SeeBeyond eGate™ Integrator System Administration Guide*.

# Object Type Definitions I

This chapter introduces Object Type Definitions (OTDs), explains data encoding features and the different marshaling and unmarshaling methods, and describes the features of the OTD Editor.

## What's in This Chapter

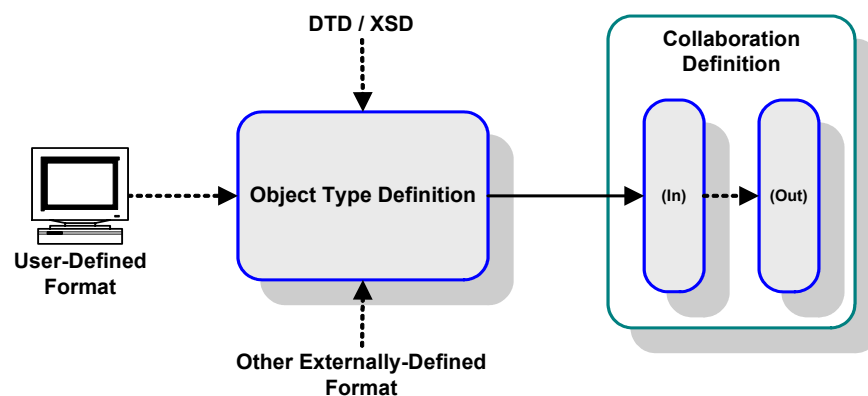
- [Understanding OTDs](#) on page 146
- [Specifying Data Conversion and Encoding](#) on page 151
- [Using the OTD Editor](#) on page 159

---

## 6.1 Understanding OTDs

The basic functionality of a Project is processing and routing messages, which it accomplishes by means of Collaborations. To operate on a message, the Collaboration needs a description of the message structure and format. This description is embodied in an Object Type Definition, or OTD, instances of which are incorporated into the Collaboration Definition.

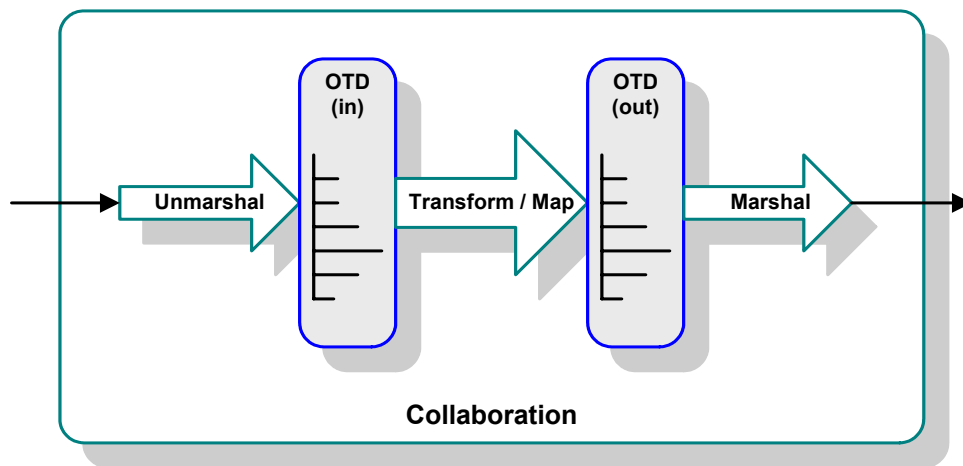
**Figure 110** OTD Generation



In some cases, the OTD can be generated semi-automatically based on the external data source itself; in other cases, you must create the OTD using the tools provided in Enterprise Designer.

An OTD represents the structure of data, and also contains methods that are used in the Collaboration to interact with the data. At run time, a Collaboration typically receives a message as a serialized data stream (see Figure 111). Following its Collaboration Definition, it uses its *unmarshal* method, acting on an instance of a specified OTD, to parse the data and make it accessible through the structure described by that OTD. Then it performs some set of operations on the parsed data and presents the results to another instance of the OTD. Finally, it invokes its *marshal* method, acting on the output OTD instance, to render the contents of the OTD's data structure as a single, serialized data stream for further transport.

**Figure 111** Message Processing



The OTD instance is accessed either directly from Java in a Java-based Collaboration, using accessors resembling JavaBeans, or from BPEL using XPath expressions. In the case of Java, each of the nodes comprising the hierarchy of the data structure has a set of properties with *get* and *set* methods.

### 6.1.1 OTD Fundamentals

Here we introduce some basic concepts and terminology. This information applies to OTDs in general, but is of primary interest in regard to user-defined OTDs.

#### Metadata and OTDs

Metadata is data about data: it can describe how and when and by whom a particular set of data was collected, and how the data is formatted. Metadata is also persistent, in that it endures beyond a single instance of use. The Java™ Composite Application Platform Suite employs a message metadata description feature called an OTD. Using a wizard, you can reference metadata files that describe a message layout and build an OTD that conforms to that description.

Internally, the metadata in the OTD is used to generate a Java archive (**.jar**) file containing a Java implementation of a runtime OTD class, each instance of which can parse and serialize one message of the format described by the metadata. (The generated class and its instances are also loosely referred to as OTDs.)

An OTD contains definitions of:

- A hierarchical data structure forming the representation of a message, internal to the Java™ Composite Application Platform Suite.
- A grammar to scan an input message in its external representation, and rules on mapping the result to the internal representation (an operation known as *unmarshaling* or *parsing*).
- Rules on generating the external representation of an output message from the internal representation (an operation called *marshaling* or *serialization*).

The OTD also contains the methods (send, receive, etc.) associated with given elements and fields within the hierarchical data structure.

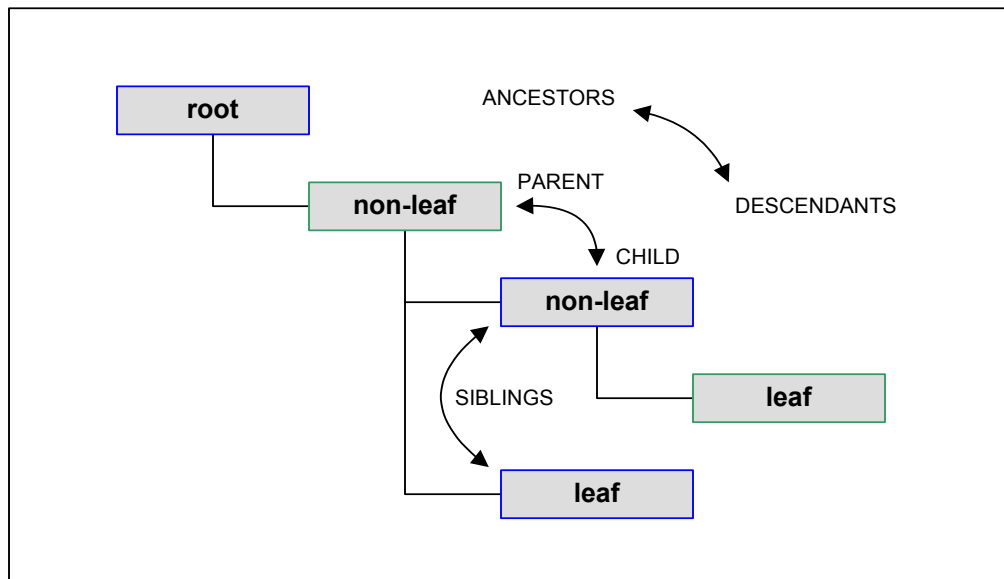
## Data Structure

The run-time data structure is composed of a hierarchical system of *nodes*. These nodes are characterized by terms indicating their relationships with each other:

### Parent, Child, and Sibling Nodes

Any subnode of a given node is called a *child* node, and the given node, in turn, is the child's *parent*. *Sibling* nodes are nodes on the same hierarchical level under the same parent node. Nodes higher than a given node in the same lineage are *ancestors* and those below it are *descendants*.

**Figure 112** OTD Node Nomenclature



### Root and Leaf Nodes

The *root* node is the highest node in the tree structure, and has no parent. This node represents the entire OTD. It may have one or more child nodes, but can never have sibling nodes or be repeating. The properties of the root node (other than its name) cannot be edited.



*Leaf* nodes have no children, and normally carry the actual data from the message. *Non-leaf* nodes, which can have children, provide the framework through which this data is accessed and organized.

### Non-leaf Nodes

Non-leaf nodes represent JavaBean links to the data contained in the leaf nodes (a JavaBean is a reusable software component that works with Java and can be visually manipulated in builder tools such as the OTD Editor). There are two basic types of non-leaf nodes (aside from a root node, which is a special case):

- *Group* nodes, which provide organizational grouping for purposes such as repetition.
- *Choice* nodes, which represent sets of alternatives — only one of which is valid at any given time for an instance of that node. For example, a choice node named **order** might have two children, respectively named **domestic** and **overseas**. For each order instance, only one of these children will be present.

At the JavaBean interface level, a choice node is characterized by having a *hasX* method for each child *X*, and a *choice* method that returns the index of the currently present child (**0** for first child, **1** for second child, and so on). Calling *getX* causes child *X* to be present, creating it if necessary. The *optional* and *repeating* flags cannot be set for the children of a choice node.

### Data Types

The basic data types are *fixed* and *delimited*, but there are variations to these (see [Specifying the Node Type](#) on page 220).

- With fixed-length data, the length of the unit of data is always the same. The position of the data within the message string is described by *byte offset* and *length*.
- With delimited data, the length of the unit of data is variable. Information is separated by a pre-determined system of delimiters defined within the properties of the OTD (see [Specifying Delimiters](#) on page 225).

**Note:** *In order to be compatible with BPEL, an OTD cannot support a pure string-BLOB.*

### Name Properties

An OTD node has two distinct name properties: a *display name*, which is essentially an arbitrary string, and a *Java name*, which is the accessor basename. (For example, if a node has the Java name *ElementX*, then the implementing class for that node will contain a method *getElementX*.) The Java name is normally derived from the display name, modified to suit the restrictions on Java identifiers, and supplied automatically by eGate Integrator.

**Important:** *Do not modify the Java name property.*

As with other Project components, it is essential to manage versions of Object Type Definitions carefully. See [Managing Component Versions](#) on page 78 for descriptions of various version control features applicable to Object Type Definitions.

## 6.1.2 OTD Types

### Externally-Defined OTDs

Externally-defined OTDs can be generated by connecting to external data sources based on any combination of tables and stored procedures or prepared SQL statements. Some of these OTDs are *messagable*, others are API-based. Externally-defined OTDs are read-only; their structures and properties can be viewed using the OTD Editor, but they cannot be modified. See [Creating Externally-Defined OTDs](#) on page 174.

### User-Defined OTDs

User-defined OTD are native to eGate Integrator. You can create a User-defined OTD from scratch using the *User-Defined OTD Wizard* and the *OTD Editor*. You can also create a User-defined OTD from a flat file using the *UD OTD from File Wizard*. User-defined OTDs are read/write — you can add or delete nodes and edit their properties. See [Creating User-Defined OTDs](#) on page 191.

### Predefined OTDs

#### JMS OTD

The Java Message Service (JMS) OTD is a special type of OTD that allows Collaborations to read from and write to topics or queues. It indicates to the Collaboration which topic or queue it expects to receive messages from or send messages to, and allows you to build the JMS business rules. The procedure for adding a JMS OTD to a Collaboration Definition is described in [Using JMS OTDs](#) on page 197.

#### OTD Libraries

Several libraries containing large numbers of OTDs for specific purposes are available to use with eGate Integrator. These OTDs are templates corresponding to message types used by industry-specific data exchange systems and open-source standards. The templates are predefined and can be used as-is, or modified using the OTD Editor — whichever your application requires.

## 6.2 Specifying Data Conversion and Encoding

For eGate Integrator to correctly handle data in byte-oriented protocol, the encoding method for inbound and outbound OTDs and the native code used for parsing must be specified in the OTD properties. If you do not specify otherwise, eGate Integrator assumes UTF-8 to be the encoding method in each case.

Supporting UTF-8 by default allows the use of the Unicode character set in both ASCII and non-ASCII based environments without further specification. eGate Integrator also supports ASCII for English, Japanese, and Korean locales, and the localized country-specific encoding methods shown in Table 24.

The data encoding you specify when configuring the OTD modifies the Java methods used for marshaling and unmarshaling. The marshaling and unmarshaling processes differ from one another depending upon which Java method you use, and whether you are marshaling to or unmarshaling from `byte[]` or `string` fields. The diagrams shown on the following pages illustrate these differences. Note that the term *encoding* has a specific meaning in this context.

### 6.2.1 Encoding Options

The encoding options available to you depend on the locale specified by your version of eGate Integrator. UTF-8 is the default in all locales.

**Table 24** Encoding Options According to Locale

English	Japanese	Korean	Simplified Chinese	Traditional Chinese
UTF-8	UTF-8	UTF-8	UTF-8	UTF-8
ASCII	ASCII	ASCII	GB2312	Big5
	EUC-JP	EUC-KR		
	SJIS	MS949		
	MS932			

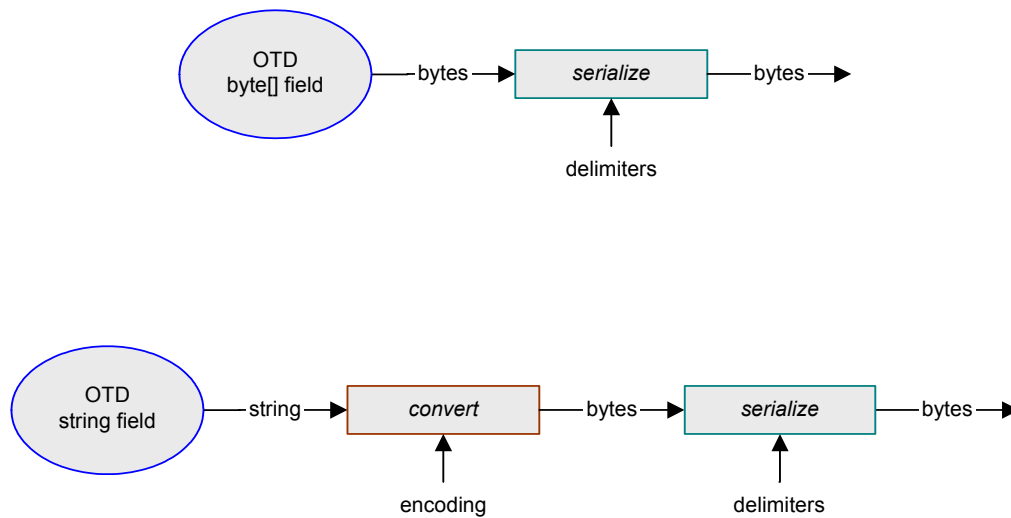
## 6.2.2 Marshaling and Unmarshaling Methods

The parsing and serializing operations require data to be in byte-array form, so different methods for marshaling and unmarshaling data must be used to accommodate different input and output data formats.

### **marshal()**

The simplest marshaling method, **marshal()**, requires *encoding* only when marshaling from a string field, since the parser requires the data in bytes and the output is also in bytes (see Figure 113).

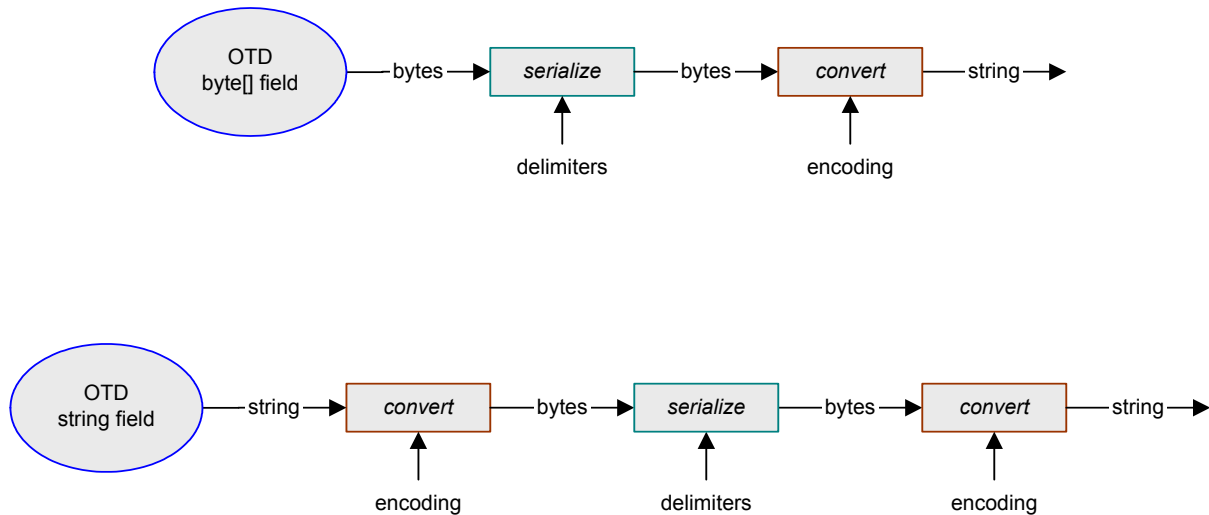
**Figure 113** marshal()



## marshalToString()

The **marshalToString()** method requires *encoding* to produce an output string after marshaling from a byte[] field (see Figure 114). This method also requires *encoding* when marshaling from a string field, since the parser requires the data in bytes, and *encoding* again to produce an output string.

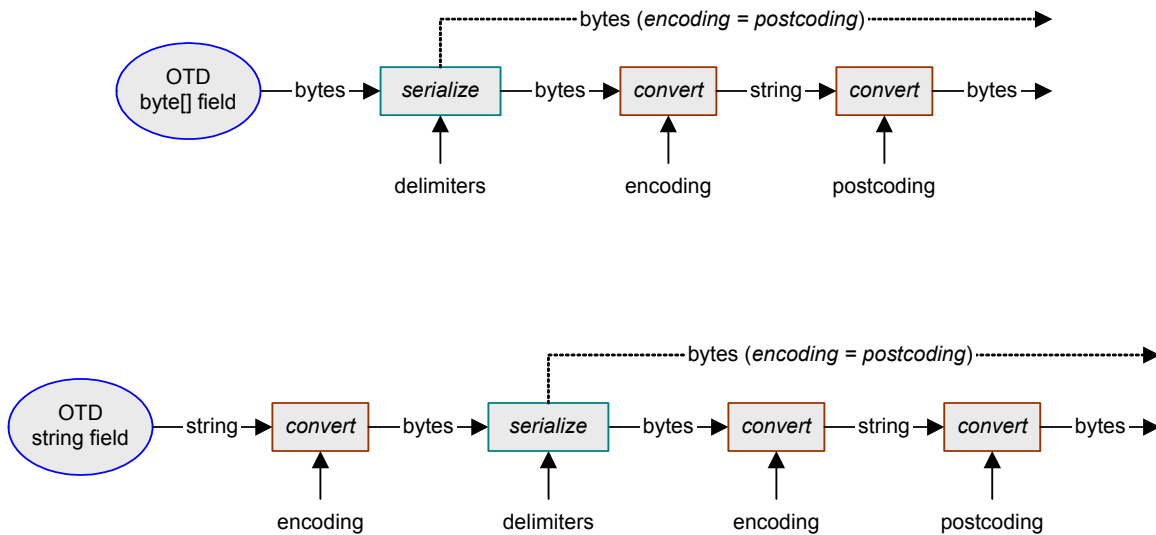
Figure 114 marshalToString()



## marshalToBytes()

The **marshalToBytes()** method requires *encoding* to produce bytes after marshaling from a string field (see Figure 115). Following serialization, this method also requires *encoding* and *postcoding* to produce an output (in bytes) having a different format from that used by the parser. If the same format is desired, then the *postcoding* property is left undefined, the *encoding* property is substituted by default, and the double conversion is bypassed.

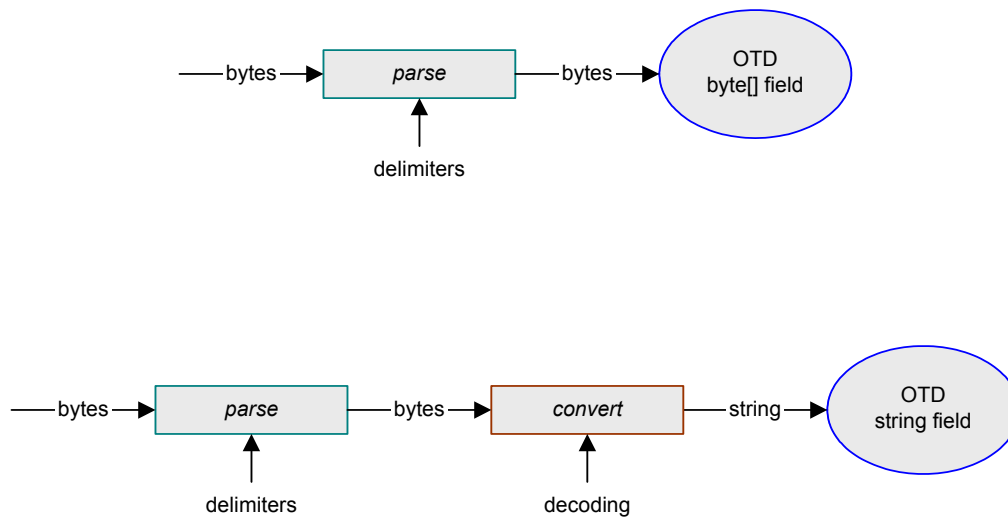
**Figure 115** marshalToBytes()



## unmarshal()

The simplest unmarshaling method, **unmarshal()**, requires *decoding* only when unmarshaling to a string field, since the input is in bytes as required by the parser (see Figure 116).

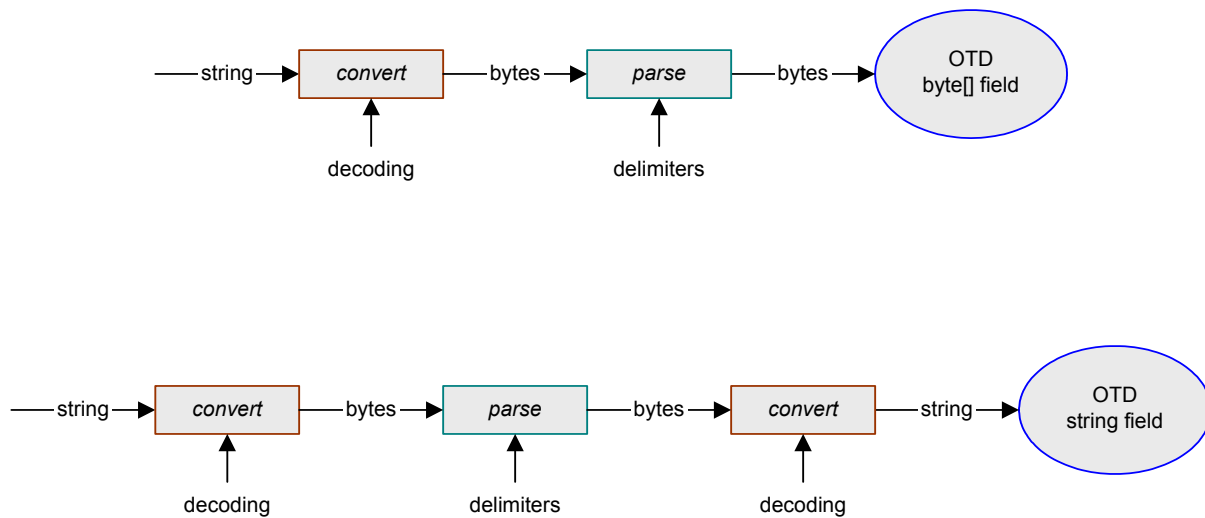
**Figure 116** unmarshal()



## unmarshalFromString()

The **unmarshalFromString()** method requires *decoding* of the input string, since the parser requires the data in bytes (see Figure 117). This method requires a second *decoding* when unmarshaling to a string field.

**Figure 117** unmarshalFromString()

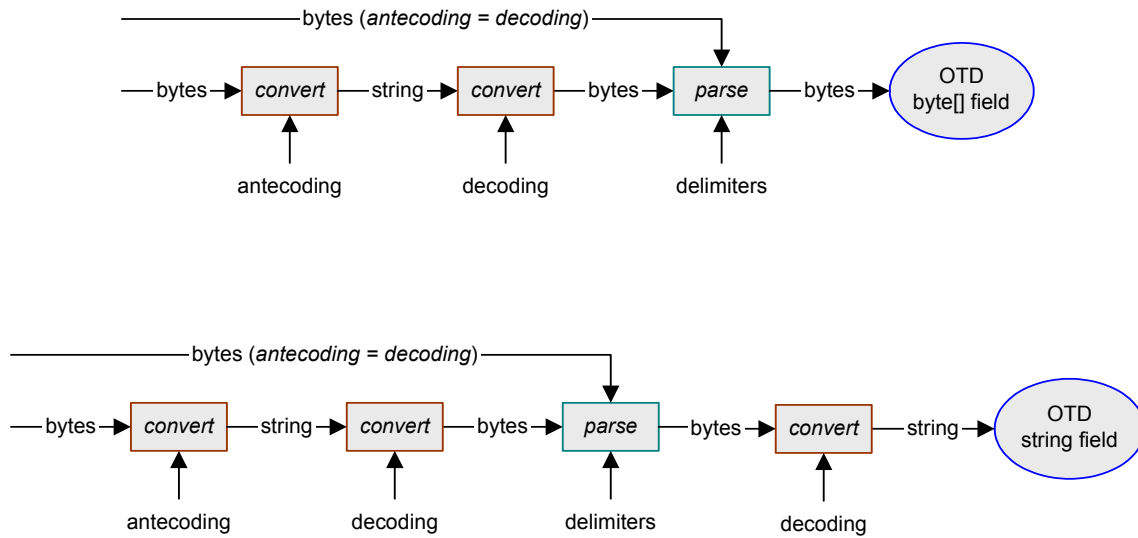




## unmarshalFromBytes()

The **unmarshalFromBytes()** method requires *antecoding* and *decoding* if the input data has a different byte format from that used by the parser (see Figure 118). If the same format is desired, then the *antecoding* property is left undefined, the *decoding* property is substituted by default, and the double conversion is bypassed. After parsing, this method requires further *decoding* if unmarshaling to a string field.

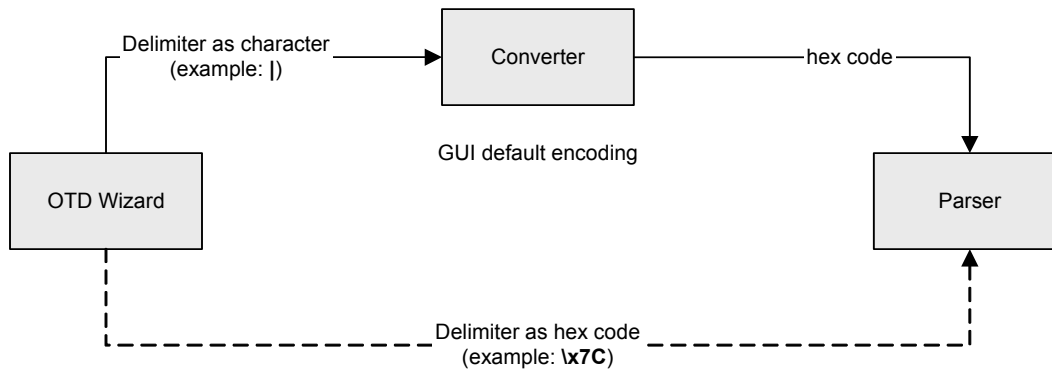
**Figure 118** unmarshalFromBytes()



### 6.2.3 Setting Delimiters

Figure 119 illustrates how the delimiter gets set and passed into the parser. For example, if you select a delimiter in the OTD Editor by hex code (such as `\x7C`), it is passed directly into the parser. If you type the delimiter in as a pipe (`|`), however, then the pipe character is first converted to hex code, using the GUI's encoding, and then sent to the parser.

**Figure 119** Setting Delimiters



## 6.3 Using the OTD Editor

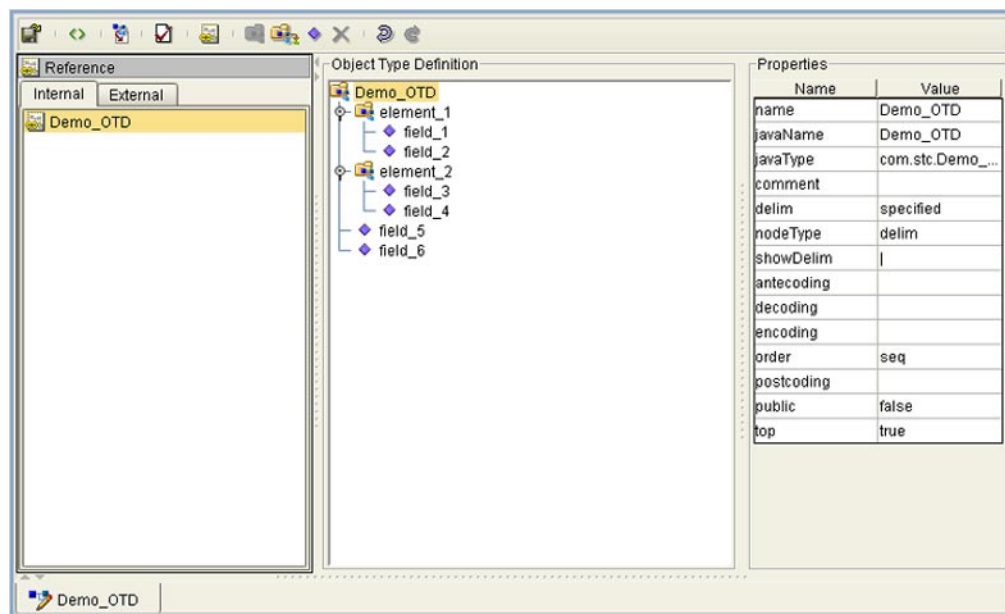
The OTD Editor displays the structure of the selected Object Type Definition (OTD) and allows you to verify its operation with a built-in tester. You can also use the editor to create and modify User-Defined OTDs.

After you create an OTD file using the OTD Wizard, the OTD Editor appears in the editor panel of the Enterprise Designer, as shown in Figure 120. You can also invoke the OTD Editor by selecting **Open** in the context menu for an existing OTD in the Project Explorer. OTDs are saved to the Project automatically.

Use of the editing features of the OTD Editor are described in [Object Type Definitions III](#) on page 204.

**Note:** Remember that externally-defined OTDs are read-only, and cannot be edited. You can, however, test them to verify correctness of the build.

**Figure 120** OTD Editor





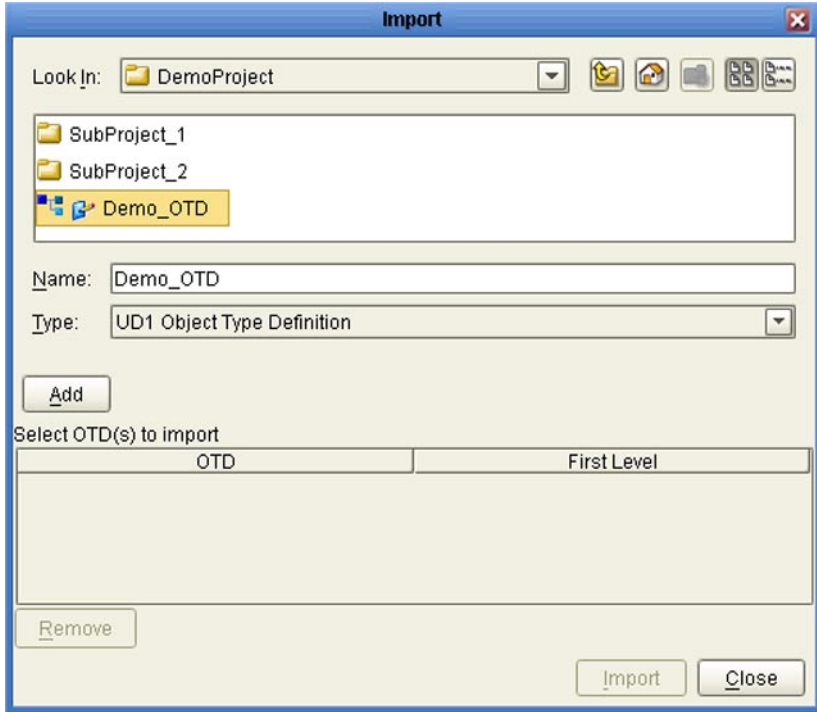



**Table 25** OTD Editor User Interface








Panel Name	Description
Reference	This panel contains internal and external templates for the OTD file.
Object Type Definition	This panel displays each field and element included in the OTD file.
Properties	This panel displays details about the OTD file or field selected in the Object Type Definition list.

## 6.3.1 OTD Editor Toolbar Icons

**Table 26** OTD Editor Toolbar Icons

Icon	Command	Function
	Save as New Name to Repository	<p>Presents a dialog with which you can save the current OTD under a new name to any Project or Subproject in the Repository.</p> 
	Generate Code	
	Import OTD to External Template	<p>Presents a dialog with which you can import an OTD as an external template (it will appear in the <i>Reference: External</i> tab).</p> 
	Tester	Displays/hides the OTD Tester.

**Table 26** OTD Editor Toolbar Icons

Icon	Command	Function
	Toggle Reference Tab Panel	Displays/hides the Reference panel.
	Element As Peer	Adds a new element at the same hierarchical level as the selected node.
	Element (As Child)	Adds a new element as a child of the selected node.
	Field	Adds a new field as a child of the selected node.
	Delete	Removes the selected element or field node.
	Undo	Reverses the previous action, up to a limit of 100 steps during the current editing session. Saving the OTD ends the ability to undo.
	Redo	Restores the previous action that was undone using the undo command, up to a limit of 100 steps during the current editing session. Saving the OTD ends the ability to undo or redo.

## 6.3.2 OTD Tester

The OTD Tester facility allows you to simulate the operation of a Collaboration containing a specific OTD, thereby checking the correctness of the OTD during the design phase. You can enter input data values, perform the unmarshal and marshal operations, and also manipulate the OTD tree structure as a Collaboration might do by using the *Add Instance* and *Delete Instance* buttons. By using these latter features, you can prepare an output data file that can then be used as an input data file for testing purposes.

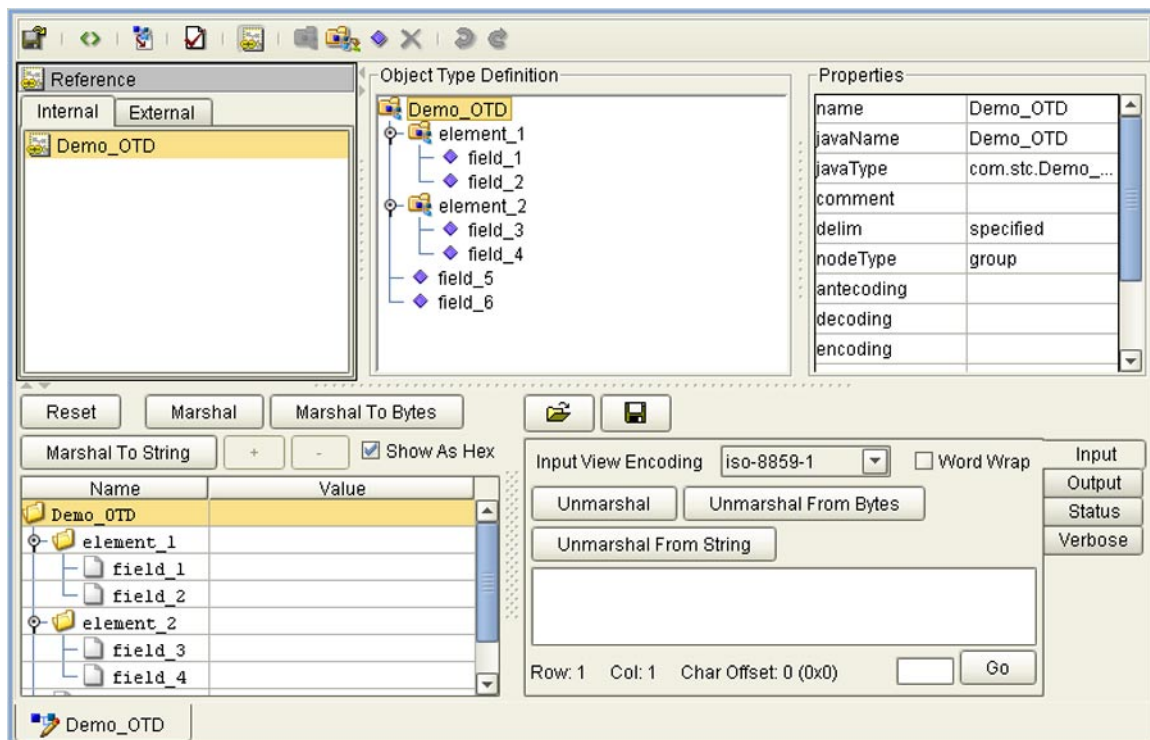
Making proper use of these capabilities, you can prevent data errors at run time by verifying that all required data elements are available and that all used data formats are correct.

Clicking the **Tester** icon (see Figure 121) in the OTD Editor toolbar saves the currently displayed OTD to the Repository and displays the OTD Tester in the lower part of the editor, as shown in Figure 122. Clicking the icon again hides the Tester. Use of the OTD Tester is described in [Using the OTD Tester](#) on page 169.

**Figure 121** Tester Icon



**Figure 122** OTD Tester Interface



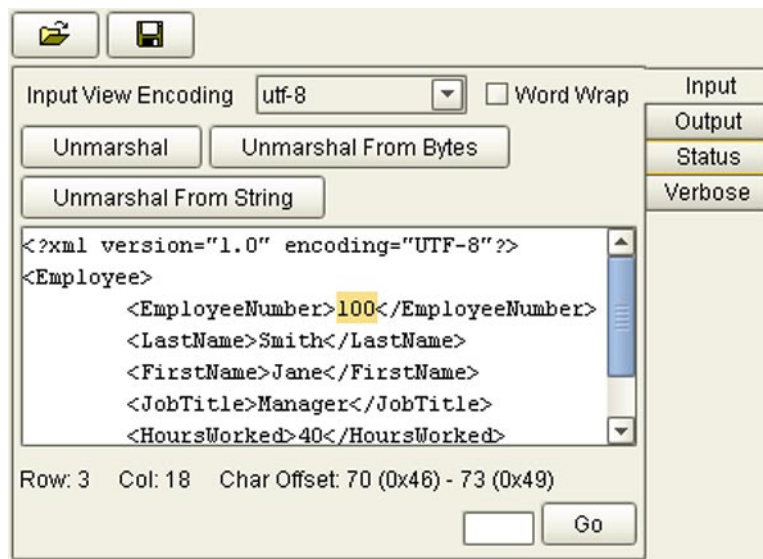
The data display panel on the right has four data display modes, selectable by tabs:

- Input
- Output
- Status
- Verbose

## Input Panel

The *Input* panel is displayed by default. If you click any character in the data appearing in the *Input* or *Output* panel, the row, column, and offset for that character is displayed below the panel. Double-clicking highlights the data field, and the offset range for the field is displayed (see Figure 123).

**Figure 123** Data Field Display

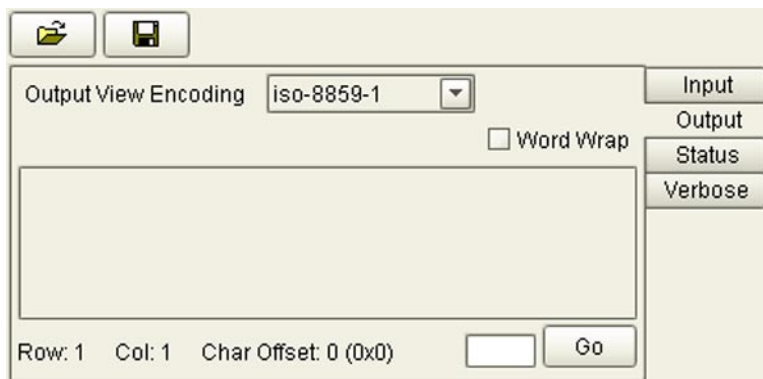


The *Input* panel also has buttons for running the tester in the three different unmarshaling modes (see [OTD Tester Controls](#) on page 166). Usage is described in [To test data unmarshaling functionality](#) on page 169.

## Output Panel

The *Output* panel displays the results of a marshaling operation. Usage is described in [To test data marshaling functionality](#) on page 170.

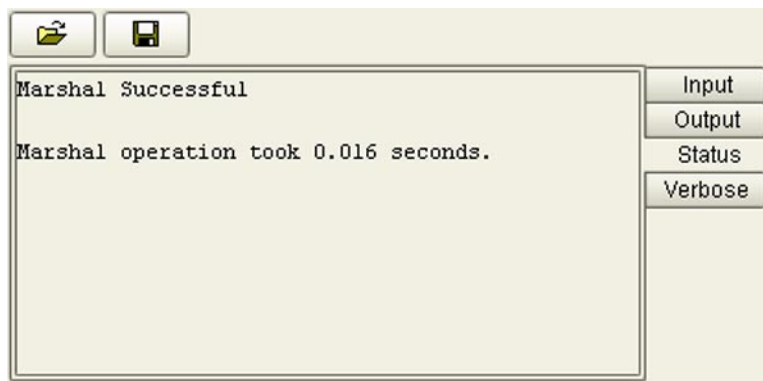
**Figure 124** Output Panel



## Status Panel

The *Status* panel shows the status of an attempted operation. Figure 125 shows the results of a successful operation; if the operation had failed, error information would be shown.

**Figure 125** Status Panel

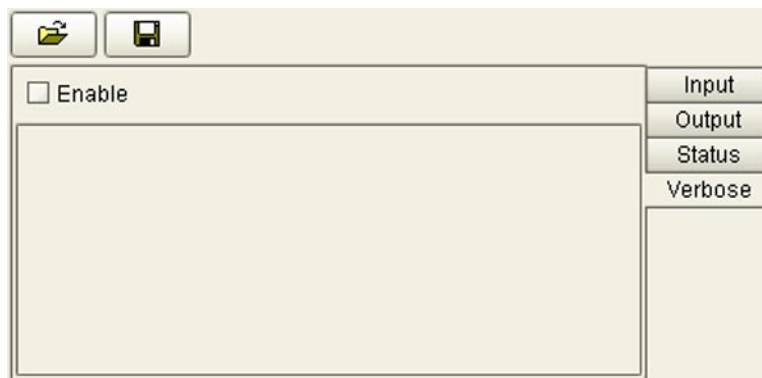




## Verbose Panel

For selected OTDs, the *Verbose* option provides a trace of parsing actions during the unmarshal process to aid in debugging the OTD structure. Selecting the **Enable** check box causes parsing information to appear on the panel (see Figure 126). The format and content of the data display are OTD-specific.


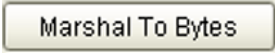
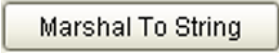





**Figure 126** Verbose Panel




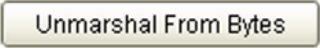


## OTD Tester Controls

A large number of buttons, icons, and text appear in the OTD Tester interface. Table 27 contains a list of these artifacts, with brief descriptions of their respective functions.

**Table 27** OTD Tester Controls

Button/Icon	Command	Function
		Runs the tester with the data values entered in the table; the data is marshaled from the OTD tree using the <i>marshal</i> method (see Table 28) and displayed in the <i>Output</i> panel.
		Runs the tester with the data values entered in the table; the data is marshaled from the OTD tree using the <i>marshalToBytes</i> method (see Table 28) and displayed in the <i>Output</i> panel.
		Runs the tester with the data values entered in the table; the data is marshaled from the OTD tree using the <i>marshalToString</i> method (see Table 28) and displayed in the <i>Output</i> panel.
	Add Instance	Adds an optional node (button is deactivated when an optional node already exists) or an additional instance of the selected repeating node. The repeating node is added immediately following the selected instance. You can also select the length field for a repeating node, in which case the new node will be added as the first instance.
	Delete Instance	Deletes the selected instance of an optional or repeating node. Button is deactivated if the optional instance does not exist.
		Resets the tester, repopulating the OTD with the current data from the <i>Input</i> panel.
(None)	Show as hex	When checked, the values for byte[] nodes are shown in hexadecimal format. When not checked, the values are shown as standard alphanumeric text (default setting).
	Open File	Presents the <i>Open File</i> dialog, where you can select a data file to use for testing the currently open OTD, along with the encoding to use when reading the file. The data is then loaded into the <i>Input</i> panel of the tester.
	Save File	Presents the <i>Save File</i> dialog, with which you can save the data from the selected OTD Tester panel, along with the encoding to use.

**Table 27** OTD Tester Controls

Button/Icon	Command	Function
		Runs the tester, unmarshaling the data from the <i>Input</i> panel to the OTD tree using the <i>unmarshal</i> method (see Table 28).
		Runs the tester, unmarshaling the data from the <i>Input</i> panel to the OTD tree using the <i>unmarshalFromBytes</i> method (see Table 28).
		Runs the tester, unmarshaling the data from the <i>Input</i> panel to the OTD tree using the <i>unmarshalFromString</i> method (see Table 28).
(None)	Word Wrap	Selecting this check box causes the content of the <i>Input</i> or <i>Output</i> panel to wrap, avoiding the need to scroll horizontally to read each line. (At times this is desirable, at other times, not.)
(None)	Input/Output View Encoding	See <a href="#">Data Encoding</a> on page 168.
		Typing a character offset into the text box and clicking <b>Go</b> advances the cursor to that point in the currently selected line. (In the <i>Output</i> panel, the cursor position is indicated by a faint, yellow, vertical line that can be difficult to see.)

## Marshal/Unmarshal Processes

**Table 28** Marshal and Unmarshal Process Differences

Method	Basic Process
unmarshal	parsing
unmarshalFromBytes	antecoding > decoding > parsing
unmarshalFromString	decoding > parsing
marshal	serialization
marshalToBytes	serialization > encoding > postcoding
marshalToString	serialization > encoding

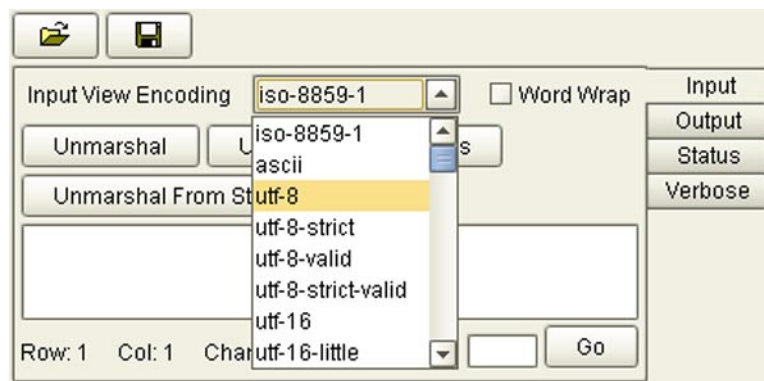
**Note:** See [Marshaling and Unmarshaling Methods](#) on page 152 for expanded descriptions of the marshaling and unmarshaling processes and the associated default conditions.

## Data Encoding

### Unmarshaling

If you call **unmarshalFromBytes**, the string data contained in the input panel will be converted to a byte array using the character encoding selected in the *Input View Encoding* drop-down menu shown in Figure 127.

**Figure 127** Input View Encoding Option Menu



### Marshaling

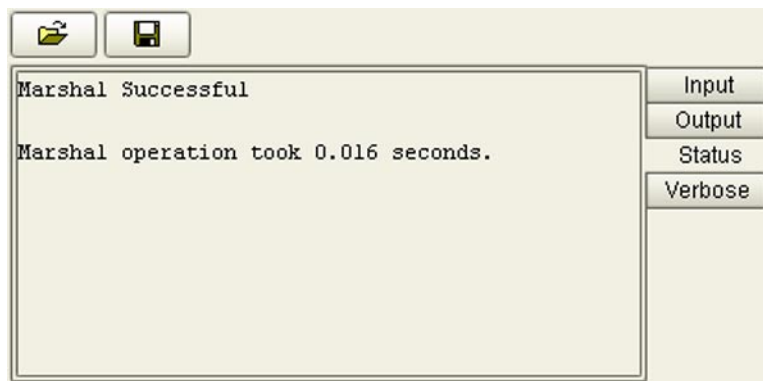
If you call **marshalToBytes**, the byte array returned by the method will be converted to a string using the character encoding selected in the *Output View Encoding* drop-down menu, which is identical to the *Input View Encoding* menu shown in Figure 127. That string will then be displayed in the *Output* panel.

## Using the OTD Tester

To test data unmarshaling functionality

- 1 Open or create an OTD.
- 2 Click the **Tester** icon to display the OTD Tester (see [Table 26 on page 160](#)).
- 3 Click the **Open File** button to display the *Open File* dialog.
- 4 Provide the input test data by selecting a data file.
- 5 Click **Open**. The data will appear in the *Input* panel of the tester.
- 6 Click the desired unmarshal command to unmarshal the data from the *Input* panel to the OTD tree.
- 7 Verify the unmarshal process by checking the values for each element for correctness.
- 8 Save your input test data to a file for re-use by selecting the *Input* panel and clicking the **Save** icon.
- 9 You can also change your test data in the *Input* panel, then re-test the OTD by clicking the **Refresh** icon (see [Table 27 on page 166](#)) to repopulate your OTD object elements with the new values.
- 10 If there are errors in your input data, the **Status** panel is automatically invoked, showing the appropriate error messages. Confirmation of correct operation is also reported, as shown in Figure 128.

**Figure 128** Data Display - Status Panel



- 11 For selected OTDs, the **Verbose** option provides a trace of parsing actions during the unmarshal process to aid in debugging the OTD structure. Selecting the **Verbose** check box causes parsing information to appear on the *Verbose* panel. The format and content of the data display are OTD-specific.

### To test data marshaling functionality

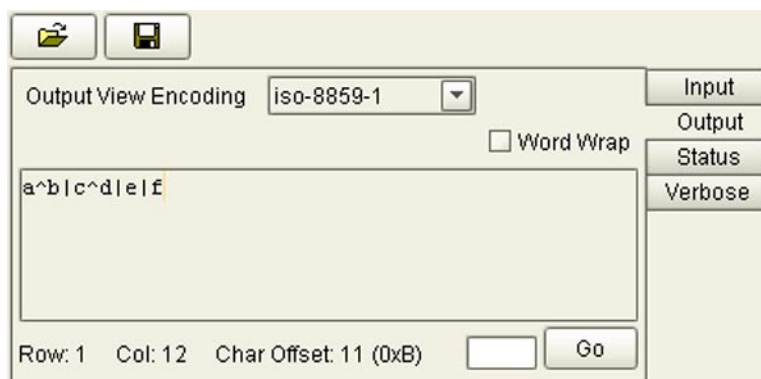
- 1 Open or create an OTD.
- 2 Click the **Tester** icon to display the OTD Tester.
- 3 Enter or change data values for each node in the **Value** column of the node table (see Figure 129). Use the Add/Delete Instance (+/-) buttons to add or remove instances where appropriate.

**Figure 129** OTD Tester Node Table

Name	Value
Demo_OTD	
element_1	
field_1	a
field_2	b
element_2	
field_3	c
field_4	d
field_5	e
field_6	f

- 4 Specify the appropriate output encoding, as described in [Data Encoding](#) on page 168.
- 5 Click the desired marshal button to marshal (serialize) the data, as described in [Table 27 on page 166](#).
- 6 The output is displayed in the *Output* panel (see Figure 130).

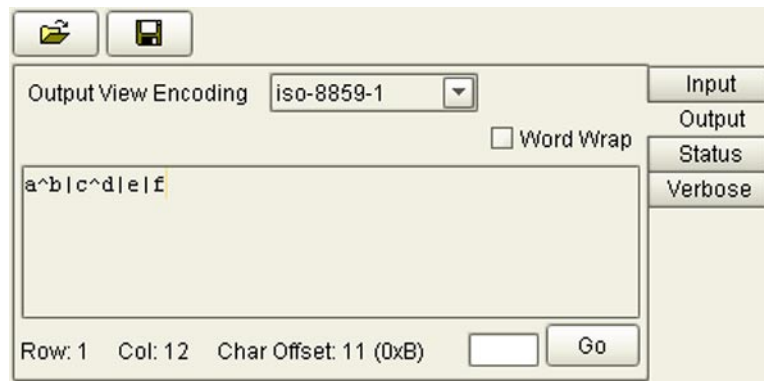
**Figure 130** Serialized Data in Output Panel



### To save tester information to a file

- 1 Click the tab for the panel containing the data you want to save to a file, for example *Output* (see Figure 130).

**Figure 131** Saving Test Data



- 2 Click the **Save File** button to display the *Save File* dialog.
- 3 Specify the desired location for the file.
- 4 Click **Save**.

# Object Type Definitions II

This chapter outlines procedures for creating both externally-defined and user-defined OTDs, and adding predefined OTDs to Collaboration Definitions.

## What's in This Chapter

- [Overview](#) on page 172
- [Using the New OTD Wizard](#) on page 173
- [Creating Externally-Defined OTDs](#) on page 174
- [Creating User-Defined OTDs](#) on page 191
- [Using Predefined OTDs](#) on page 196
- [Using the OTD Context Menu](#) on page 202

---

## 7.1 Overview

As explained in the previous chapter, OTDs are of three basic types:

- Externally-defined
- User-defined
- Predefined

For externally-defined and user-defined OTDs, wizards are provided in Enterprise Designer to guide you through the OTD generation process. These wizards call back-end builders that actually implement the building of the code, based on the provided information. Note that some OTD Libraries fall into the externally-defined class, and employ this type of wizard; others fall into the predefined class, and are used more directly.

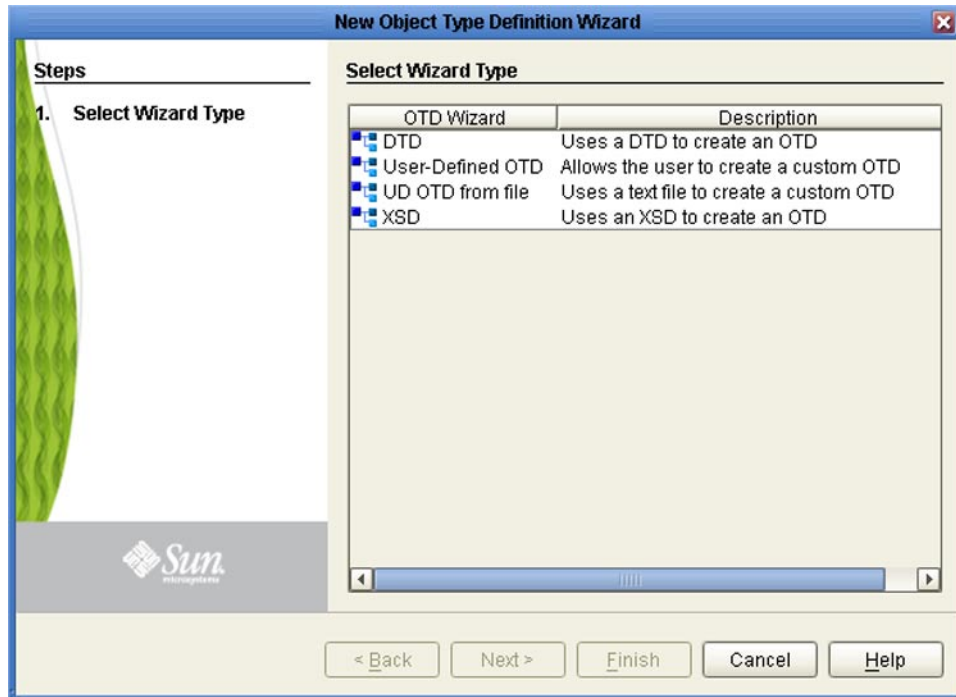
Predefined OTDs do not require generation, but wizards are usually used to implement them. For example, you add a JMS OTD to a Collaboration Definition when you create the Collaboration Definition itself using the Collaboration Definition Wizard.



## 7.2 Using the New OTD Wizard

Right-clicking a Project in Enterprise Explorer displays the Project context menu, from which you can select **New > Object Type Definition** to display the OTD wizard selection dialog, shown in Figure 132. This initial dialog allows you to select the specific type of OTD Wizard needed for your application.

**Figure 132** OTD Wizard Selection Dialog



The basic wizards supplied with eGate Integrator are described in the following sections:

### For externally-defined OTDs

- [Using the DTD Wizard](#) on page 174
- [Using the XSD Wizard](#) on page 181

### For user-defined OTDs

- [Using the User-Defined OTD Wizard](#) on page 191
- [Using the UD OTD from File Wizard](#) on page 193

Additional OTD Wizards are supplied with eGate Integrator add-on components, and are described in the User's Guides for the specific products. When these products are installed, the OTD Wizards are added to the list shown in Figure 132.

For externally-defined OTDs, the OTD Wizards guide you through the OTD generation process, and then invoke the OTD Editor where you can view and check the result. In the case of user-defined OTDs, the Wizard simply asks for an OTD name and then opens the editor, in which you actually create the OTD yourself.

## 7.3 Creating Externally-Defined OTDs

If you have an existing document type definition (DTD) or XML schema definition (XSD), you can build an OTD based on either by using the appropriate wizard. Wizards for creating both DTD and XSD-based OTDs are included with eGate Integrator.

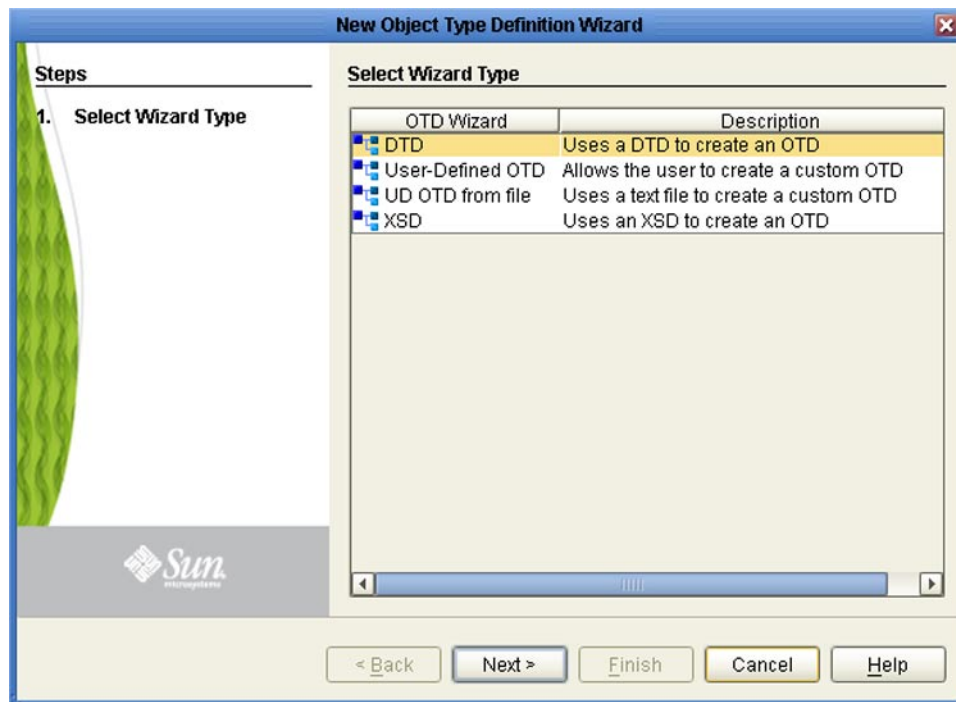
### 7.3.1 Using the DTD Wizard

**Note:** The *reset()* method resets the DTD OTD to its initial conditions, including any default values. It is highly recommended that this method be used in your Collaborations whenever marshaling in a loop, to conserve system resources.

To create an OTD file from a DTD file

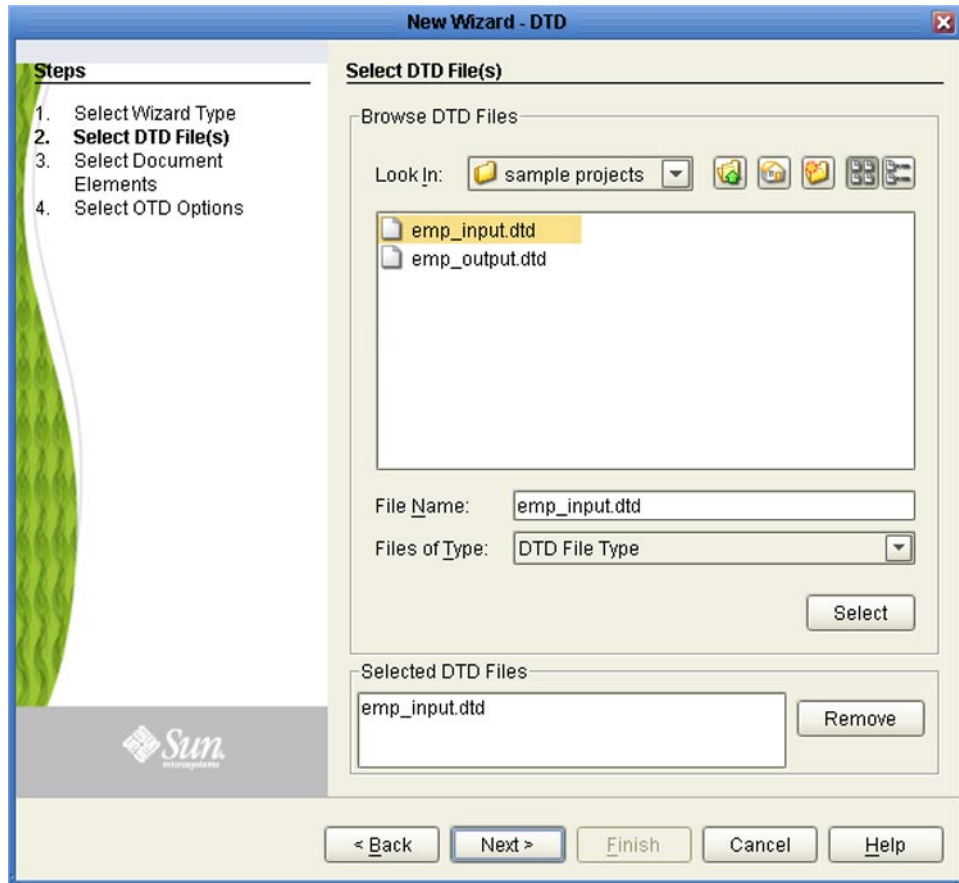
- 1 In the *Select Wizard Type* dialog, select **DTD** from the *OTD Wizard* list (see Figure 133) to create an OTD from a Data Type Definition (DTD) file.

**Figure 133** OTD Wizard Selection: DTD Wizard



- 2 Click **Next** to display the *Select DTD File(s)* dialog, shown in Figure 134.

**Figure 134** Select DTD File(s) Dialog



- 3 In the *Look In* drop-down list, navigate to the DTD file or files that you want to use to create the OTD. Click **Select** to add the files to the *List of Selected DTDs*.

**Note:** Make sure the file name does not contain any blank spaces.

- ♦ If the DTD file does not contain all information required for building an OTD (such as element definitions) a warning box such as that shown in Figure 135 will be displayed.

**Figure 135** Cannot Create OTD Warning Box

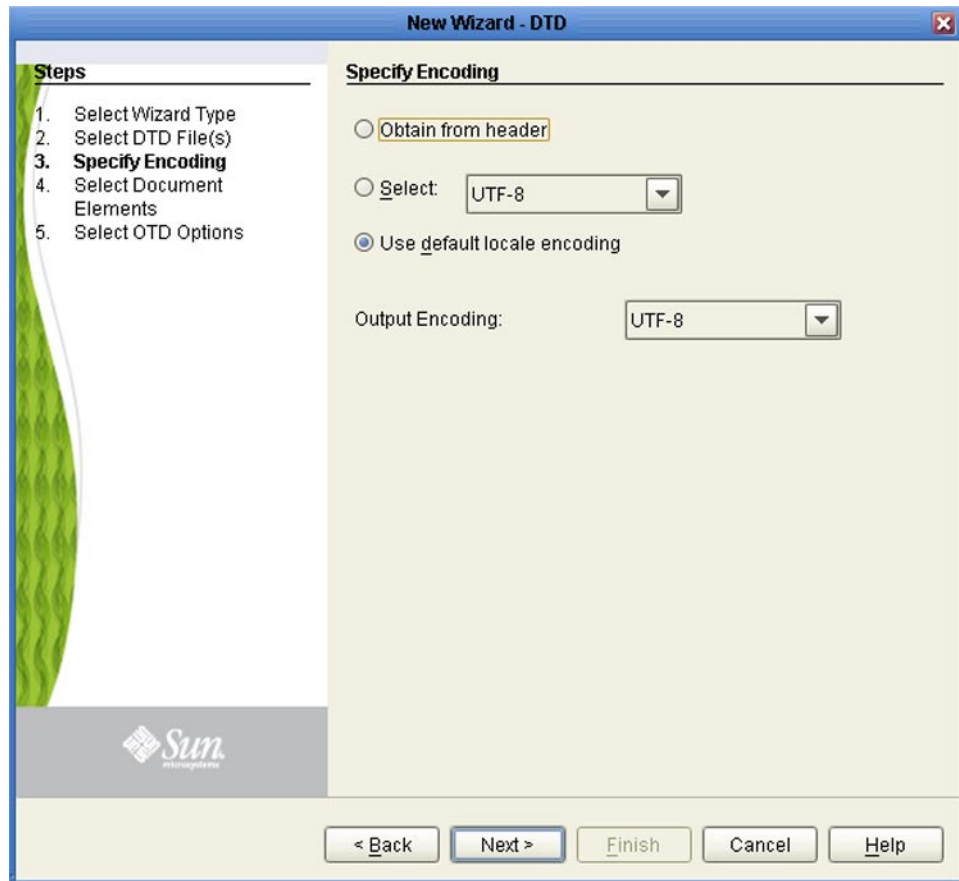


- 4 If you are using the extended language options (see [Setting Up Options](#) on page 53) click **Next** to display the *Specify Encoding* dialog, shown in Figure 136. If

you are *not* using the extended language options, this step will not appear in the Wizard—proceed to step 7.

**Note:** This OTD encoding is currently supported for Java Collaboration Definitions only.

**Figure 136** Select Encoding Dialog

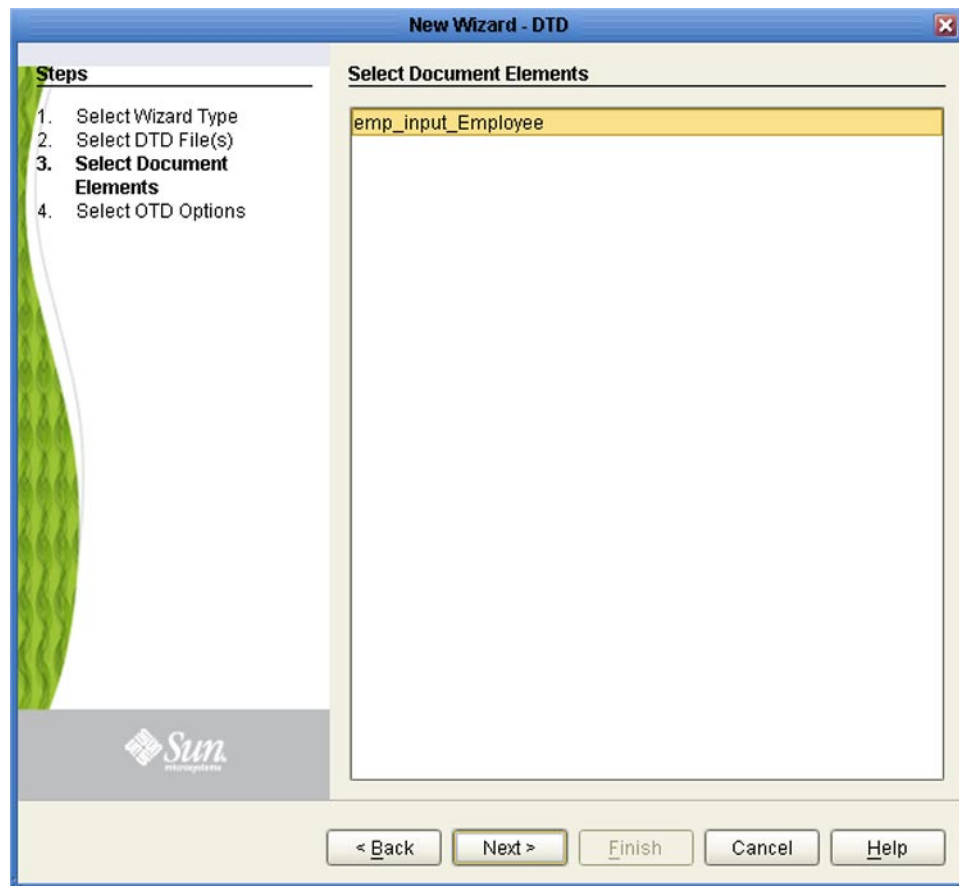


- 5 Click the appropriate option button to specify the data encoding for the input XML file:
  - ♦ **Obtain from header** does not apply to DTD OTDs.
  - ♦ Click **Select** to explicitly specify the encoding from a drop-down list, the contents of which depends upon the locale (see [Specifying Data Conversion and Encoding](#) on page 151).
  - ♦ Select **Use default locale encoding** to use the default encoding for the locale.
- 6 Select **Output Encoding** to specify the header encoding of the output XML file from a drop-down list, the contents of which depends upon the locale (see [Specifying Data Conversion and Encoding](#) on page 151).

**Note:** For the Output Encoding property to apply to an imported project from a previous eGate Integrator release, you must recreate the OTD and specify the new encoding option.

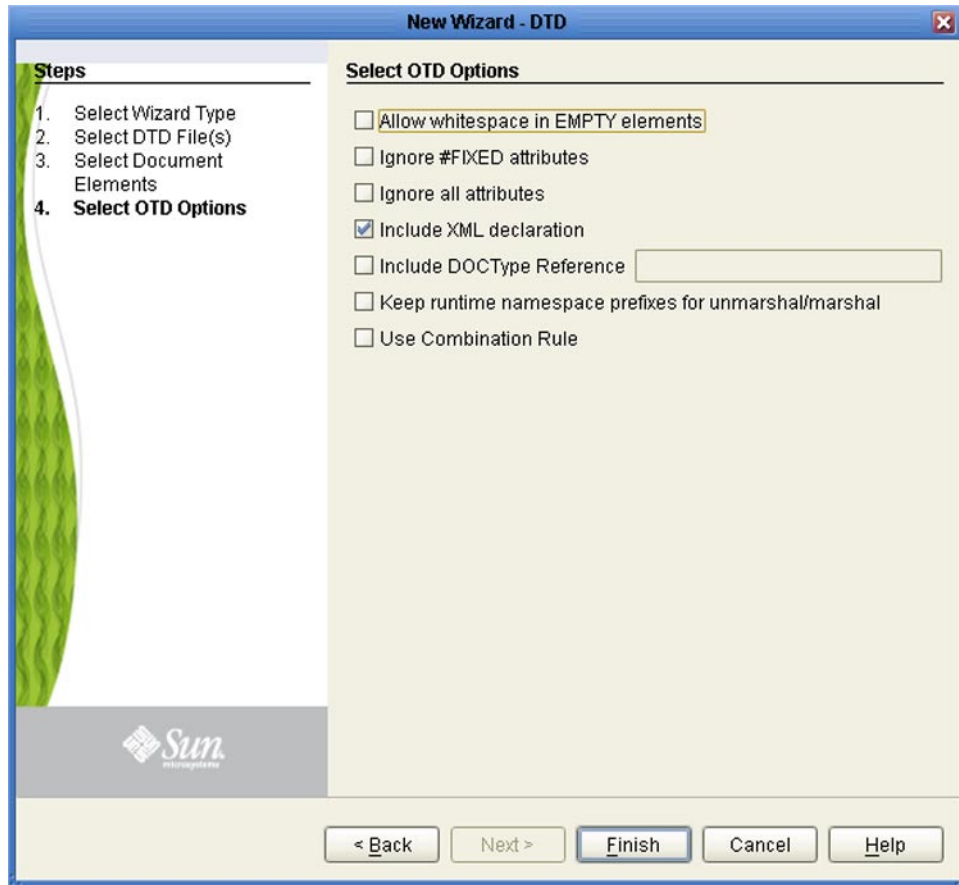
- 7 Click **Next** to display the *Select Document Elements* dialog, shown in Figure 137.

**Figure 137** Select Document Elements Dialog



- 8 Select the elements of the document that you want to include in the OTD.
- 9 Click **Next** to display the *Select OTD Options* dialog, shown in Figure 138.

**Figure 138** Select OTD Options Dialog



- 10 Select the check boxes next to the OTD options you want to enable (see Table 29).
- 11 Click **Finish** to add the OTD to your Project. The OTD Editor will open, displaying the new OTD (see [Using the OTD Editor](#) on page 159).

**Table 29** DTD OTD Options

Option	Description
Allow whitespace in EMPTY elements	If an element is defined as EMPTY, this option controls whether or not white spaces are allowed within the element in the XML instance document.
Ignore #FIXED attributes	Controls whether or not attributes defined as FIXED are ignored during the unmarshal and marshal processes. <ul style="list-style-type: none"> <li>▪ If this option is <i>not</i> selected, the attribute is recognized and saved into the OTD's runtime structure during the unmarshal process, and also appears in the output during the marshal process.</li> <li>▪ If this option is selected, the attribute is ignored and neither of the above occurs.</li> </ul>

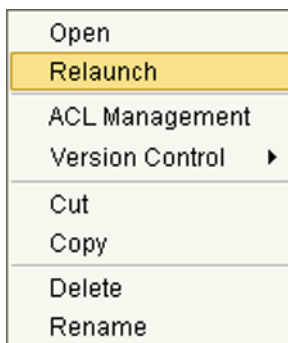
Option	Description
Ignore all attributes	Controls whether or not all attributes are ignored during the unmarshal and marshal processes. If both this option and the <i>Keep runtime namespace prefixes</i> option (below) are selected, only namespace attributes will be handled during the unmarshal process and consequently presented in the output during the marshal process. (The <i>namespace</i> attribute has the form <b>xmlns:XX</b> .)
Include XML declaration	Controls whether or not the XML declaration <b>&lt;?xml version="1.0" encoding="....." ?&gt;</b> appears in the output during the marshal process. Option is checked by default.
Include DOC Type Reference	Controls whether or not the <b>"&lt;!DOCTYPE ..."</b> string appears in the output during the marshal process.
Keep runtime namespace prefixes for unmarshal/marshal	<p>Controls whether or not the namespace prefixes used during the marshal process are identical to those used in the unmarshal process.</p> <ul style="list-style-type: none"> <li>▪ If this option is selected, all namespace attributes will be preserved once they appear in the XML instance document, and the namespace prefixes used in the marshal process will be exactly as they were presented in the XML document during the unmarshal process.</li> <li>▪ If this option is <i>not</i> selected, then the namespace prefixes used in the marshal process might be different than the ones presented in the XML document during the unmarshal process (for example, the namespace prefixes that are presented in the XSD file might be used).</li> </ul> <p><b>Note:</b> A consequence of selecting this option is that if there is no unmarshal process performed before the marshal process, then there will be no namespace attributes presented in the output (see the comment for the option below).</p>
Use Combination Rule	Not currently used.

### To redefine an existing DTD OTD

You can redefine a newly-created DTD OTD by selecting the **Relaunch** option from its context menu in Project Explorer (see Figure 146). This relaunches the DTD Wizard, and allows you to reselect files and options while preserving the original OID.

*Note: Imported OTDs cannot be modified in this manner.*

**Figure 139** OTD Context Menu



When using this option, you should leave the existing node structure intact and append any newly-created nodes to the end of the existing node list, or create a new sibling node list.



### 7.3.2 Using the XSD Wizard

You can create an XSD OTD based on an existing XSD file by using the XSD OTD Wizard. You can then use the resulting OTD to form either a Java or XSLT-based Collaboration Definition. A Java-based Collaboration Definition can further be exposed as a web service (see [To create a Java-based Collaboration Definition](#) on page 316 and [Developing Web Services](#) on page 413).

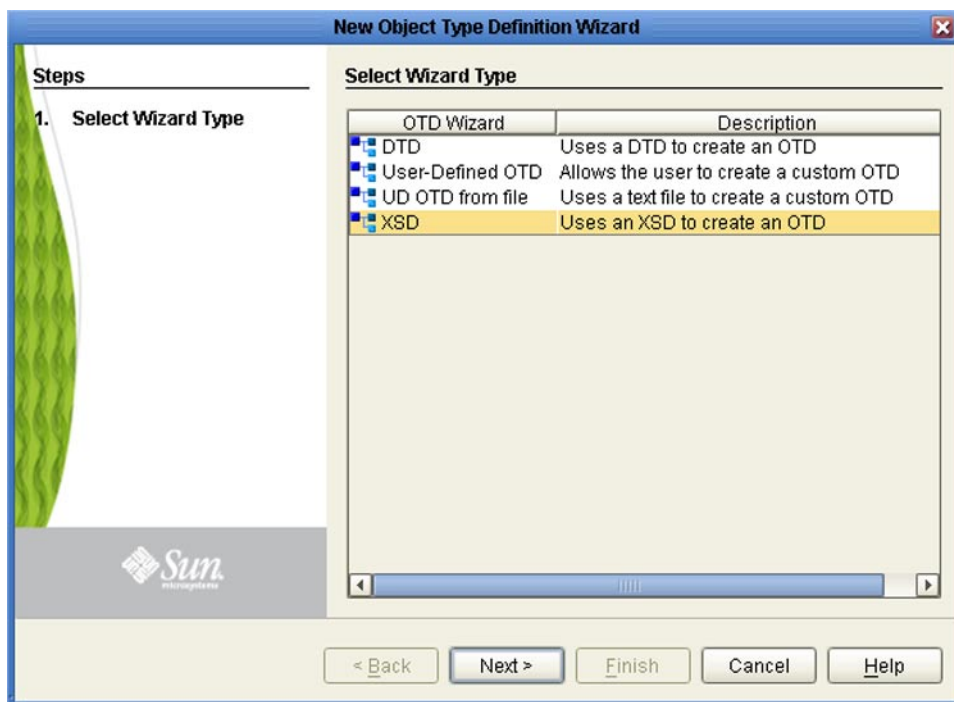
**Note:** The `reset()` method resets the XSD OTD to its initial conditions, including any default values. It is highly recommended that this method be used in your Collaborations whenever marshaling in a loop, to conserve system resources.

**Note:** If a simple type nillable element is nil in the input, it is treated as if it were absent.

To create a new OTD from an XSD file

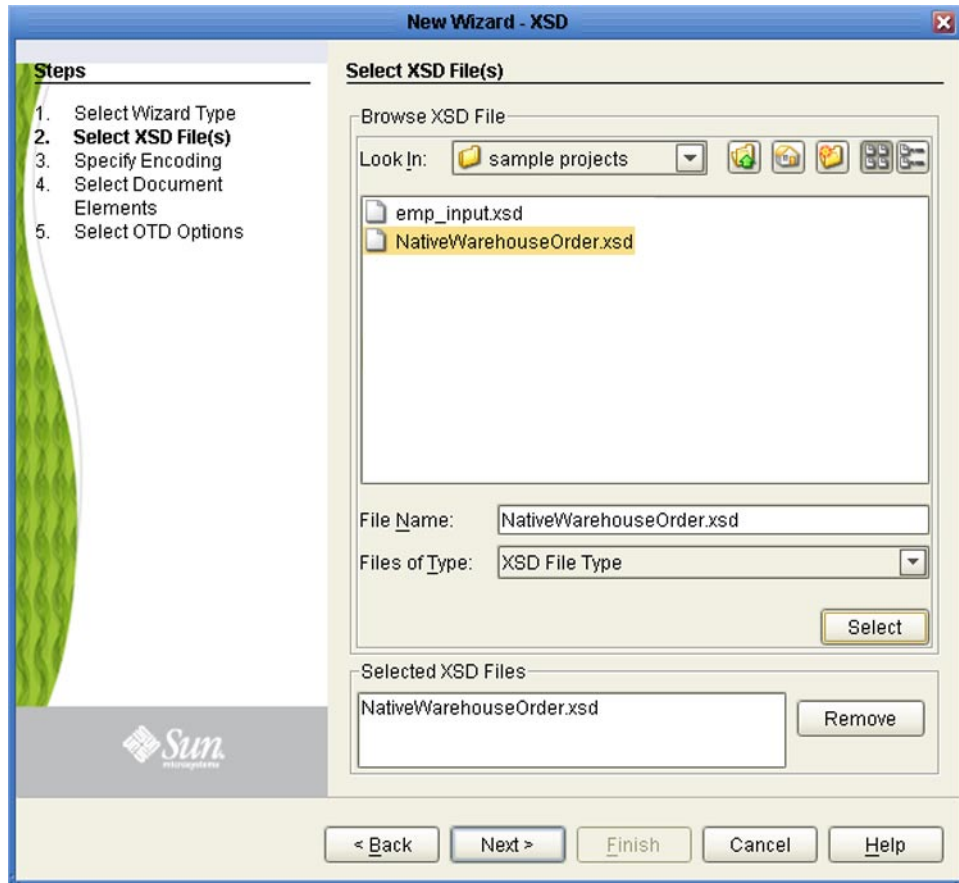
- 1 In the *Select Wizard Type* dialog, select **XSD** from the *OTD Wizard* list (see Figure 140) to create an OTD from an XSD file.

**Figure 140** OTD Wizard Selection: XSD Wizard



- 2 Click **Next** to display the Select XSD File(s) dialog, shown in Figure 141.

**Figure 141** XSD Wizard: Select XSD File(s)



- 3 In the *Look In* drop-down list, navigate to the XSD file or files that you want to use to create the OTD. Click **Select** to add the files to the *List of Selected XSDs*.

**Note:** Make sure the file name does not contain any blank spaces.

- ♦ If the XSD file does not contain all information required for building an OTD (such as element definitions) a warning box such as that shown in Figure 142 will be displayed.

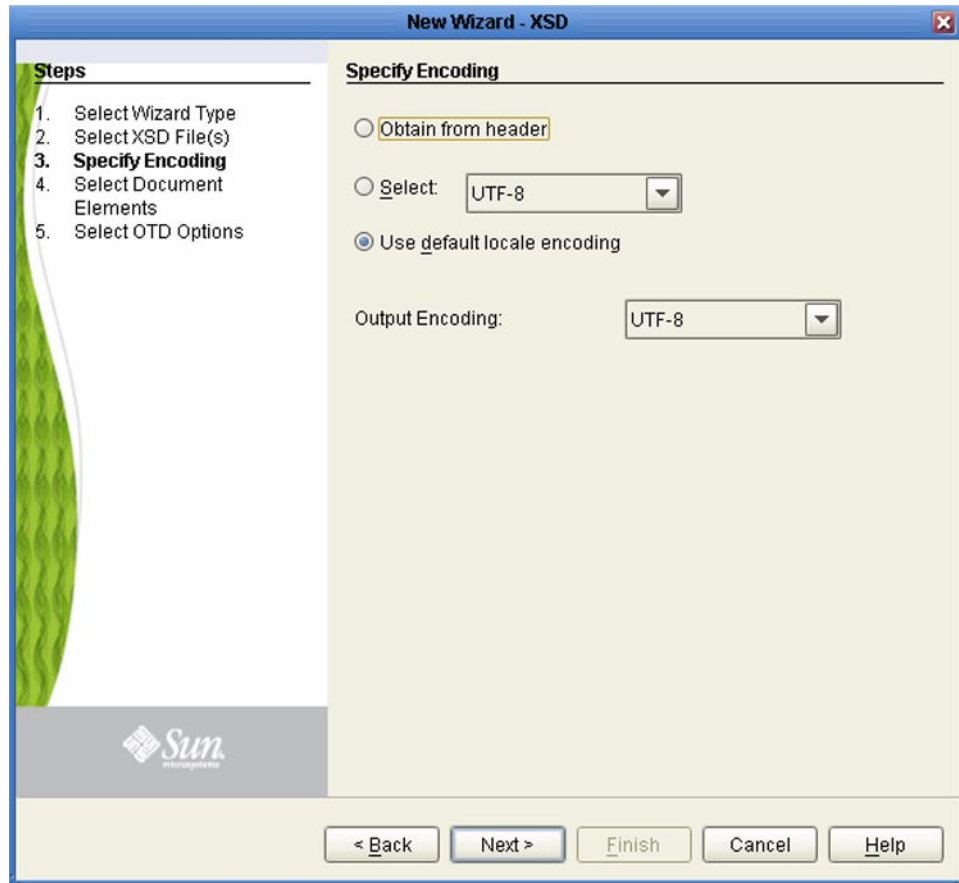
**Figure 142** Cannot Create OTD Warning Box



- 4 If you are using the extended language options (see [Setting Up Options](#) on page 53) click **Next** to display the *Specify Encoding* dialog, shown in Figure 143. If you are *not* using the extended language options, this Wizard dialog will not appear — proceed to step 7.

**Note:** This OTD encoding is currently supported for Java Collaboration Definitions only.

**Figure 143** Select Encoding Dialog



- 5 Click the appropriate option button to specify the data encoding for the input XML file:
  - ♦ Select **Obtain form header** (the default setting) to use the XML header encoding. If the header encoding does not exist, the default locale encoding is used instead.
  - ♦ Click **Select** to explicitly specify the encoding from a drop-down list, the contents of which depends upon the locale (see [Specifying Data Conversion and Encoding](#) on page 151).
  - ♦ Select **Use default locale encoding** to use the default encoding for the locale.

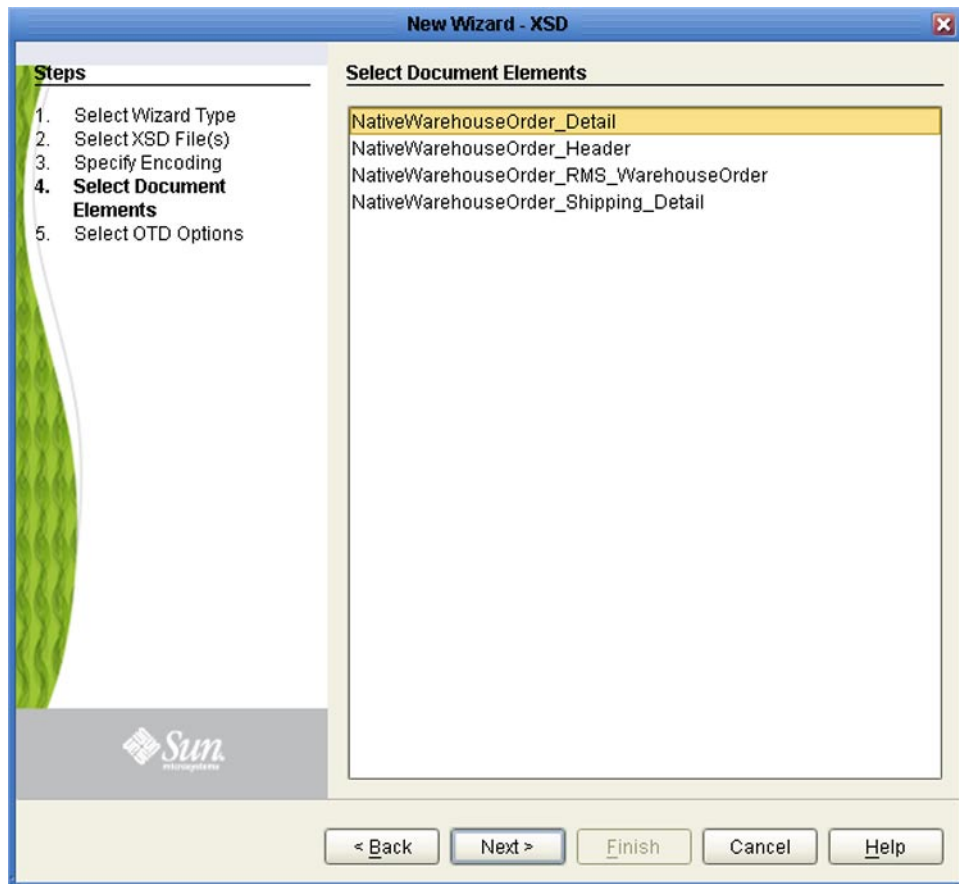
**Note:** When opening a data file, if the file is a .xml file with an encoding in the header, the input view encoding will default to the encoding in the .xml file.

- 6 Select **Output Encoding** to specify the header encoding of the output XML file from a drop-down list, the contents of which depends upon the locale (see [Specifying Data Conversion and Encoding](#) on page 151).

**Note:** For the Output Encoding property to apply to an imported project from a previous eGate Integrator release, you must recreate the OTD and specify the new encoding option.

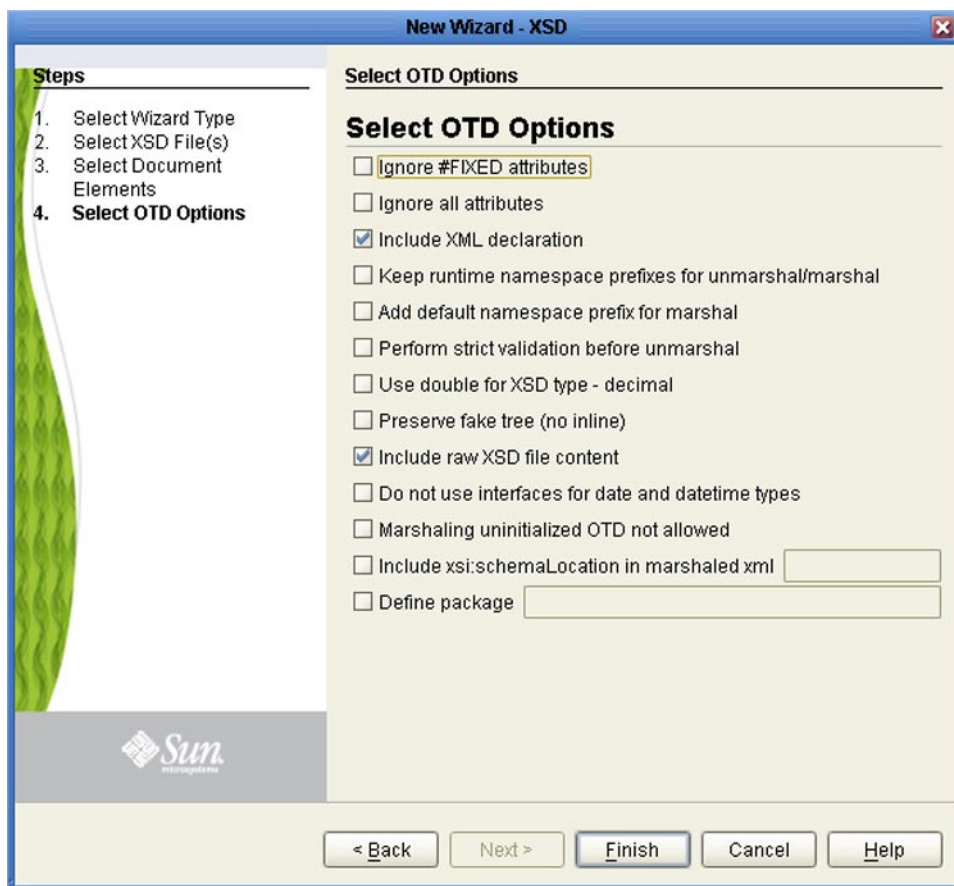
- 7 Click **Next** to display the *Select Document Elements* dialog, shown in Figure 144.

**Figure 144** Select Document Elements Dialog



- 8 Select the elements of the document that you want to include in the OTD.
- 9 Click **Next** to display the *Select OTD Options* dialog, shown in Figure 145.

**Figure 145** Select OTD Options Dialog



- 10 Select the check boxes next to the OTD options you want to enable (see Table 30).
- 11 Click **Finish** to add the OTD to your Project. The OTD Editor will open, displaying the new OTD (see [Using the OTD Editor](#) on page 159).

**Table 30** XSD OTD Options

Option	Description
Ignore #FIXED attributes	Controls whether or not attributes defined as FIXED are ignored during the unmarshal and marshal processes. <ul style="list-style-type: none"> <li>▪ If this option is <i>not</i> selected, the attribute is recognized and saved into the OTD's runtime structure during the unmarshal process, and also appears in the output during the marshal process.</li> <li>▪ If this option is selected, the attribute is ignored and neither of the above occurs.</li> </ul>
Ignore all attributes	Controls whether or not all attributes should be ignored in the unmarshal and marshal processes. If both this option and the <i>Keep runtime namespace prefixes</i> option (below) are selected, only namespace attributes will be handled during the unmarshal process and consequently presented in the output during the marshal process. (The <i>namespace</i> attribute has the form <b>xmlns:XX</b> .)
Include XML declaration	Controls whether or not the XML declaration <b>&lt;?xml version="1.0" encoding="....." ?&gt;</b> appears in the output during the marshal process. Option is checked by default.
Keep runtime namespace prefixes for unmarshal/marshal	Controls whether or not the namespace prefixes used during the marshal process are identical to those used in the unmarshal process. <ul style="list-style-type: none"> <li>▪ If this option is selected, all namespace attributes will be preserved once they appear in the XML instance document, and the namespace prefixes used in the marshal process will be exactly as they were presented in the XML document during the unmarshal process.</li> <li>▪ If this option is <i>not</i> selected, then the namespace prefixes used in the marshal process might be different than the ones presented in the XML document during the unmarshal process (for example, the namespace prefixes that are presented in the XSD file might be used).</li> </ul> <p><b>Note:</b> A consequence of selecting this option is that if there is no unmarshal process performed before the marshal process, then there will be no namespace attributes presented in the output (see the comment for the option below).</p>
Add default namespace prefix for marshal	Controls whether or not the prefix of the default target namespace of an element is applied to the element during the marshal process. <ul style="list-style-type: none"> <li>▪ If both this flag and the <i>Keep runtime namespace prefixes</i> option (above) are selected, then the default target namespace of an element will be applied to the element during the marshal process, <i>if it is a root element</i>.</li> <li>▪ If the <i>Keep runtime namespace prefixes</i> option is <i>not</i> selected, then the elements are qualified based on the XSD definition and this flag has no effect.</li> </ul>

Option	Description
Perform strict validation before unmarshal	Enables strict validation of the following (for elements only): <ul style="list-style-type: none"> <li>Union type</li> <li>List type</li> <li>Restriction</li> <li>Fundamental facets (as implied by built-in datatypes)</li> <li>Constraining facets</li> </ul> See Table 31 for additional information.
Use double for XSD type - decimal	Specifies the use of Java <b>double</b> type for the decimal type in the OTD. If not checked (default) Java <b>BigDecimal</b> type is used, which can handle big decimals with high precision.
Preserve fake tree (no inline)	Preserves the “fake tree” generated by the OTD Builder so that the XSD type information is retained and the element will be assignable in BPEL. This option is rarely used, since it degrades performance.
Include raw XSD file content	Saves the raw root-level XSD file content in the OTD to enable retrieval by other modules, if required (not retrievable by means of the GUI — only through programming). Checked by default. <b>Note:</b> All imported, redefined, and included XSD files are discarded.
Do not use interfaces for date and datetime types	This option is provided for backwards compatibility with XSD OTDs created previous releases of eGate Integrator.
Marshaling uninitialized OTD not allowed	If this option is checked: <ul style="list-style-type: none"> <li>No default values are set to any fields.</li> <li>Marshaling will throw an exception if any mandatory fields are not initialized with some value.</li> </ul>
Include xml:schemaLocation in marshaled xml	If this option is checked, and a schema location is specified in the text box, the schemaLocation attribute will be included in the marshaled output.
Define package	This option allows you to specify a package name, if needed. (This is rarely necessary, and should be used only by advanced users).

**Table 31** Strict Validation - XSD Facets

Category	Parameter	Comments
Derived Datatypes	byte	-
	ENTITY	Validates only that it is an NCName locally.
	ENTITIES	Validates only that they are NCNames locally.
	ID	Validates only that it is an NCName locally.
	IDREF	Validates only that it is an NCName locally.
	IDREFS	Validates only that they are NCNames locally.
	int	-
	integer	-
	language	-
	long	-
	Name	-
	NCName	-
	negativeInteger	-
	NMTOKEN	-
	NMTOKENS	-
	nonNegativeInteger	-
	nonPositiveInteger	-
	normalizedString	-
	positiveInteger	-
	short	-
	token	-
	unsignedByte	-
	unsignedInt	-
	unsignedLong	-
	unsignedShort	-



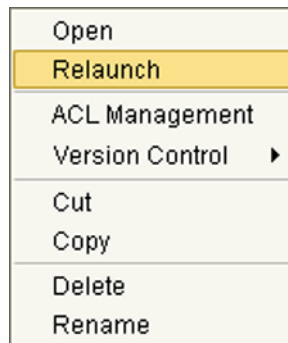
Category	Parameter	Comments
Primitive Datatypes	anyURI	-
	base64Binary	-
	boolean	-
	date	-
	dateTime	-
	decimal	-
	double	-
	duration	-
	float	-
	gDay	-
	gMonth	-
	gMonthDay	-
	gYear	-
	gYearMonth	-
	hexBinary	-
	NOTATION	-
	QName	-
	string	-
	time	-
Constraining Facets	enumeration	-
	fractionDigits	-
	length	Validates both List type and string-derived types.
	maxExclusive	Validates both numeric and chronological types.
	maxInclusive	Validates both numeric and chronological types.
	maxLength	Validates both List type and string-derived types.
	minExclusive	Validates both numeric and chronological types.
	minInclusive	Validates both numeric and chronological types.
	minLength	Validates both List type and string-derived types.
	pattern	-
	totalDigits	-
	whiteSpace	-

### To redefine an existing XSD OTD

You can redefine a newly-created XSD OTD by selecting the **Relaunch** option from its context menu in Project Explorer (see Figure 146). This relaunches the XSD Wizard, and allows you to reselect files and options while preserving the original OID.

**Note:** *Imported OTDs cannot be modified in this manner.*

**Figure 146** OTD Context Menu



You have the choice of two options for the way the Relaunch command works. The default functionality is the same as in the previous eGate release (5.0.x). When using this option, you should leave the existing node structure intact and append any newly-created nodes to the end of the existing node list, or create a new sibling node list.

If your XSD contains *choice*, *sequence*, *any*, or *all* node types, however, you should use the alternate functionality, which you must enable by modifying the following file:

```
<Sun_JavaCAPS_install_dir>\edesigner\bin\runed.bat
```

Simply locate the parameter

```
-J-DXsdOtdRelaunch.enabled
```

(it is near the end of the file), and set it to **true** (it is **false** by default). Using this functionality, you can delete, insert, or move an OTD node anywhere in the OTD structure, as long as you do not break the parent-child relationship of existing OTD nodes.

## 7.4 Creating User-Defined OTDs

When an externally-defined OTD is not a viable option, you can create your own OTD as described in this section.

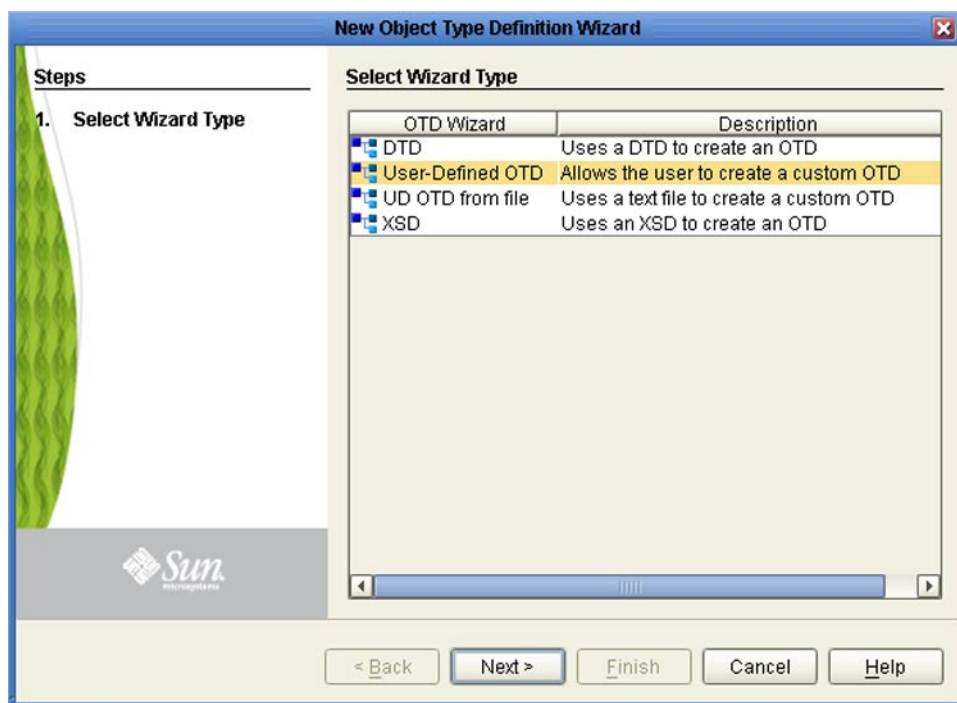
**Note:** The `reset()` method resets the User-Defined OTD to its initial conditions, including any default values. It is highly recommended that this method be used in your Collaborations whenever marshaling in a loop, to conserve system resources.

### 7.4.1 Using the User-Defined OTD Wizard

To create a User-Defined OTD

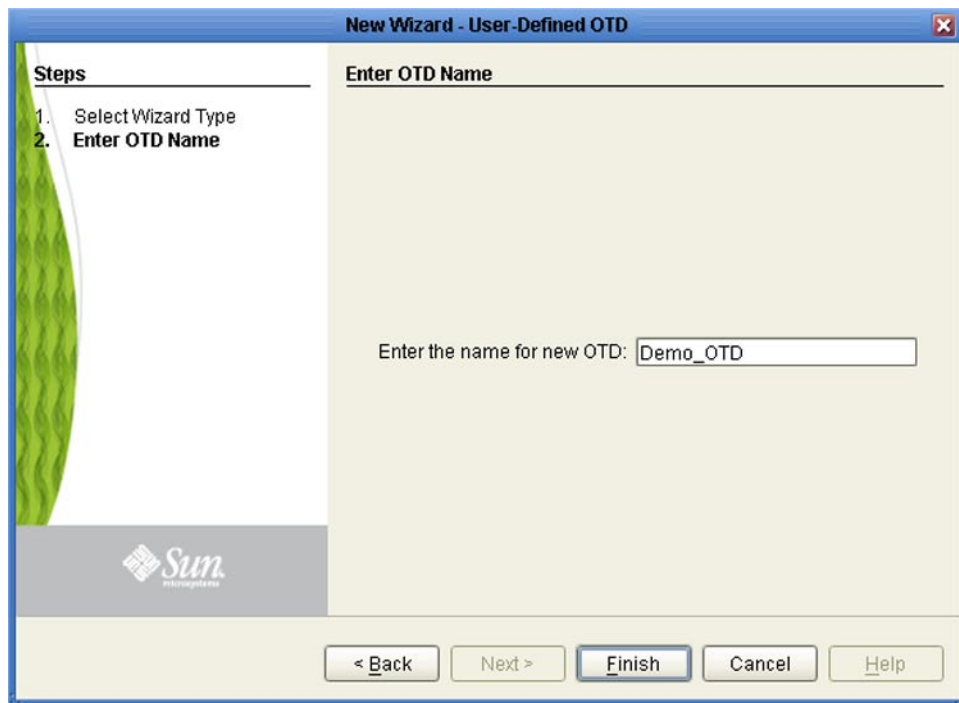
- 1 In the *Select Wizard Type* dialog, select **User-Defined OTD** from the *OTD Wizard* list (see Figure 147) to create an OTD without using a source file.

**Figure 147** OTD Wizard Selection: User-Defined OTD



- 2 Click **Next** to display the *Enter OTD Name* dialog, shown in Figure 148.

**Figure 148** Enter OTD Name



- 3 Enter a name for the OTD into the text box provided, then click **Finish** to add the OTD to your Project.

**Important:** *User-Defined OTD names must be less than 200 characters in length to avoid throwing an exception when building the application (EAR) file.*

- 4 The OTD Editor will open, displaying the new OTD (see [Using the OTD Editor](#) on page 159). You now must specify the OTD in detail, as described in the following sections.

## 7.4.2 Using the UD OTD from File Wizard

If you would like to create an OTD based on a simple but large flat data structure such as a spreadsheet, you can avoid entering all the field names manually by using the *UD OTD from File* wizard. You proceed by simply selecting a list of fields from the spreadsheet (or other document), copying the list into a text file, renaming the file with a **.ffd** extension, and then running the wizard as described below. The required format of the text file is a column of entries having the form:

```
[ <option> ] <name> [ <space><length> ]
```

where the **<space><length>** part is optional. If a value for **<length>** is entered, the nodeType is set to **fixed**. If no value is given, nodeType is set to **delim**. The encoding should be UTF-8.

**Table 32** Text File Variables

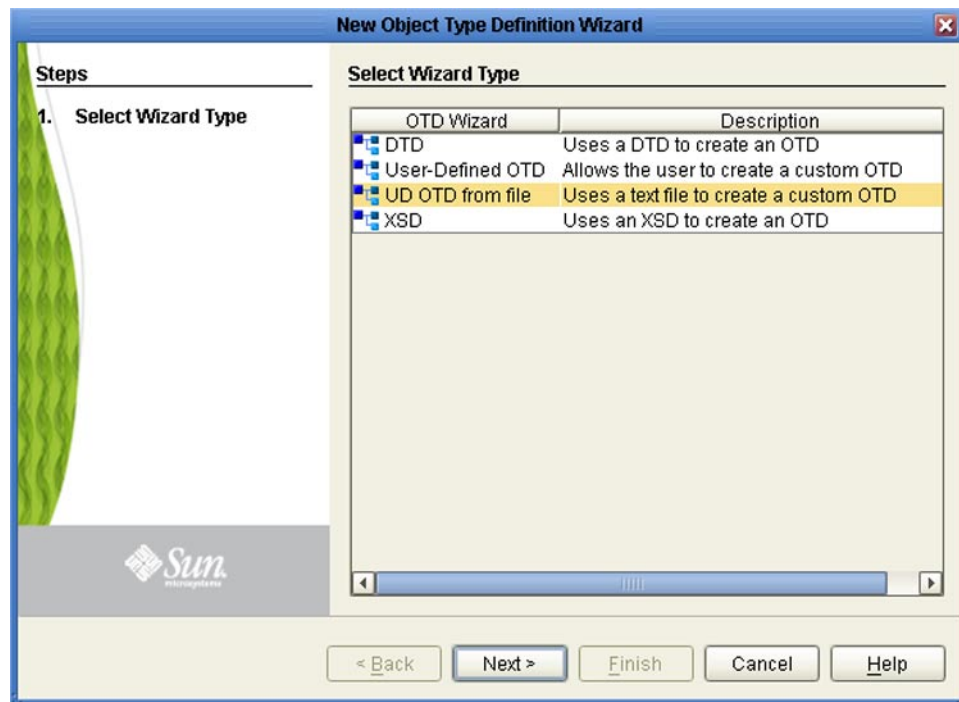
Variable	Value	Meaning
<option>	*	Repeating and optional.
	+	Repeating and not optional.
	?	Optional and not repeating.
<name>	<text>	Field name.
<space>	<none>	A blank space.
<length>	<number>	Fixed-field length.

The resulting OTD has a root with the list of fields as children. You can use the OTD Editor to modify the structure, changing the root name and adding delimiters, for example.

To create a User-Defined OTD from a flat file

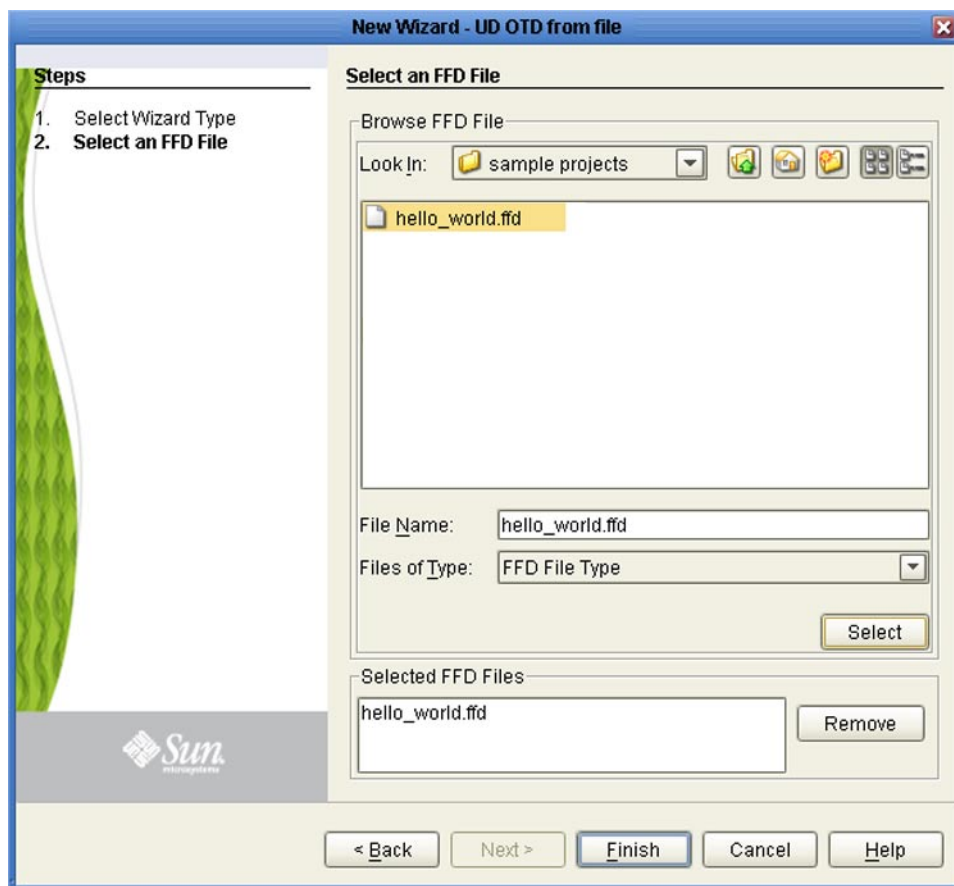
- 1 In the *Select Wizard Type* dialog, select **UD OTD from file** from the *OTD Wizard* list (see Figure 149) to create an OTD using a source file.

**Figure 149** OTD Wizard Selection: UD OTD from File



- 2 Click **Next** to display the *Select a File* dialog, shown in Figure 148.

**Figure 150** Select a File Dialog

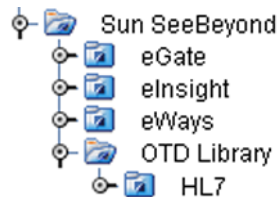


- 3 Browse for the file you want to use as a source for the OTD and click **Select**. You can select multiple files, which are listed in the *Selected Files* window.
- 4 Click **Finish** to build the OTD(s) and add them to your Project.
- 5 The OTD Editor will open, displaying the new OTD(s) (see [Using the OTD Editor](#) on page 159). You now must specify each OTD in detail, as described in the following sections.

## 7.5 Using Predefined OTDs

As the name suggests, predefined OTDs are pre-built, and instances of them are simply added to the Collaboration Definitions which use them. When present, these OTDs reside under the **Sun SeeBeyond** folder in Project Explorer (see Figure 151).

**Figure 151** Project Explorer - Sun SeeBeyond Folder



Examples of predefined OTDs are listed below:

- The *JMS OTD*, which is installed along with Enterprise Designer, resides in the **eGate** folder (see [Using JMS OTDs](#) on page 197).
- The *Scheduler OTD*, which is installed along with Enterprise Designer, resides in the **eGate** folder (see [Using Scheduler OTDs](#) on page 200).
- Other OTD libraries are separate products, and must be installed using the Java CAPS Installer. Following installation, they also reside in the **OTD Library** folder. The *HL7 OTD Library* is shown as an example in Figure 151. These libraries are described in their own user guides.

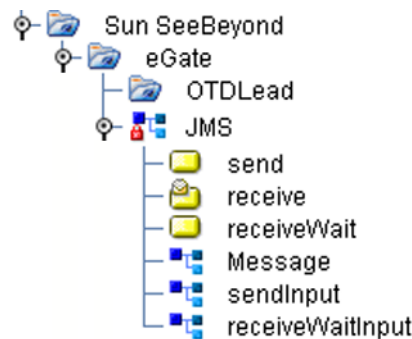


### 7.5.1 Using JMS OTDs

The Java Message Service (JMS) OTD is a special type of OTD that allows Collaborations to read from and write to topics or queues. It indicates to the Collaboration which topic or queue it expects to receive messages from or send messages to, and allows you to build the JMS business rules.

The JMS OTD is included with eGate Integrator, and is installed automatically. The template resides in the **Sun SeeBeyond\eGate** folder in Project Explorer (see Figure 152). This is the location to which you browse when you add a JMS OTD to a Collaboration Definition (Java) using the Collaboration Definition Wizard.

**Figure 152** Project Explorer - JMS OTD



Expanding the JMS OTD in the Collaboration Definition Editor exposes the property nodes shown in Figure 153.

**Figure 153** JMS OTD Property Nodes



You can set specific message properties in the JMS OTD using these property nodes. Message properties set in a Collaboration Definition override the corresponding JMS client message priorities, but only for the Collaboration that uses that definition. These properties are summarized in Table 33, and are described in detail in the *Sun SeeBeyond eGate™ Integrator JMS Reference Guide*.

**Table 33** JMS OTD Properties

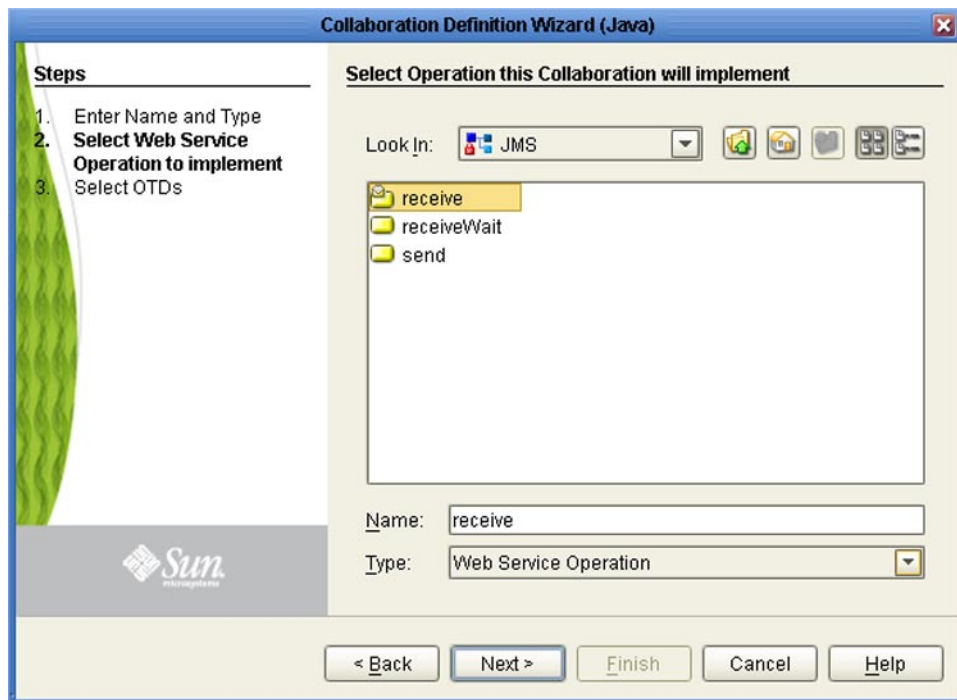
Name	Description
deliveryMode	Specifies whether the delivery mode of the message producer is persistent or non-persistent.
priority	Specifies the message priority. Values are from 0 to 9, where 9 is the highest.
timeToLive	Specifies the maximum amount of time before a live message expires.
destination	Specifies the destination topic or queue.
MessageServerURL	URL of the message server handling the messages.

You import a JMS OTD into a new Collaboration Definition (Java) when you use the Collaboration Definition Wizard, as described below.

**To import a JMS OTD into a Collaboration Definition (Java)**

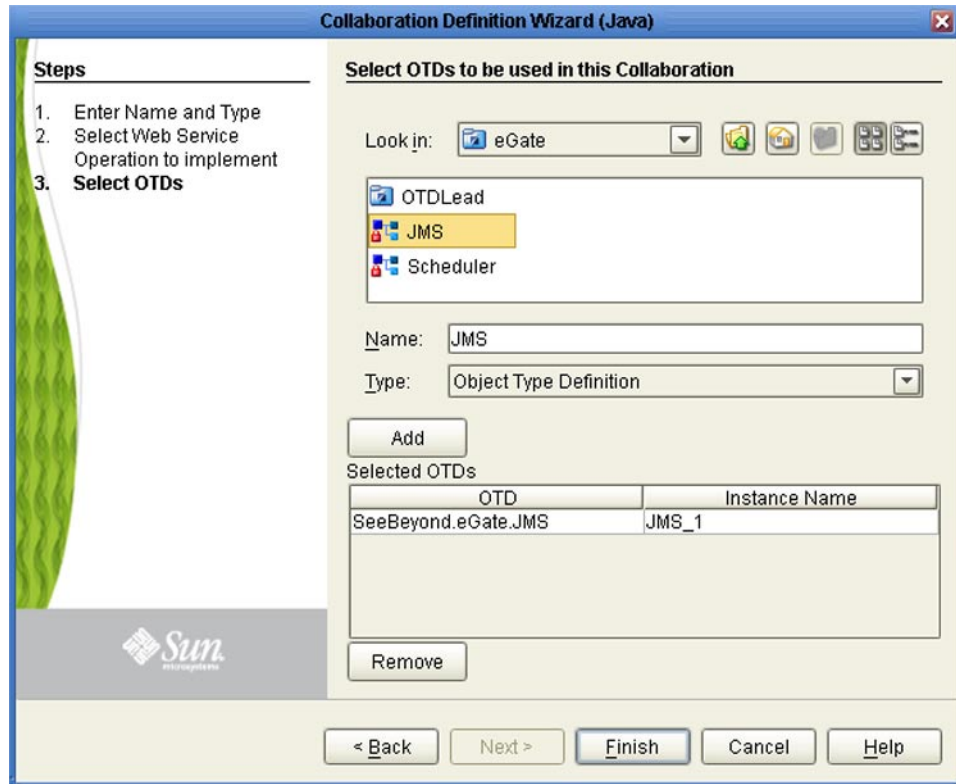
- 1 In Project Explorer, right-click the selected Project to display its context menu.
- 2 Click **New > Collaboration Definition (Java)** to display the Collaboration Definition Wizard.
- 3 Enter the name for the Collaboration Definition and select **Existing Web Service Type**.
- 4 Click **Next** to display the next wizard dialog (see Figure 154).

**Figure 154** Existing Web Service: Select Operation



- 5 Browse to open the **Sun SeeBeyond\eGate\JMS** folder and select the appropriate method — for example, **receive**.
- 6 Click **Next** to display the next wizard dialog (see Figure 155).

**Figure 155** Existing Web Service: Select OTDs



- 7 Browse to open the **Sun SeeBeyond\eGate** folder, select the **JMS** OTD, and add it to the list.
- 8 Add any other OTDs necessary for this Collaboration Definition.
- 9 Click **Finish** to open the Collaboration Editor.

The JMS OTD provides several JMS methods that enable you to build the JMS functionality for the Collaboration Definition. For information about the various JMS methods, refer to the *Sun SeeBeyond eGate™ Integrator JMS Reference Guide*.

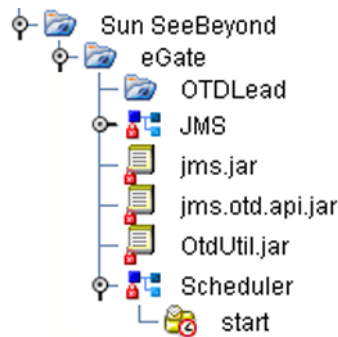
For more information on creating Java-based Collaboration Definitions, see [Creating a Java-based Collaboration Definition](#) on page 316.

## 7.5.2 Using Scheduler OTDs

The Scheduler OTD is a special type of OTD that you use in a Collaboration Definition if you are triggering the Collaboration with a Scheduler (see [Schedulers](#) on page 119).

The Scheduler OTD is included with eGate Integrator, and is installed automatically. The template resides in the **Sun Seebeyond\eGate** folder in Project Explorer (see Figure 152). This is the location you browse to when you add a Scheduler OTD to a Collaboration Definition (Java) using the Collaboration Definition Wizard.

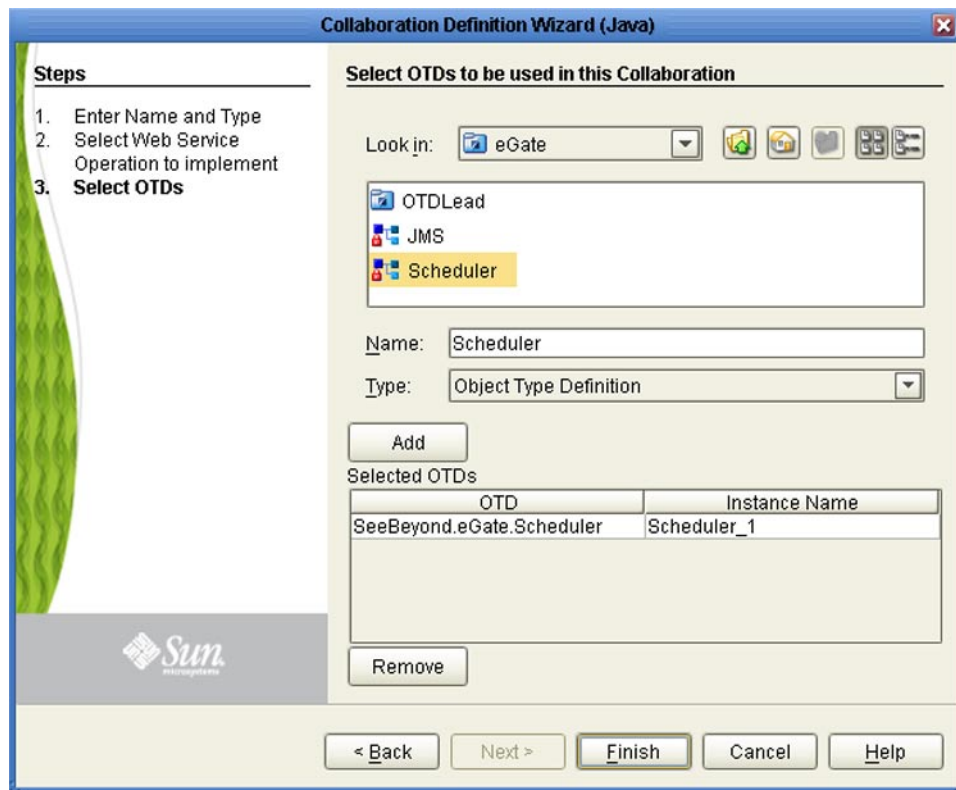
**Figure 156** Project Explorer - Scheduler OTD



### To import a Scheduler OTD into a Collaboration Definition (Java)

- 1 In Project Explorer, right-click the selected Project to display its context menu.
- 2 Click **New > Collaboration Definition (Java)** to display the Collaboration Definition Wizard.
- 3 Enter the name for the Collaboration Definition and select the appropriate web service type and web service.
- 4 Proceed to the **Select OTDs** wizard dialog (see Figure 155).

**Figure 157** Existing Web Service: Select OTDs



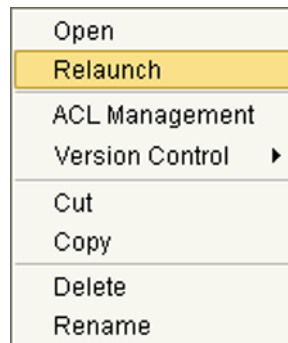
- 5 Browse to open the **Sun SeeBeyond\eGate** folder, select the **Scheduler** OTD, and add it to the list.
- 6 Add any other OTDs necessary for this Collaboration Definition.
- 7 Click **Finish** to open the Collaboration Editor.

For more information on creating Java-based Collaboration Definitions, see [Creating a Java-based Collaboration Definition](#) on page 316.

## 7.6 Using the OTD Context Menu

The context menu for an Object Type Definition is the standard menu shown in [Using Project Component Context Menus](#) on page 109, with the addition of the **Relaunch** option for DTD and XSD OTDs shown in Figure 158.

**Figure 158** OTD Context Menu



**Table 34** OTD Context Menu Options

Option	Function
Open	If shown, opens the selected Object Type Definition in the appropriate Enterprise Designer editor.
Relaunch	Available for newly-created DTD and XSD OTDs only, relaunches the OTD Wizard, and allows you to reselect files and options while preserving the original OID.
ACL Management	Presents the ACL Properties dialog, with which an Administrator can assign read/write privileges to users for the selected Object Type Definition. See the <i>Sun SeeBeyond eGate™ Integrator System Administration Guide</i> .
Version Control	Accesses the version control features for the selected Object Type Definition. See <a href="#">Accessing Component Version Control Features</a> on page 79 for additional information.
Cut	Copies the selected Object Type Definition and removes it from the current Project, after which you can paste it to another Project (once only). All changes must be committed before you can cut the Object Type Definition. Cut and paste is disabled for other users when you have the Object Type Definition checked out. See <a href="#">Moving and Replicating Repository Objects</a> on page 97 for additional information.

**Table 34** OTD Context Menu Options

Option	Function
Copy	Copies the selected Object Type Definition, after which you can paste it to other Projects (multiple times). All changes must be committed before you can copy the Object Type Definition. You can copy and paste a component even when another user has the Object Type Definition checked out. See <a href="#">Moving and Replicating Repository Objects</a> on page 97 for additional information.
Delete	<p>Deletes the selected Object Type Definition, subject to the following conditions:</p> <ul style="list-style-type: none"> <li>▪ You have <i>write</i> privileges for the Object Type Definition (see ACL Management, above).</li> <li>▪ You have the Object Type Definition checked out to your workspace.</li> <li>▪ The Object Type Definition is not checked out by anyone other than yourself.</li> </ul> <p>If these conditions are met, a dialog is presented in which you confirm that you want to delete the selected Object Type Definition. Clicking <b>Yes</b> then deletes the Object Type Definition.</p>
Rename	Activates the field, allowing you to rename the selected component.

**Important:** All Project component names should contain only alphanumeric characters (letters and numbers), dashes, and underscores.

**Important:** If you delete an OTD in the Project Explorer, any Collaboration Definitions that have been built using that OTD will be affected. It is recommended that you run the Impact Analyzer before attempting to delete any OTDs (see [Analyzing Component Dependencies](#) on page 99).

# Object Type Definitions III

This chapter describes procedures for creating and modifying user-defined OTDs.

## What's in This Chapter

- [Overview](#) on page 204
- [Creating and Managing OTD Nodes](#) on page 205
- [Creating and Using Templates](#) on page 209
- [Editing OTD Properties](#) on page 214
- [Specifying the Node Type](#) on page 220
- [Specifying Read/Write Access](#) on page 222
- [Matching Data Patterns](#) on page 223
- [Specifying Delimiters](#) on page 225
- [Creating a Delimiter List](#) on page 236

---

## 8.1 Overview

Many requirements for OTDs are satisfied by externally-defined types based on industry standard or proprietary metadata formats, at times available as templates in a library. When an externally-defined OTD is not a viable option, you can define your own OTD using the User-Defined OTD Wizard, as described in [Creating User-Defined OTDs](#) on page 191, and the OTD Editor, which is described in [Using the OTD Editor](#) on page 159.

This chapter explores the many details of OTDs and describes how to configure the various properties of User-Defined OTDs using the OTD Editor.

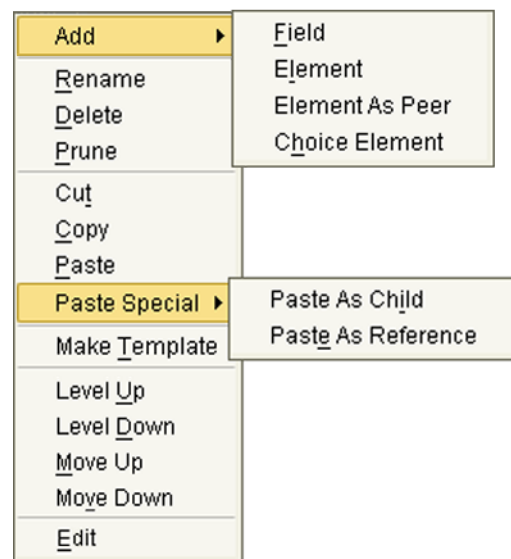


## 8.2 Creating and Managing OTD Nodes

Right-clicking a node in the OTD Editor displays the context menu for that node (see Figure 159). Once they are created, Elements and Fields can be renamed or deleted, and their properties can be configured as described in the following sections. Nodes that you have cut or copied from an OTD tree structure can be pasted to other nodes, either in the same OTD or another one. This is a good alternative to adding a “fresh” node to the OTD when you want the new node to inherit properties from an existing node (see [Cutting, Copying, and Pasting Nodes](#) on page 208).

**Note:** Options that would result in non-allowed situations are automatically disabled.

**Figure 159** OTD Node Context Menu



**Table 35** OTD Node Context Menu Options

Option		Action
Add	Field	Adds a field node to the OTD tree as a child of the selected root or element node.
	Element	Adds an element node to the OTD tree as a child the selected root or element node.
	Element As Peer	Adds an element node to the OTD tree as a sibling the selected element or field node.
	Choice Element	Adds a choice element node to the OTD tree as a child the selected root or element node.
Rename		Allows you to rename the selected node. <b>Note:</b> Renaming the root node renames the OTD.
Delete		When a node is deleted, both the node and its descendants are removed.

**Table 35** OTD Node Context Menu Options

Option		Action
Prune		When a node is pruned, only its descendants are removed, while the node itself is preserved. Selecting this option presents a dialog in which you must confirm your intentions before the operation can be completed.
Cut		Cutting a node removes it and its descendants from the tree, and temporarily places them in an internal buffer for subsequent pasting elsewhere. All recent changes must be committed before you can cut the node. See <a href="#">Cutting, Copying, and Pasting Nodes</a> on page 208 for additional information.
Copy		Copying a node temporarily places an instance of it and its descendants in an internal buffer for subsequent pasting elsewhere. All recent changes must be committed before you can copy the node. You can copy a node even when another user has the OTD checked out. See <a href="#">Cutting, Copying, and Pasting Nodes</a> on page 208 for additional information.
Paste		Pastes the cut or copied node (with its descendants) from the temporary buffer into the OTD tree below the selected node, as a sibling of the selected node. You cannot paste to a root node. See <a href="#">Cutting, Copying, and Pasting Nodes</a> on page 208 for additional information.
Paste Special	Paste As Child	Pastes the cut or copied node (with its descendants) from the temporary buffer into the OTD tree as a child of the selected node. <i>Paste As Child</i> is disabled for field nodes. See <a href="#">Cutting, Copying, and Pasting Nodes</a> on page 208 for additional information.
	Paste As Reference	Pastes an instance of the copied node (with its descendants) from the temporary buffer into the OTD tree as a reference to the copied node, which now serves as an embedded template. See <a href="#">Cutting, Copying, and Pasting Nodes</a> on page 208 for additional information.
Make Template		Copies the selected node and its descendants and pastes them into the <i>Reference</i> panel as an internal template. It appears as a new OTD, with the selected node as the root. See <a href="#">Creating and Using Templates</a> on page 209 for additional information.
Level Up		Moves the selected node one level to the right in the hierarchy of the tree structure, thereby becoming a child of the node above its original position (see <a href="#">Moving Nodes</a> on page 207).
Level Down		Moves the selected node one level to the left in the hierarchy of the tree structure, thereby becoming a parent of the node below its original position (see <a href="#">Moving Nodes</a> on page 207).
Move Up		Moves the selected node up one line in the tree structure, at the same hierarchical level (see <a href="#">Moving Nodes</a> on page 207).
Move Down		Moves the selected node down one line in the tree structure, at the same hierarchical level (see <a href="#">Moving Nodes</a> on page 207).

**Table 35** OTD Node Context Menu Options

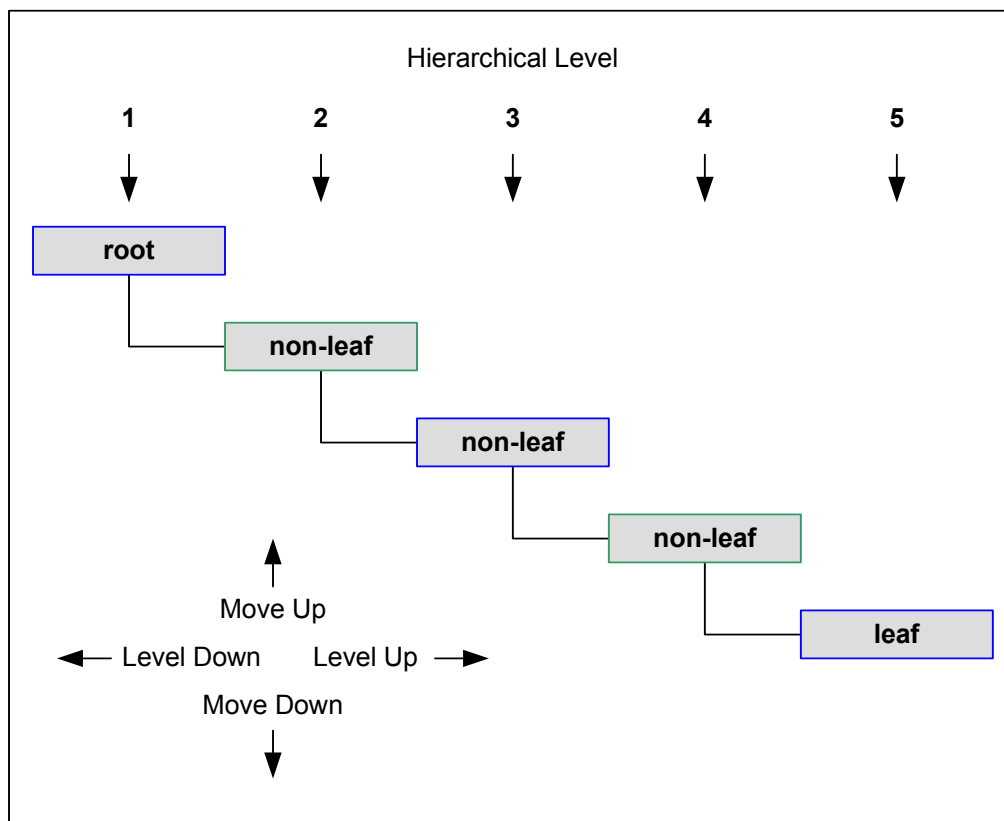
Option	Action
Edit	For external templates only, enables the properties for selected node to be edited (see <a href="#">Cutting, Copying, and Pasting Nodes</a> on page 208).

### 8.2.1 Moving Nodes

Nodes can be moved within the OTD tree structure using the **Level Up/Down** and **Move Up/Down** options in the node context menu as described in Table 35. To supplement these descriptions, the respective actions of these options is illustrated in Figure 160.

**Note:** Remember that no other node can exist at the same hierarchical level as the root node, and that leaf nodes cannot have children.

**Figure 160** Node-moving Options



**Important:** If you move an OTD node, you must reset the **nodeType** for that node. See [Editing OTD Properties](#) on page 214.

You can also move individual nodes by dragging and dropping them onto a new parent node.

## 8.2.2 Cutting, Copying, and Pasting Nodes

You can cut or copy a node and paste it to a new location, either within an OTD or between different OTDs. If the node is an Element or a template reference, the pasted node retains the descendant node structure, properties and configuration of the source node.

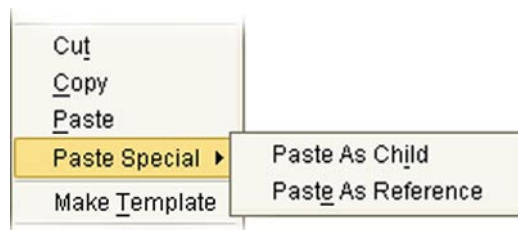
**Note:** *If nodes in the template are delimited but not specified individually, the delimiter properties in each pasted instance change to reflect the new parentage settings.*

All changes made during the current editing session must be committed before you can cut or copy a node. If the operation is a cut, the source node is removed; if the operation is a copy, the source node is unchanged. You can perform the same operation on multiple nodes by holding the **Ctrl** key while you select the nodes.

Paste operations attempt to retain the same node name as the source. If the name would cause a conflict, however, the name has a number appended to it. The numbering begins with 1 and is incremented, if necessary, until the conflict is resolved.

You have several options for pasting nodes and their descendant structures. These options determine the placement of the pasted nodes, and the presence or absence of dependencies between instances of nodes that have been copied and pasted (see Figure 161).

**Figure 161** Node Context Menu - Paste Options



### Paste

Pastes the cut node or an instance of the copied node (including descendants) into the OTD tree as a sibling of the selected node, following the selected node. You cannot paste to a root node, since root nodes cannot have siblings. Changes made to the pasted node are independent of any other instances of the node.

### Paste As Child

Pastes the cut node or an instance of the copied node (including descendants) into the OTD tree as a child of the selected node. This option is disabled for field nodes, since field nodes cannot have children. Changes made to the pasted node are independent of any other instances of the node.

### Paste As Reference

Pastes an instance of the copied node (including descendants) into the OTD tree as a sibling of the selected node, following the selected node. The pasted instance has a dependency relationship with the copied (source) node, which now serves as an embedded template. Changes made to one instance appear in all instances of the node.

## 8.3 Creating and Using Templates

OTD templates allow you to reuse existing OTD structures within other OTDs. This can save time and effort and improve maintainability if you use a common metadata structure in multiple OTD files, or multiple times within a single OTD. Since templates are used by reference, they allow you to update all instances of the template by simply updating the template itself.

The different icons used to represent OTD nodes based on internal and external templates are shown in [OTD Node Representations](#) on page 212.

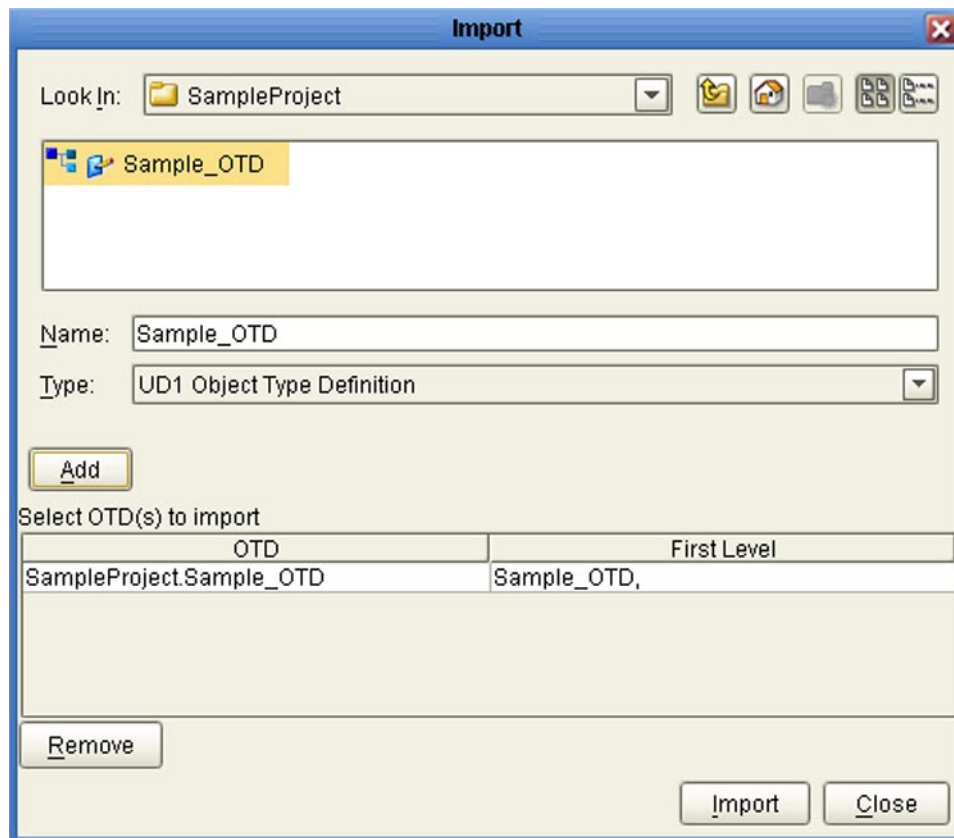
### 8.3.1 External Templates

Any OTD file in your Repository that is outside the currently open OTD can become an external template, including files from OTD libraries.

To import an OTD as an external template

- 1 Click the **Import OTD to External Template** icon (see [OTD Editor Toolbar Icons](#) on page 160) to display the *Import* dialog, shown in Figure 162.

**Figure 162** Import Dialog



- 2 Browse for the OTD you want to reference in the currently active OTD (it must reside in your Repository).
- 3 Click **Add**.
- 4 Browse for and add any other OTDs you want to reference.
- 5 Click **Import**.
- 6 Click **Close** after all selected OTDs have imported successfully.
- 7 The OTDs you have imported now appear on the *External* page of the editor's *Reference* panel.

#### To reference an external template in an OTD

- 1 Expand the open OTD as needed to expose the node you want to be the parent of the new template instance.
- 2 Left-click the template on the *External* page of the editor's *Reference* panel, drag it onto the *Object Type Definition* panel, and drop it onto the intended parent node.
- 3 A new instance of the template appears under the node selected as the parent.
- 4 Click **Save**.

**Note:** *Individually-specified delimiter settings and other properties remain the same as in the template. If nodes in the template are delimited but not specified individually, the delimiter properties in each instance change to reflect the new parentage settings.*

#### To modify an external template

- 1 Select the template you want to modify on the *External* page of the editor's *Reference* panel. The template will immediately appear in the *Object Type Definition* panel.
- 2 Right-click to display the template's context menu, and select **Edit**.
- 3 The original OTD (in the original Project) is opened for editing.
- 4 Edit the OTD and click **Save**. All instances of the template in the Repository are now updated according to your edits.

**Note:** *To see the changes in the editor, you must close and reopen the OTD.*

## 8.3.2 Internal Templates

You can select a portion of an OTD — a node and its descendants — and save it as an internal template. You can then reuse it multiple times as you continue to build the OTD. Unlike external templates, internal templates are available only within the OTD in which you created them. The advantage over copying and pasting nodes is that you can update all instances of the template by simply updating one instance.

### To create an internal template

- 1 In the open OTD, select the node you want to use as a template, and right-click to display its context menu.
- 2 Select the **Make Template** option from the menu.

A copy of the node and its descendants appear on the *Internal* page of the editor's *Reference* panel. The properties for the node you selected now have the same options as for a root node (the existing *optional* and *repeat* properties are lost).

### To use an internal template in an OTD

- 1 Expand the open OTD as needed to expose the node you want to be the parent of the new template instance.
- 2 Left-click the template on the *Internal* page of the editor's *Reference* panel, drag it onto the *Object Type Definition* panel, and drop it onto the intended parent node.
- 3 A new instance of the template appears under the node selected as the parent. The previous *optional* and *repeat* properties are restored.
- 4 Click **Save**.

**Note:** *Individually-specified delimiter settings and other properties remain the same as in the template. If nodes in the template are delimited but not specified individually, the delimiter properties in each instance change to reflect the new parentage settings.*

**Important:** *Any editing you perform on the new instance will affect all other instances of the template.*

### To modify an internal template

- 1 Select either the template you want to modify on the *Internal* page of the editor's *Reference* panel, or any instance of the template in the open OTD.
- 2 Edit the template as desired.
- 3 Click **Save**. All instances of the template in the Repository are now updated according to your edits.

### 8.3.3 OTD Node Representations

In the OTD Editor, different icons are used to differentiate between nodes that belong to the OTD itself, and those belonging to either internal or external templates. This information is important when you are contemplating a change to a node, since changing a node belonging to a template changes all instances of it. See also [Creating and Using Templates](#) on page 209.

#### Element Nodes

##### Belonging to OTD

Element nodes that are added to an OTD in the OTD Editor are represented by the same icon (*Add Element*) that appears in the toolbar (see Figure 163).

**Figure 163** Element Node Created for OTD



##### Belonging to External Template

When you drag an external template from the *External Reference* panel of the OTD Editor to an open OTD, all Element nodes in the template are represented by the icon shown in Figure 164. This icon indicates that the node is read-only in the open OTD; to edit the node, you must do so in the external template itself (see [External Templates](#) on page 209).

**Figure 164** Element Node from External Template



##### Instance of an Internal Template

When you create an internal template from an Element node, all instances of it are represented by the icon shown in Figure 165. All Element nodes within an internal template, however, are represented by the icon shown above in Figure 163.

**Figure 165** Element Node from Internal Template





## Field Nodes

### Belonging to OTD

Field nodes that are added to an OTD in the OTD Editor are represented by the same icon (*Add Field*) that appears in the toolbar (see Figure 166).

**Figure 166** Field Node Created for OTD



### Belonging to External Template

When you drag an external template from the *External Reference* panel of the OTD Editor to an open OTD, all Field nodes in the template are represented by the icon shown in Figure 167. This icon indicates that the node is read-only in the open OTD; to edit the node, you must do so in the external template itself (see [External Templates](#) on page 209).

**Figure 167** Field Node from External Template



### Instance of an Internal Template

When you create an internal template from a Field node, all instances of it are represented by the icon shown in Figure 168. All Field nodes within an internal template, however, are represented by the icon shown above in Figure 166.

**Figure 168** Internal Template - Field Node



## 8.4 Editing OTD Properties

**Note:** Clicking a **Value** field selects it, after which you can edit the text or select an option from a drop-down menu (in the case of the **delim** property, you open the *Delimiter List Editor*).

### Root Node Properties

The set of properties associated with Root nodes is shown in Figure 169.

**Figure 169** Root Node Properties (delim nodeType)

Properties	
Name	Value
name	Demo_OTD
javaName	Demo_OTD
javaType	com.stc.Demo_OTD_...
comment	
delim	specified
nodeType	delim
showDelim	
antecoding	
decoding	
encoding	
order	seq
postcoding	
public	false
top	true

**Table 36** Root Node Properties

Name	Description
name	Node display name (see <a href="#">Understanding OTDs</a> on page 146).
javaName	Property accessor basename, automatically generated — <i>do not modify</i> . See <a href="#">Understanding OTDs</a> on page 146.
javaType	Java type; automatically assigned, not editable.
comment	Free-form text (no run-time effect).
delim	Delimiter specification (see <a href="#">Specifying Delimiters</a> on page 225). Default value is <b>not set</b> .
nodeType	Specifies the marshal/unmarshal format (see <a href="#">Specifying the Node Type</a> on page 220). The default value is <b>delim</b> .

Name	Description
showDelim	Displayed only if <b>nodeType</b> is <i>delim</i> . The default value is <b>-none-</b> ; once delimiters are specified, value field shows the delimiters. (Root nodes are not typically delimited, however.)
antecoding	Specifies the <i>input data</i> coding (see <a href="#">Specifying Data Conversion and Encoding</a> on page 151). If this property is not specified, the value specified for the <b>decoding</b> property will be used for the input data. This property is displayed only when the <i>top</i> property is set to <b>true</b> .
decoding	Specifies the <i>unmarshal</i> coding (see <a href="#">Specifying Data Conversion and Encoding</a> on page 151). (It is recommended to use UTF-8 for DBCS data, since the hex value of some ASCII delimiter may coincide with a hex value contained within a double-byte character.) This property is displayed only when the <i>top</i> property is set to <b>true</b> .
encoding	Specifies the <i>marshal</i> coding (see <a href="#">Specifying Data Conversion and Encoding</a> on page 151). This property is displayed only when the <i>top</i> property is set to <b>true</b> .
length	Displayed only if <b>nodeType</b> is <i>fixed</i> . Specifies the length of the field; the default value is <b>0</b> .
order	Specifies the ordering of the root node's children. <ul style="list-style-type: none"> <li>▪ <b>seq</b> specifies that the child nodes must appear in sequence.</li> <li>▪ <b>any</b> specifies that the child nodes can appear in any order.</li> <li>▪ <b>mix</b></li> </ul>
postcoding	Specifies the <i>output data</i> coding (see <a href="#">Specifying Data Conversion and Encoding</a> on page 151). If this property is not specified, the value specified for the <b>encoding</b> property will be used for the output data. This property is displayed only when the <i>top</i> property is set to <b>true</b> .
public	For future use, not currently active; <b>false</b> by default.
top	Specifies whether or not marshal/unmarshal is supported ( <b>true</b> or <b>false</b> ). The default value is <b>true</b> .

**Important:** User-Defined OTD names must be less than 200 characters in length to avoid throwing an exception when building the application (EAR) file.

## 8.4.1 Element Node Properties

The set of properties associated with Element and Choice Element nodes is shown in Figure 170.

**Figure 170** Element Node Properties (delim nodeType)

Properties	
Name	Value
name	element_1
javaName	Element1
javaType	com.stc.Demo_OTD_...
comment	
access	modify
optional	false
repeat	false
maxOccurs	-1
delim	not set
nodeType	delim
showDelim	^
order	seq

**Table 37** Element Node Properties

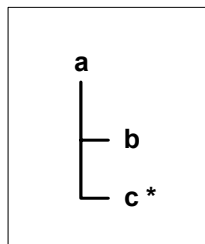
Name	Description
name	Element display name.
javaName	Property accessor basename, automatically generated — <i>do not modify</i> .
javaType	Java type; automatically assigned, not editable.
comment	Free-form text (no run-time effect).
access	Access specification (see <a href="#">Specifying Read/Write Access</a> on page 222).
optional	Specifies whether or not the node can be absent from an instance. Clicking the <i>Value</i> field toggles between <b>true</b> and <b>false</b> . Not applicable if the element is the child of a <i>choice element</i> node.
repeat	Specifies whether or not the node can appear multiple times. Clicking the <i>Value</i> field toggles between <b>true</b> and <b>false</b> . Not applicable if the element is the child of a <i>choice element</i> node.
maxOccurs	Specifies the maximum number of occurrences of the node if the node is repeating. Property has no effect if node is non-repeating, but may show error during validation if set to value >1.
delim	Delimiter specification (see <a href="#">Specifying Delimiters</a> on page 225).

Name	Description
nodeType	Governs the marshal/unmarshal format. The default value is <b>delim</b> , but is defaulted to parent nodeType if parent nodeType is <i>fixed</i> or <i>trans</i> .
showDelim	Displayed only if <b>nodeType</b> is <i>delim</i> . The default value is <b>-none-</b> ; once delimiters are specified, value field shows the delimiters. Does not apply to <i>choice element</i> nodes.
length	Displayed only if <b>nodeType</b> is <i>fixed</i> . Specifies the length of the field; the default value is <b>0</b> .
order	Specifies the ordering of the selected node's children during the unmarshaling process. <ul style="list-style-type: none"> <li>▪ <b>seq</b> specifies that the child nodes must appear in the sequence given in the metadata.</li> <li>▪ <b>any</b> specifies that the child nodes must remain grouped, but the groups can appear in any order.</li> <li>▪ <b>mix</b> specifies that the child nodes can appear in any order.</li> </ul> See the illustration below for additional information. <b>Note:</b> Does not apply to <i>choice element</i> nodes.

## Order Property

To illustrate how the **order** property works, consider the simple tree structure shown in Figure 171, where **a** is an element node, **b** is a non-repeating field node, and **c** is a repeating field node. The value set for the **order** property allows the field nodes to appear as shown in Table 38.

**Figure 171** Order Property Example



**Table 38** Order Property Example

Value	Allowed Node Order
seq	b, c1, c2
any	b, c1, c2, <b>or</b> c1, c2, b
mix	b, c1, c2, <b>or</b> c1, c2, b, <b>or</b> c1, b, c2

## 8.4.2 Field Node Properties

The set of properties associated with Field nodes is shown in Figure 172.

**Figure 172** Field Node Properties (delim nodeType)

Properties	
Name	Value
name	field_1
javaName	Field1
javaType	java.lang.String
comment	
access	modify
optional	false
repeat	false
maxOccurs	-1
delim	not set
initial	
match	
nodeType	delim
showDelim	^
align	blind

**Table 39** Field Node Properties

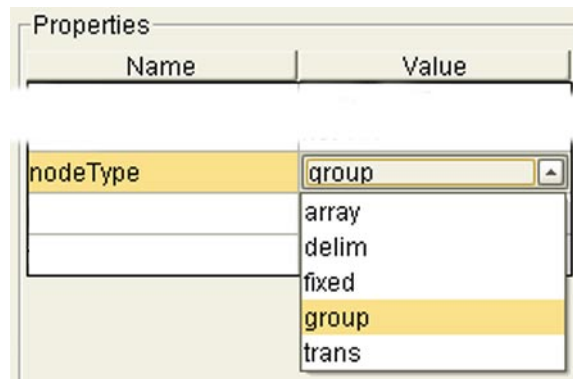
Name	Description
name	Field display name.
javaName	Property accessor basename, automatically generated — <i>do not modify</i> .
javaType	Java type; can be either <b>java.lang.String</b> or byte array ( <b>byte[]</b> ).
comment	Free-form text (no run-time effect).
access	Access specification (see <a href="#">Specifying Read/Write Access</a> on page 222).
optional	Specifies whether or not the field can be absent from an instance. Clicking the <i>Value</i> field toggles between <b>true</b> and <b>false</b> . Not applicable if the field is the child of a <i>choice element</i> node.
repeat	Specifies whether or not the node can appear multiple times. Clicking the <i>Value</i> field toggles between <b>true</b> and <b>false</b> . Not applicable if the field is the child of a <i>choice element</i> node.
maxOccurs	Specifies the maximum number of occurrences of the node if the node is repeating. Property has no effect if node is non-repeating, but may show error during validation if set to value >1.
delim	Delimiter specification (see <a href="#">Specifying Delimiters</a> on page 225).

Name	Description
initial	Initial field value, set when the parent node is created or reset. When provided, it is assigned to the node before the node is populated with any data.
match	If <b>nodeType</b> is <i>delimited</i> or <i>fixed</i> , defines a byte pattern used to perform a match during the unmarshal operation, as further specified by the <b>align</b> property (see <a href="#">Defining Byte Patterns</a> on page 223).
nodeType	Specifies the marshal/unmarshal format. The default value is <b>delim</b> , but is defaulted to parent nodeType if parent nodeType is <i>fixed</i> or <i>trans</i> .
showDelim	Displayed only if <b>nodeType</b> is <i>delim</i> . The default value is <b>-none-</b> ; once delimiters are specified, value field shows the delimiters.
align	Specifies the byte alignment criteria for the <b>match</b> property (see <a href="#">Specifying Pattern Alignment</a> on page 223).
decoding	Displayed only if <b>nodeType</b> is <i>fixed</i> . Specifies the <i>unmarshal</i> coding (see <a href="#">Specifying Data Conversion and Encoding</a> on page 151). (It is recommended to use UTF-8 for DBCS data, since the hex value of some ASCII delimiter may coincide with a hex value contained within a double-byte character.)
encoding	Displayed only if <b>nodeType</b> is <i>fixed</i> . Specifies the <i>marshal</i> coding (see <a href="#">Specifying Data Conversion and Encoding</a> on page 151).
length	Displayed only if <b>nodeType</b> is <i>fixed</i> . Specifies the length of the field; the default value is <b>0</b> .

## 8.5 Specifying the Node Type

Double-clicking the **nodeType** properties field activates the field for editing. Click the arrow button to display the menu (see Figure 173). Descriptions of the property options are listed in Table 40.

**Figure 173** nodeType Property Menu (Root/Element Example)



**Table 40** nodeType Property Options

Option	Description
alter	Applies only to <i>Choice Element</i> nodes. Alter ( <i>alternate</i> ) selects one child or another. One, and only one, child is always present after the unmarshal operation.
array	Array is a delimited structure. If repeated, occurrences are separated by the <i>repeat</i> delimiter. The last occurrence may be terminated by a <i>normal</i> delimiter. Does not apply to <i>Choice Element</i> nodes.
delim	Delim ( <i>delimited</i> ) structure. If repeated, occurrences are separated by a <i>normal</i> delimiter. Does not apply to <i>Choice Element</i> nodes. See <a href="#">Specifying Delimiters</a> on page 225.
fixed	Fixed indicates a fixed length, which is specified by non-negative integer (or zero to indicate end of parent node data). Does not apply to <i>Choice Element</i> nodes.
group	Group provides organizational grouping for purposes such as repetition. Does not apply to <i>Choice Element</i> or <i>Field</i> nodes.
trans	Trans ( <i>transient</i> ) appears only in an internal tree as a scratchpad field. It does not appear in external data representation, and can only have <i>trans</i> nodeTypes as children.



### 8.5.1 Default Values

The basic default value for the `nodeType` property is **delim**. If, however, the node is the child of a parent node whose `nodeType` is **fixed** or **trans**, then the child takes on the same `nodeType` as the parent (see Table 41). This rule does not apply to *Choice Element* nodes.

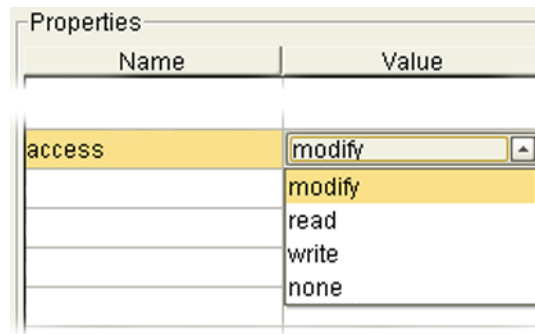
**Table 41** `nodeType` Default Values

Parent	Child
alter	N/A
array	delim
delim	delim
fixed	fixed
group	delim
trans	trans

## 8.6 Specifying Read/Write Access

The *access* property restricts read/write access to the associated element or field node.

**Figure 174** Access Property Menu



**Table 42** Access Parameter Options

Option	Description
modify	Read and write access, generates both <i>get</i> and <i>set</i> methods. This is the default setting.
read	Read access only, generates <i>get</i> method.
write	Write access only, generates <i>set</i> method.
none	Neither read nor write access, does not generate either <i>get</i> or <i>set</i> method.

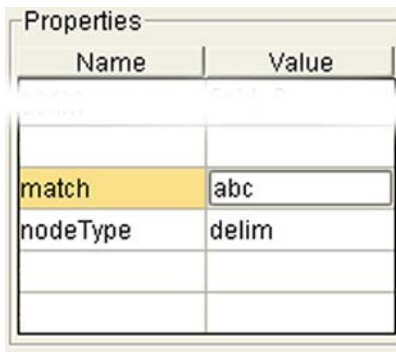
## 8.7 Matching Data Patterns

One of the parsing techniques that can be applied to the unmarshaling of an input data stream is that of matching a specific byte pattern within a data sequence. You can accomplish this in a User-Defined OTD by using the **match** and **align** field-node properties, when the nodeType is either **delim** or **fixed**. During the unmarshal operation, a field is successfully matched if it complies with the value of the **match** property, interpreted according to the value of the **align** property, as set for that field.

### 8.7.1 Defining Byte Patterns

The value you enter for the **match** property defines the byte pattern for the data you want to match. As an example, a value of **abc** has been entered into the value field shown in Figure 175 (provides a reference for Table 43).

**Figure 175** Match Property

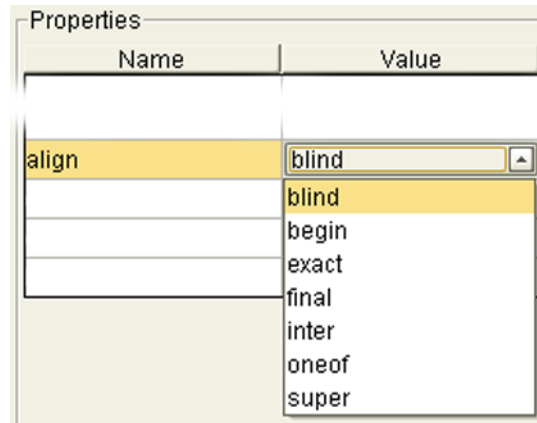


Properties	
Name	Value
match	abc
nodeType	delim

### 8.7.2 Specifying Pattern Alignment

The **align** property supplements the **match** property, specifying criteria on which to base the match. The default value is **blind**; if this is specified, the match property has no meaning.

**Figure 176** Align Property Menu



**Table 43** Align Parameter Options

Option	Description
blind	Always performs a match (default value). Any value set for the <b>match</b> property is ignored.
begin	When the leading bytes of an input byte sequence match the value set for the <b>match</b> property (for example, [ <b>abc</b> .....]), the unmarshal method matches the field to the input byte sequence.
exact	When an input byte sequence exactly matches the specified byte pattern (for example, [ <b>abc</b> ]), the unmarshal method matches the field to the input byte sequence.
final	When the trailing bytes of an input byte sequence match the value set for the <b>match</b> property (for example, [..... <b>abc</b> ]), the unmarshal method matches the field to the input byte sequence.
inter	When the input byte sequence contains a byte pattern that includes the value set for the <b>match</b> property, (for example, [... <b>abc</b> ...]), the unmarshal method matches the field to the input byte sequence.
oneof	If the value set for the <b>match</b> property is a repeating pattern of the form <separator><value>... (for example, [\bmon\bwed\bfri]), and the input byte sequence contains a byte pattern that matches one of the <value> entries (for example, [ <b>wed</b> ]), the unmarshal method matches the field to the input byte sequence.
super	When an input byte sequence is a subsequence of the value set for the <b>match</b> property (for example, [ <b>bc</b> ]), the unmarshal method matches the field to the input byte sequence.

**Note:** The value entered for the **match** property is interpreted as a Latin1 string, rather than following the specified encoding.

## 8.8 Specifying Delimiters

### 8.8.1 Delimiter List

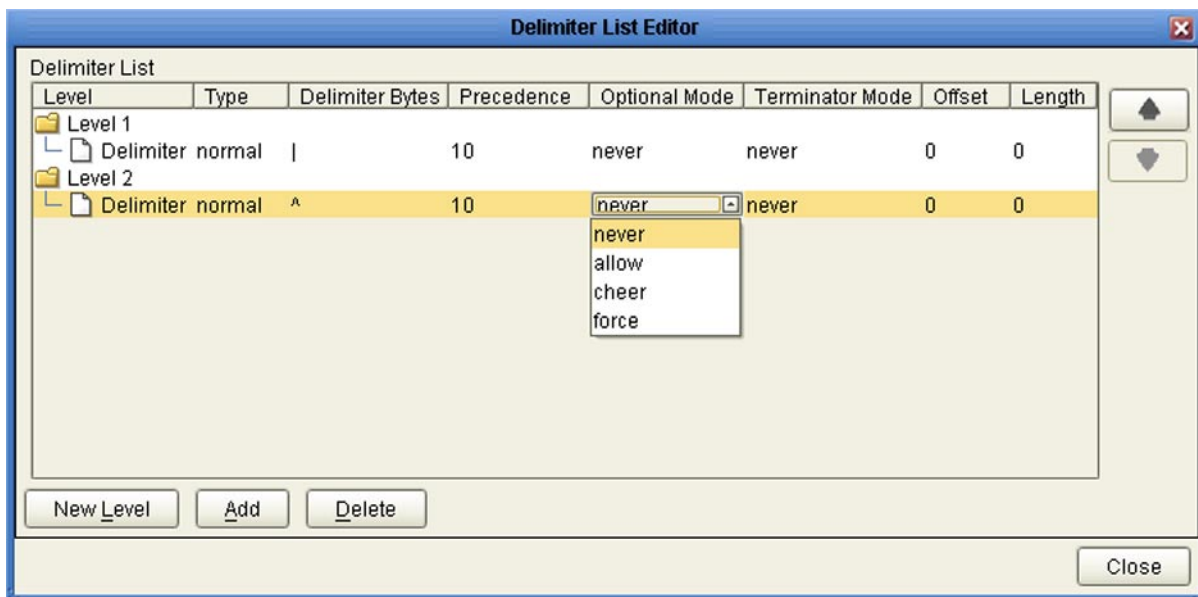
You can define a set of delimiters — a *delimiter list* — for any node in the hierarchical data structure. This delimiter list is used in the external data representation for that node and its descendants. A delimiter list defined for any node overrides the effect of any ancestor node's delimiter list on the node and its descendants.

Delimiters are defined using the Delimiter List Editor (see Figure 177). The editor is invoked by clicking in the **delim** property *value* field and clicking the ellipsis (...) button, or by double-clicking in the field. See [Creating a Delimiter List](#) on page 236.






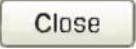
Clicking within a field in the Delimiter List Editor enables the field for editing. Double-clicking within one of the following three fields displays its menu, as illustrated in Figure 177.

- Type
- Optional Mode
- Terminator Mode

**Figure 177** Delimiter List Editor



**Table 44** Delimiter List Editor Command Buttons

Button	Command	Action
	New Level	Adds a new level after the selected level.
	Add Delimiter	Adds a new delimiter after the selected delimiter, or to the bottom of list under the selected level.
	Delete	Deletes the selected line item (level or delimiter) from the list.
	Move Up	Moves the selected level up one level; moves the selected delimiter up one step within the same level or to the bottom of the previous level.
	Move Down	Moves the selected level down one level; moves the selected delimiter down one step within the same level, or to the top of the next level (if one exists).
	Close	Closes the editor and saves your entries.

## 8.8.2 Delimiter Properties

**Table 45** Delimiter Properties

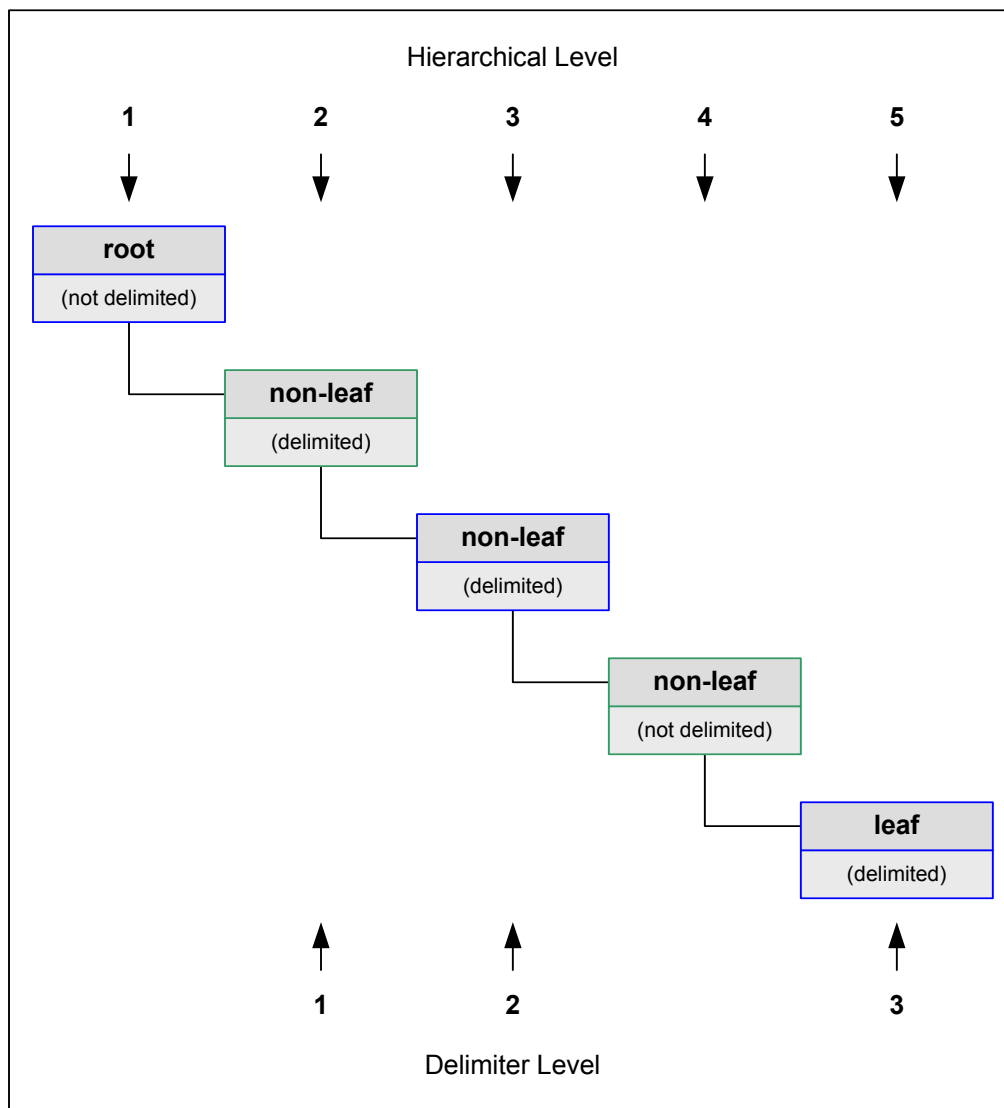
Property	Description
Level	Delimiter levels, assigned consecutively to delimited nodes in the OTD node hierarchy (see <a href="#">Delimiter Levels</a> on page 227).
Type	See <a href="#">Delimiter Type</a> on page 229.
Delimiter Bytes	Delimiter characters (see <a href="#">Delimiter Bytes</a> on page 230). <b>Note:</b> Entering a value for <i>Length</i> (see below) clears this field.
Precedence	See <a href="#">Precedence</a> on page 232.
Optional Mode	See <a href="#">Optional Mode</a> on page 233.
Terminator Mode	See <a href="#">Terminator Mode</a> on page 235.
Offset	Offset of the data field in bytes from the beginning of the data stream (byte 0). Value must be a non-negative integer; the default is <b>0</b> .
Length	Length of the data field in bytes. Value must be positive integer. Entering a value clears the <i>Delimiter Bytes</i> field.

### 8.8.3 Delimiter Levels

Delimiter levels are assigned in order to those hierarchical levels of an OTD that contain at least one node that is specified as being delimited. If none of the nodes at a particular hierarchical level is delimited, that hierarchical level is skipped in assigning delimiter levels.

Delimiter lists are typically specified on the root node, so that the list applies to the entire OTD. The root node itself is typically not delimited, so that *Level 1* would apply to those nodes that are children of the root node. See Figure 178 and the following example.

**Figure 178** OTD Hierarchical and Delimiter Levels



For example, if you want to parse the following data:

a^b|c^d|e

you might create a User-Defined OTD as follows:

- root
  - ♦ element\_1
    - ♦ field\_1
    - ♦ field\_2
  - ♦ element\_2
    - ♦ field\_3
    - ♦ field\_4
  - ♦ field\_5

In this example, the delimiter list is specified on the *root* node, which is not delimited; therefore, the list has two levels:

- Level 1
  - ♦ Delimiter |
- Level 2
  - ♦ Delimiter ^

The *Level 1* delimiter (|) applies to element\_1, element\_2, and field\_5. The *Level 2* delimiter (^) applies to field\_1 - field\_4.

If the root node is set to be delimited, the *Level 1* delimiters will then apply to it. Using the above example, the *Level 2* delimiter (^) would then apply to element\_1, element\_2, and field\_5, and a new *Level 3* delimiter would apply to field\_1 - field\_4.

Delimiter lists can be much more complex than this very simple example. For instance, you can create multiple delimiters of different types at any given level, and you can specify a delimiter list on any node within the OTD — not only the root node as shown in the example. See [Creating a Delimiter List](#) on page 236 for a step-by-step description of the procedure for creating a Delimiter List.



## 8.8.4 Delimiter Type

The *Delimiter Type* property specifies whether the delimiter is a terminator at the end of the byte sequence (*normal*), a separator between byte sequences in an array (*repeat*) or an escape sequence.

**Table 46** Delimiter Type Options

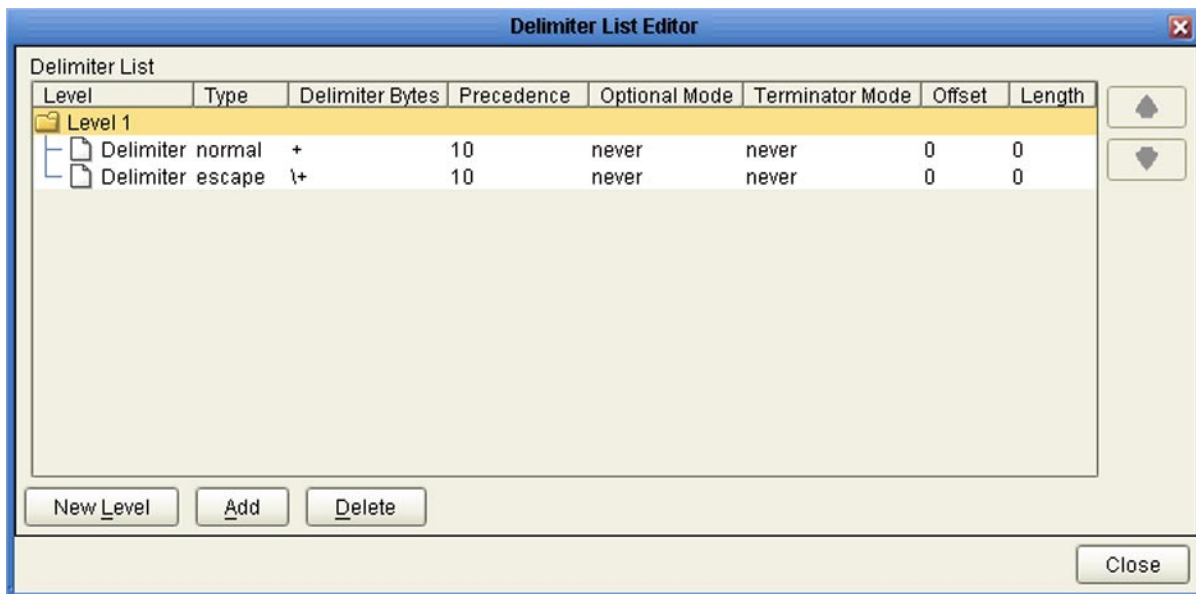
Option	Description
escape	Escape sequence.
repeat	Array delimiter/separator.
normal	Terminator.

### Escape Option

An *escape* delimiter is simply a sequence that is recognized *and ignored* during parsing. Its purpose is to allow the use of escape sequences to embed byte sequences in data that would otherwise be seen as delimiter occurrences.

For example, if there is a normal delimiter “+” at a given level, and we define an escape delimiter “\+” (see Figure 179), then **aaa+b\+c+ddd** will parse as three fields: **aaa**, **b\+c**, and **ddd**. If the escape delimiter were not defined, the sequence would then parse as four fields: **aaa**, **b\**, **c**, and **ddd**.

**Figure 179** Delimiter Type - Escape



If there is *only* an escape delimiter on a given level, however, it presents a *no delimiter defined* situation for **delim** and **array** nodes.

## 8.8.5 Delimiter Bytes

There is essentially no limitation on what characters you can use as delimiters; however, you obviously want to avoid characters that can be confused with data or interfere with escape sequences (see [Escape Option](#) on page 229). The backslash (\) is normally used as an escape character (the HL7 protocol uses a double backslash as part of an escape sequence that provides special text formatting instructions).

**Note:** *You should avoid using a colon (:) as a delimiter character, since it is used as a literal in system-generated time strings. This can interfere with recovery procedures, for example following a Domain shutdown.*

## Escape Sequences

Use a backslash (\) to escape special characters. Table 47 lists the currently supported escape sequences.

**Table 47** Escape Sequences

Sequence	Description
\\	Backslash
\b	Backspace
\f	Linefeed
\n	Newline
\r	Carriage return
\t	Tab
\ddd	Octal number
\xdd	Hexadecimal number

For octal values, the leading variable **d** can only be **0 - 3** (inclusive), while the other two can be **0 - 7** (inclusive). The maximum value is **\377**.

For hexadecimal values, the variable **d** can be **0 - 9** (inclusive) and **A - F** (inclusive, either upper or lower case). The maximum value is **\xFF**.

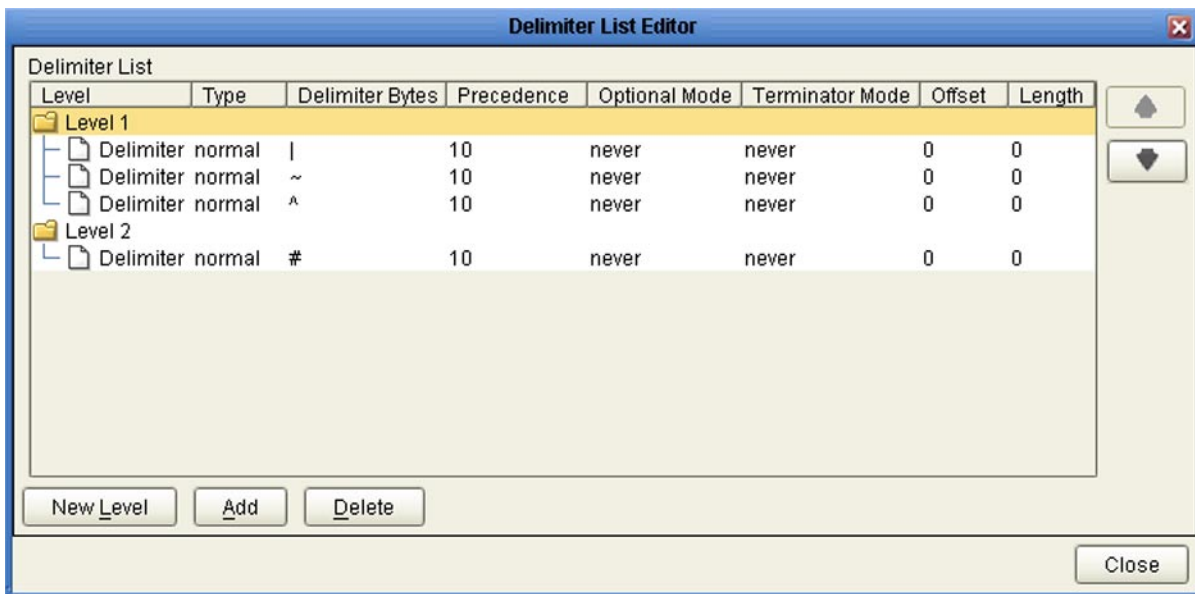
## 8.8.6 Multiple Delimiters

You can specify multiple delimiters at a given level; for example, if you specify |, ~, and ^ as delimiters for a specific level (see Figure 180), the parser will accept any of these delimiters:

- root
  - ♦ element (delimiters = "|", "~", "^")
    - ♦ field\_1 (delimiter = "#")
    - ♦ field\_2 (delimiters = "|", "~", "^")

This will successfully parse the data **abc|def**, **abc~def**, and **abc^def**.

**Figure 180** Multiple Delimiter Example



### 8.8.7 Precedence

Precedence (see [Figure 190 on page 239](#)) indicates the priority of a certain delimiter, relative to the other delimiters. Precedence is used to resolve delimiter conflicts when one delimiter is a copy or prefix of another. In case of equal precedence, the innermost prevails.

By default, all delimiters are at precedence 10, which means they are all considered the same; fixed fields are hard-coded at precedence 10. Delimiters on parent nodes are not considered when parsing the child fields; only the child's delimiter (or if it is a fixed field, its length). The range of valid precedence values is from 1 to 100, inclusive.

Changing the precedence of a delimiter will cause them to be applied to the input data-stream in different ways. For example:

- root
  - ♦ element (type delim, delimiter = "^", repeat)
    - ♦ field\_1 (type fixed, length = 5)
    - ♦ field\_2 (type fixed, length = 8, optional)

Although this will parse 'abcde12345678^zyxvuABCDEFGH', it will *not* parse the text 'abcde^zyxvuABCDEFGH' even though the second fixed field is optional. The reason is that the element's delimiter is ignored within the fixed field because they have the same precedence. If you want the element's delimiter to be examined within the fixed field data, you must change its precedence, for example:

- root
  - ♦ element (type delim, delimiter = "^", repeat, **precedence = 11**)
    - ♦ field\_1 (type fixed, length = 5)
    - ♦ field\_2 (type fixed, length = 8, optional)

This will successfully parse the text 'abcde^zyxvuABCDEFGH'.

A similar argument can be applied to delimited child nodes. The parser normally attempts to match the child delimiter — setting the precedence to 11 forces the parser to match the parent delimiter first.

## 8.8.8 Optional Mode

The *Optional Mode* property specifies how delimiters for optional nodes are to be handled when the nodes are absent from the input instance or when their fields are empty.

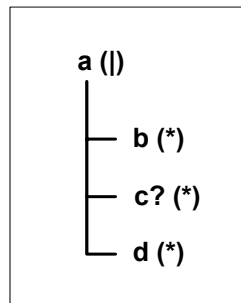
**Table 48** Optional Mode Options

Option	Rule
never	Do not allow on input, do not emit on output (empty field between delimiters implies zero length data field).
allow	Skip empty field if present; if absent, do not delimit on output.
cheer	Skip empty field if present; if absent, delimit on output.
force	Require empty, delimited field on input; always delimit on output. <b>Note:</b> Only this option allows trailing delimiters for a sequence of absent optional nodes.

As illustrative examples, consider the tree structures shown in Figure 181 and Figure 182, where the node **a** has a pipe (|) as its delimiter, and the child nodes **b**, **c**, and **d** all have asterisks (\*) as their delimiters.

- **Example 1:** Child node **c** is *optional*. (Child nodes **c** and **d** must have different values for the *match* parameter.)

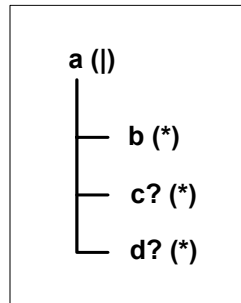
**Figure 181** Optional Mode Property (Example 1)



Option	Input	Output
never	<b>b*d </b>	<b>b*d </b>
allow	<b>b**d </b>	<b>b*d </b>
cheer	<b>b**d </b>	<b>b**d </b>
force	<b>b**d </b>	<b>b**d </b>

- **Example 2:** Child nodes **c** and **d** are both *optional*.

**Figure 182** Optional Mode Property (Example 2)



Option	Input	Output
never	<b>b </b>	<b>b </b>
allow	<b>b , b* , or b** </b>	<b>b </b>
cheer	<b>b , b* , or b** </b>	<b>b** </b>
force	<b>b** </b>	<b>b** </b>

### 8.8.9 Terminator Mode

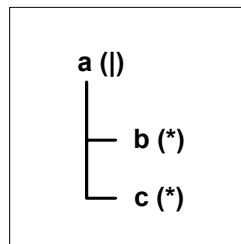
The *Terminator Mode* property specifies how delimiters are to be handled for a specific terminator node in the OTD tree.

**Table 49** Terminator Mode Options

Option	Rule
never	Do not allow on input, do not emit on output (pure separator).
allow	Allow on input, do not emit on output.
cheer	Allow on input, always emit on output.
force	Require on input, always emit on output (pure terminator).

Consider the tree structure shown in Figure 183, where the node **a** has a pipe (|) as its delimiter, and its child nodes **b** and **c** have asterisks (\*) as their delimiters.

**Figure 183** Terminator Mode Property Example

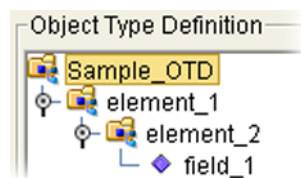


Option	Input	Output
never	c	c
allow	c  or c*	c
cheer	c  or c*	c*
force	c*	c*

## 8.9 Creating a Delimiter List

As an example, we shall create a delimiter list for the simple OTD structure shown in Figure 184.

**Figure 184** Sample OTD Tree



To create a delimiter list

- 1 In the OTD Editor, select the node for which you want to define a set of delimiters (this example uses the *root* node). Initially, the value for the **nodeType** property is set to *group* and the value for the **delim** property appears as *not set*, as shown in Figure 185.

**Figure 185** Initial Root Node Properties

Properties	
Name	Value
name	Sample_OTD
javaName	SampleOTD
comment	
delim	not set
nodeType	group

- 2 Click in the **nodeType** value field to display the option menu, and select **delim**. The **showDelim** property fields automatically appear.
- 3 Click in the **delim** value field to activate it for editing; an ellipsis (...) button appears, as shown in Figure 186.

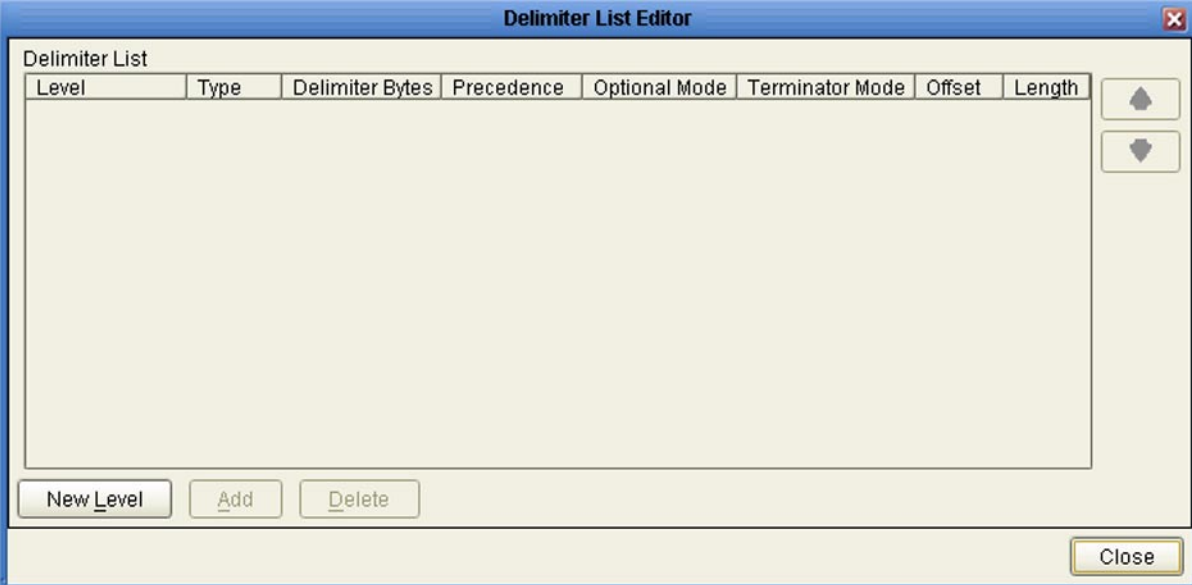


**Figure 186** Activated delim Value Field

Properties	
Name	Value
name	Sample_OTD
javaName	SampleOTD
comment	
access	modify
delim	not set <input data-bbox="1036 485 1068 516" type="button" value="..."/>
nodeType	delim
showDelim	-none-
antecoding	

- Click the ellipsis button to display the Delimiter List Editor, which is initially blank (see Figure 187).

**Figure 187** Delimiter List Editor - Delimiters Not Set

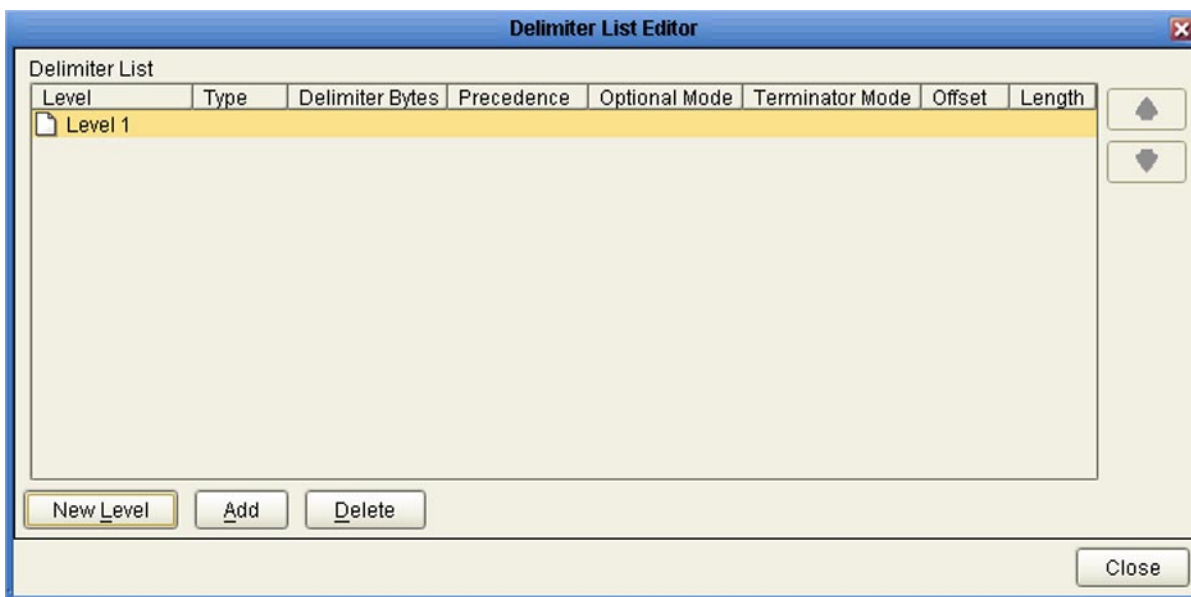


Level	Type	Delimiter Bytes	Precedence	Optional Mode	Terminator Mode	Offset	Length
-------	------	-----------------	------------	---------------	-----------------	--------	--------

Buttons: New Level, Add, Delete, Close

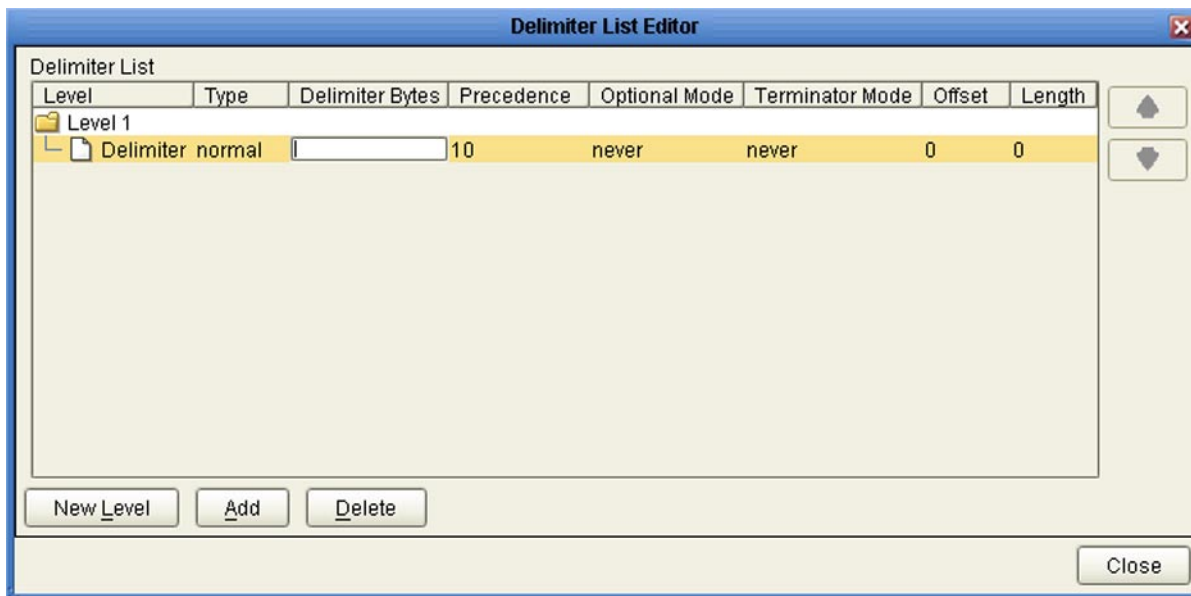
- Click **New Level** to add a level to the delimiter list, as shown in Figure 188.

**Figure 188** Delimiter List Editor - Insert Level



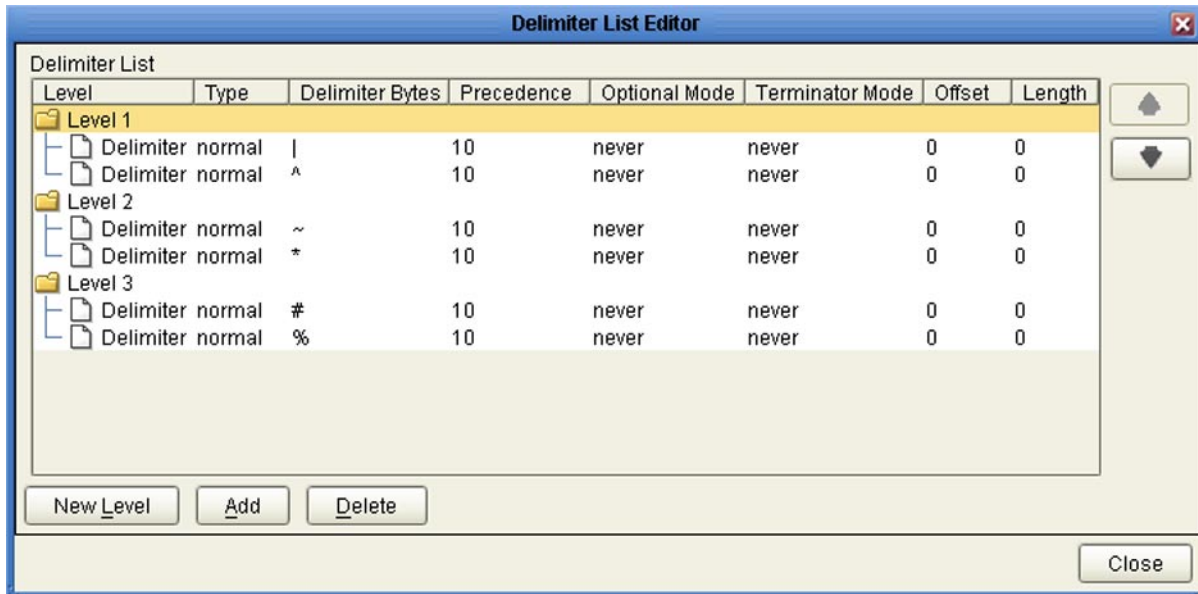
- 6 Select a level and click **Add Delimiter** to add a delimiter to the selected level. Double-click in the Bytes field and type in the delimiter characters (see Figure 189).

**Figure 189** Delimiter List Editor - Add Delimiter



- 7 Continue adding levels and delimiters as required (see Figure 190).

**Figure 190** Delimiter List Editor - Add Levels and Delimiters



- 8 Click **Close** to close the editor and save your work.
- 9 The value for the **delim** property will now appear as *specified*, as shown in Figure 191.

**Figure 191** Root Node - Delimiter Specified

Properties	
Name	Value
name	Sample_OTD
javaName	SampleOTD
comment	
delim	specified
nodeType	delim
showDelim	,^
encoding	

- 10 Since the **nodeType** values for elements and fields are *trans* by default, you must reset their **nodeType** properties to *delim* to assign delimiters to them.
  - A The properties for *Element\_1* are displayed in Figure 192. Leave the **nodeType** property as *trans*, and do not delimit this node. If you change the **nodeType** property to *delim*, it would automatically pick up the delimiters for *Level 2*, since the existing delimiter list is defined for the *root* node. Defining another delimiter list here would override the existing list.

**Figure 192** Element\_1 Node Properties

Properties	
Name	Value
name	element_1
javaName	Element1
comment	
access	modify
optional	false
repeat	false
maxOccurs	-1
delim	not set
nodeType	trans
order	seq

- B** Delimit *Element\_2* by setting its **nodeType** property to *delim*; it automatically picks up the delimiters for *Level 2* from the list defined for the *root* node, as displayed in Figure 193. The **delim** property remains *not set*, indicating that the delimiter list is not set on *this* node. Defining another delimiter list here would override the list defined on the *root* node, and the **delim** property would change to *specified*.

**Figure 193** Element\_2 Node Properties

Properties	
Name	Value
name	element_2
javaName	Element2
comment	
access	modify
optional	false
repeat	false
maxOccurs	-1
delim	not set
nodeType	delim
showDelim	~,*
order	seq

- C** Delimit *Field\_1* by setting its **nodeType** property to *delim*; it automatically picks up the delimiters for *Level 3* from the list defined for the *root* node, as displayed in Figure 194. Again, the **delim** property remains *not set*.

**Figure 194** Field\_1 Node Properties

Properties	
Name	Value
name	field_1
javaName	Field1
javaType	java.lang.String
comment	
access	modify
optional	false
repeat	false
maxOccurs	-1
delim	not set
initial	
match	
nodeType	delim
showDelim	#,%
align	blind

- 11 Once you have defined your delimiter list, you should test the OTD to verify that it parses correctly (see [OTD Tester](#) on page 162).

# Collaboration Definitions (Java) I

This chapter describes the features of the Collaboration Definition Editor (Java™) and associated tools.

## What's in This Chapter

- [Overview](#) on page 242
- [Using the Collaboration Definition Context Menu](#) on page 244
- [Using the Collaboration Editor \(Java\)](#) on page 246
- [Graphical Collaboration Methods \(Java\)](#) on page 263
- [Using the Class Browser](#) on page 282
- [Using the Method Dialog](#) on page 287
- [Auto Type Conversion](#) on page 290
- [Using the Collaboration Tester \(Java\)](#) on page 296
- [Using the Java Debugger](#) on page 303

---

## 9.1 Overview

Collaboration Definitions define how data should be processed and routed between Project components, how databases should be queried in response to requests, and how APIs to one or more applications should be invoked. The external data formats that characterize the input and output data structures in a Collaboration Definition are described by Object Type Definitions (OTDs).

A Collaboration will typically receive a message containing the external representation of a particular OTD. It will use the *unmarshal* method of an instance of that OTD to parse the data and make it accessible through the hierarchical data structure. Then it will perform some operation — for example, copying parts of the data to another OTD instance. Finally, it will invoke the *marshal* method on the other OTD instance to render the contents of its data structure as a single, serialized data stream for further transport.

At run time, an OTD instance is accessed directly from Java, using accessors resembling JavaBeans. Each of the nodes comprising the hierarchy of the data structure has a set of properties with *get* and *set* methods.

**Important:** *If you delete an OTD in the Project Explorer, any Collaboration Definitions that have been built using that OTD will be affected. It is recommended that you run the Impact Analyzer before attempting to delete any OTDs (see [Analyzing Component Dependencies](#) on page 99).*

As with other eGate Integrator components, it is essential to manage versions of Collaboration Definitions carefully. See [Managing Component Versions](#) on page 78 for descriptions of various version control features applicable to Collaboration Definitions.

The Enterprise Designer includes two primary tools, the Collaboration Definition Wizard (Java) and Collaboration Editor (Java), that you use to create and customize your Java-based Collaboration Definitions. These tools are described in the following sections.

## 9.2 Using the Collaboration Definition Context Menu

**Figure 195** Collaboration Definition Context Menu



**Table 50** Collaboration Definition Context Menu Options

Option	Function
Open	Opens the appropriate Collaboration Editor, showing the selected Collaboration Definition. See <a href="#">Using the Collaboration Editor (Java)</a> on page 246.
ACL Management	Presents the ACL Properties dialog, with which an Administrator can assign read/write privileges to users for the selected Collaboration Definition. See the <i>Sun SeeBeyond eGate™ Integrator System Administration Guide</i> .
Version Control	Accesses the version control features for the selected component. See <a href="#">Accessing Component Version Control Features</a> on page 79 for additional information.
Cut	Copies the selected Collaboration Definition (Java only) and removes it from the current Project, after which you can paste it to another Project (once only). All changes must be committed before you can cut the Collaboration. Cut and paste is disabled for other users when you have the Collaboration checked out. See <a href="#">Moving and Replicating Repository Objects</a> on page 97 for additional information.
Copy	Copies the selected Collaboration Definition (Java only), after which you can paste it to other Projects (multiple times). All changes must be committed before you can copy the Collaboration. You can copy and paste a Collaboration even when another user has the Collaboration checked out. See <a href="#">Moving and Replicating Repository Objects</a> on page 97 for additional information.



**Table 50** Collaboration Definition Context Menu Options

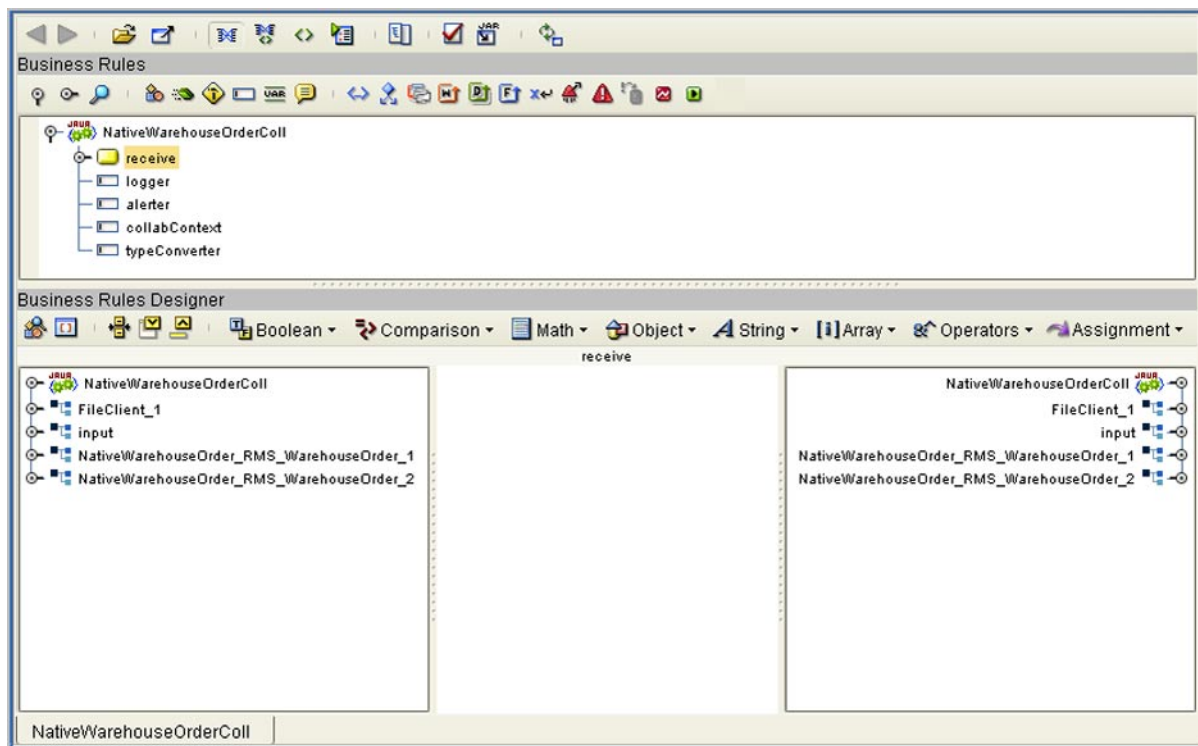
Option	Function
Create Diff	Presents a dialog with which you can create a difference file representing two versions of the selected Collaboration Definition. See <a href="#">Merging Changes from Another Version</a> on page 94.
Merge Diff	Presents a dialog with which you can merge modifications into a specific version of the selected Collaboration Definition. See <a href="#">Merging Changes from Another Version</a> on page 94.
Properties	Presents the appropriate Collaboration Definition Properties dialog for the selected Collaboration Definition.
Generate Report	Presents a dialog in which you can generate an HTML-formatted report on the currently open Collaboration. The primary utility for this feature is for business processes.
Delete	<p>Deletes the selected Collaboration Definition, subject to the following conditions:</p> <ul style="list-style-type: none"> <li>▪ You have <i>write</i> privileges for the component (see ACL Management, above).</li> <li>▪ You have the component checked out to your workspace.</li> <li>▪ The component is not checked out by anyone other than yourself.</li> </ul> <p>If these conditions are met, a dialog is presented in which you confirm that you want to delete the selected component. Clicking <b>Yes</b> then deletes the component.</p>
Rename	Activates the field, allowing you to rename the selected Project.

## 9.3 Using the Collaboration Editor (Java)

The Collaboration Editor (Java) displays the structure of the selected Java-based Collaboration Definition, and allows you to define the transformations that the Collaboration will perform at run time.

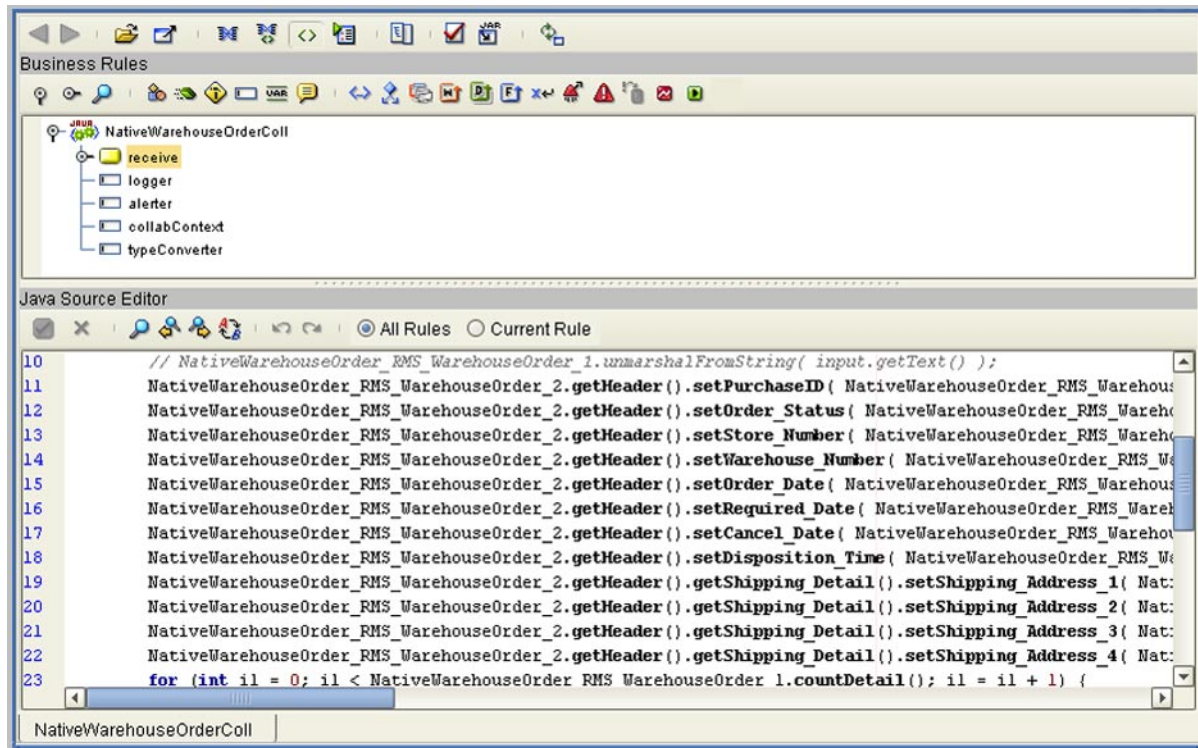
After you create a Java-based Collaboration Definition file using the Collaboration Definition Wizard (Java), the Collaboration Editor (Java) appears in the editor panel of the Enterprise Designer (see Figure 196). By default, the Business Rules Editor and Business Rules Designer panels are displayed.

**Figure 196** Collaboration Editor (Java)



Selecting the *Source Code Mode* icon (see [Collaboration Editor Toolbar Icons](#) on page 248) substitutes the Java Source Editor for the Business Rules Designer, as shown in Figure 197.

**Figure 197** Collaboration Editor (Java) - Source Code Mode




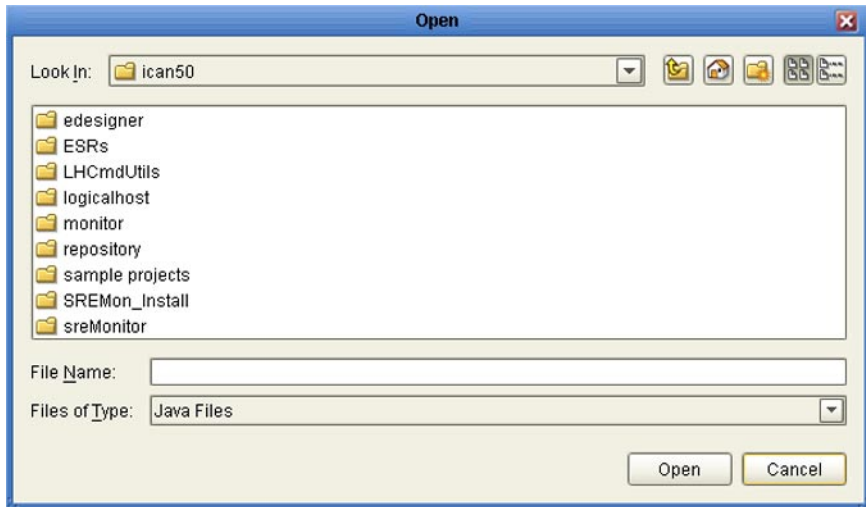

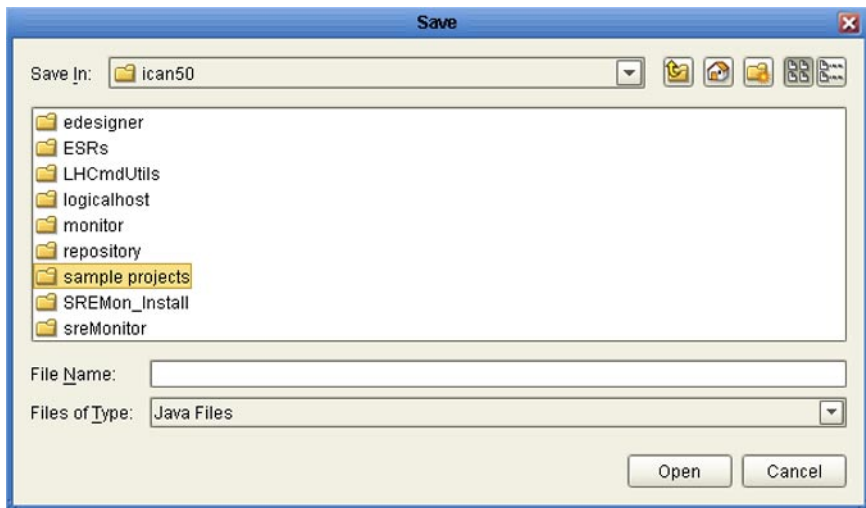


Selecting the *Advanced Mode* icon (see [Collaboration Editor Toolbar Icons](#) on page 248) displays both the Java Source Editor and the Business Rules Designer, in addition to the Business Rules Editor. These three parts of the Collaboration Editor (Java) are described in the following sections:








- [Business Rules Editor](#) on page 251.
- [Business Rules Designer](#) on page 256.
- [Java Source Editor](#) on page 254.

### 9.3.1 Collaboration Editor Toolbar Icons


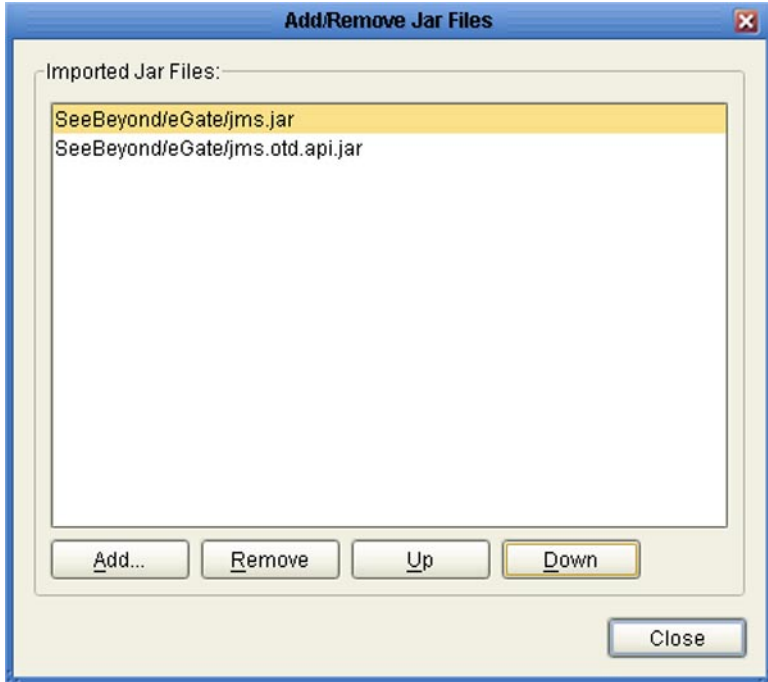

**Table 51** Collaboration Editor Toolbar Icons

Icon	Command	Function
	Back	Steps back in your usage history for the current session.
	Forward	Steps forward in your usage history for the current session.
	Import File	<p>Presents a dialog with which you can locate and select a Collaboration Definition (Java) to import. When you import a file, the imported code replaces any existing code.</p> 
	Export File	<p>Presents a dialog with which you can save the selected Collaboration Definition (Java) to a file.</p> 

**Table 51** Collaboration Editor Toolbar Icons

Icon	Command	Function
	Standard Mode	Displays the Business Rules Editor and Business Rules Designer only (default setting).
	Advanced Mode	Displays the Business Rules Editor, Business Rules Designer, and Java Source Editor.
	Source Code Mode	Displays the Business Rules Editor and Java Source Editor only.
	Test Mode	Displays or hides the Java Collaboration Tester (see <a href="#">Using the Collaboration Tester (Java)</a> on page 296).
	Business Rules on Left	Changes the editor layout to display the Business Rules panel to the left of the Business Rules Designer. Toggles with <i>Business Rules on Top</i> .
	Business Rules on Top	Changes the editor layout to display the Business Rules panel above the Business Rules Designer (default setting). Toggles with <i>Business Rules on Left</i> .
	Validate	Verifies that changes made to the Java code are valid.

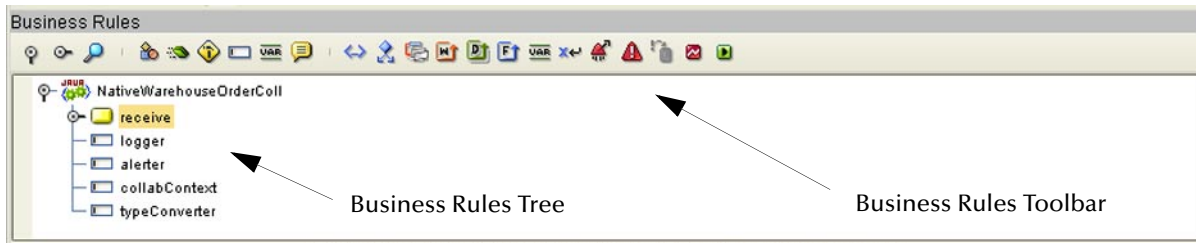
**Table 51** Collaboration Editor Toolbar Icons

Icon	Command	Function
	Import JAR File	<p>Presents a dialog with which you can import a <b>.jar</b> file (see <a href="#">Importing Third-Party Java Classes</a> on page 352).</p> 
	Refresh Collaboration	Refreshes the Collaborations from the Repository.

## 9.3.2 Business Rules Editor


The Business Rules Editor lists each business rule included with the Java-based Collaboration Definition. This editor has its own toolbar which you use to add business rules to the Collaboration Definition (see Figure 198). You add rules simply by dragging and dropping the rules into the Business Rules tree. Click the **Save** icon in the Enterprise Designer toolbar when you are finished.

**Figure 198** Business Rules Editor
















### Business Rules Editor Toolbar Icons

**Table 52** Business Rules Editor Toolbar Buttons




Icon	Command	Function
	Expand All Rules	Expands the tree to show all rules.
	Collapse All Rules	Collapses all rules in the tree.
	Find	Presents a dialog in which you can enter text to search for in the Java Source Editor.  
	Class	Presents the <i>Create Class</i> dialog (see <a href="#">Creating Classes</a> on page 327).
	Method	Presents a dialog with which you can add a new method to the Business Rules tree (see <a href="#">Creating Methods</a> on page 329).

**Table 52** Business Rules Editor Toolbar Buttons

Icon	Command	Function
	Constructor	Presents the <i>Create Constructor</i> dialog (see <a href="#">Creating Constructors</a> on page 333).
	Field	Presents a dialog with which you can add a new field to the Business Rules tree (see <a href="#">Creating Fields</a> on page 336).
	Local Variable	Presents a dialog with which you can add a local variable to the Business Rules tree.
	Comment	Presents a dialog in which you can enter a comment.
	Rule	Adds a new (empty) <b>rule</b> statement to the Business Rules tree. Use this command as a safeguard regarding positioning.
	If-Then	Adds an <b>if-then</b> statement to the Business Rules tree.
	Switch	Adds a <b>switch</b> statement to the Business Rules tree, which provides an alternative to the <b>if-then</b> statement.
	While	Adds a <b>while</b> statement to the Business Rules tree, which starts a repetition of business rules if a specified condition is true. You can configure the condition using drag-and-drop when the <b>while</b> statement is selected.
	Do-While	Adds a <b>do-while</b> statement to the Business Rules tree, which performs a set of business rules which repeats if a specified condition is true. You can configure the condition using drag-and-drop when the <b>do-while</b> statement is selected.
	For	Adds a <b>for</b> statement to the Business Rules tree, starting a specific iteration (repetition) of business rules.
	Return	Adds a <b>return</b> statement to the Business Rules tree.
	Throw	Adds a <b>throw</b> statement for throwing exceptions.
	Try	Adds a <b>try</b> statement to the Business Rules tree, initiating a number of programming statements that are monitored to see whether they succeed or fail (see <a href="#">Using Try-Catch</a> on page 358). Automatically enables the <b>Catch</b> icon.



**Table 52** Business Rules Editor Toolbar Buttons

Icon	Command	Function
	Catch	Adds a <b>catch</b> statement to an existing <b>try</b> statement (disabled in the absence of a <b>try</b> statement).
	Break	Breaks out of a business rule loop, for example when using a <b>while</b> , <b>for</b> , or <b>switch</b> command.
	Continue	Continues the execution of business rules in a <b>for</b> or <b>while</b> loop by starting the next iteration.

## Business Rules Tree

The business rules tree consists of method and field nodes. These are the top level nodes on the tree and cannot be moved.

You add a **method** when you want to create a reusable set of instructions inside a Java-based Collaboration for a specific purpose. Methods are implemented as Java methods, and can be enhanced by means of parameters, return values, and access types (such as **public**, which means they are also available to other Java-based Collaborations).

You add a **field** when you want to create a *container* within a Java-based Collaboration for some specific purpose, such as storing a temporary variable. As soon as the field has been created, all other rules within the Collaboration are able to read or write from it.

The *rule*, *if-then*, *while*, *for*, *local variable*, *return*, and *try* statements can be added as subnodes to methods, but not to fields.

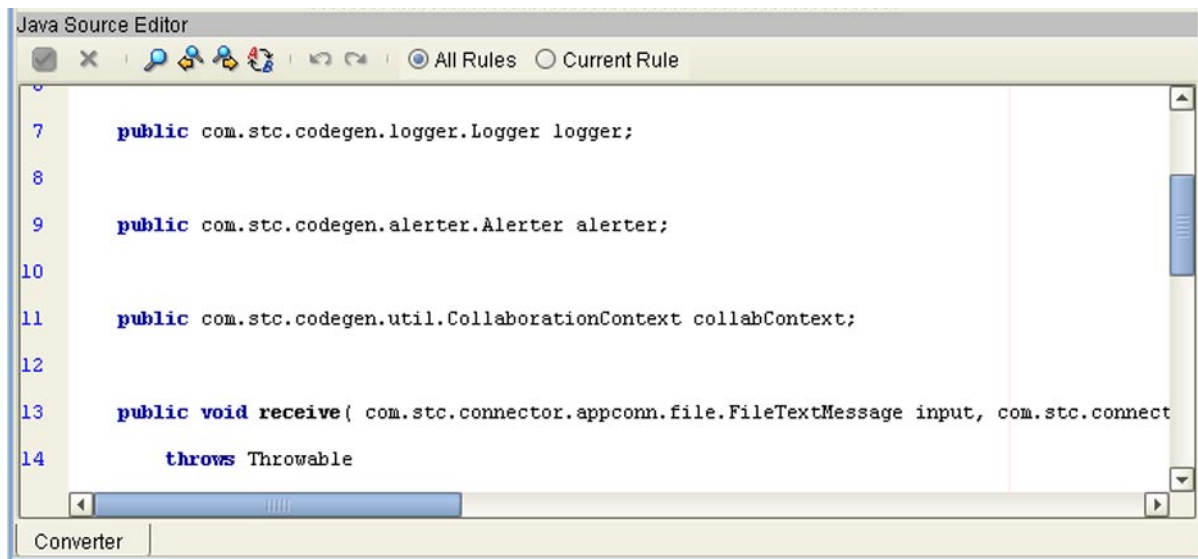
**Note:** *When adding a method as a sibling to another method, make sure you add it after the last rule in the preceding code block. Otherwise, it will be incorporated into the code block as a child of the preceding method.*

### 9.3.3 Java Source Editor

The Java Source Editor (see Figure 199) displays the actual Java code as you map the Collaboration Definition. You can display all rules, or only the rule currently selected in the Business Rules Editor, by selecting the appropriate option button in the toolbar. At your option, you can enter and edit the raw Java code for the Collaboration Definition. Click the **Commit Changes** button in the toolbar when you are finished.

**Note:** You must click either the Source Code Mode or Advanced Mode toolbar icon to display the Java Source Editor.



**Figure 199** Java Source Editor




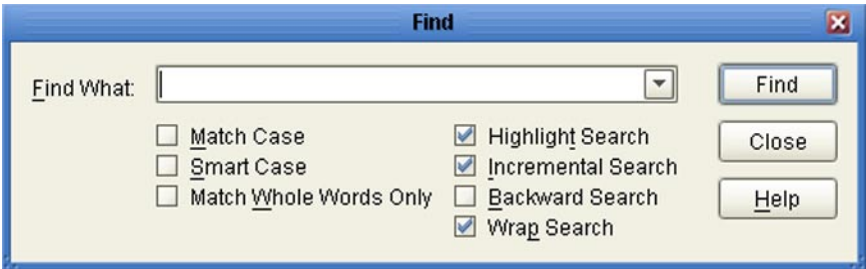






Code completion in the Java Source Editor shows standard Java library classes, classes from OTD .jar files, and classes from any imported third-party .jar files.

### Java Source Editor Toolbar Icons

**Table 53** Java Source Editor Toolbar Icons

Icon	Command	Function
	Commit Changes	Saves changes made in the Java Source Editor. Re-enables the Business Rules Editor and Business Rules Designer after using the Java Source Editor.
	Roll Back Changes	Cancels changes made in the Java Source Editor and returns the Java-based Collaboration Definition to the last saved state.

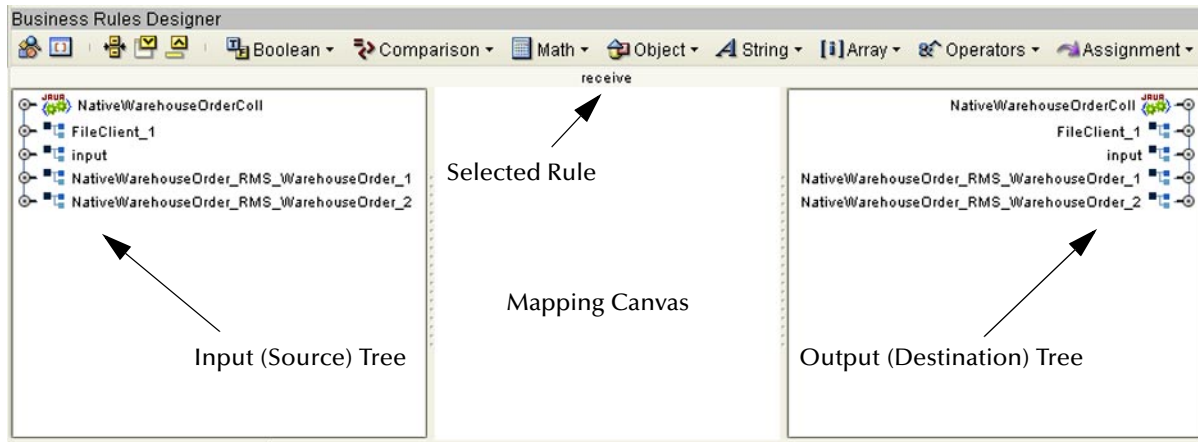
**Table 53** Java Source Editor Toolbar Icons

Icon	Command	Function
	Find	Opens the <b>Find</b> dialog, where you can enter text to search for in the Java Source Editor. 
	Find Previous	Searches backward for a previous instance of the text entered in the <b>Find</b> dialog.
	Find Next	Searches forward for the next instance of the text entered in the <b>Find</b> dialog.
	Replace	Opens the <b>Replace</b> dialog, where you can enter text to search for, and to replace with, in the Java Source Editor. 
	Undo	Undoes the last operation.
	Redo	Restores the last operation, if undone.

### 9.3.4 Business Rules Designer

The Business Rules Designer (see Figure 200) lists each field included in the Java-based Collaboration Definition, and allows you to graphically add methods and map data paths. The Business Rules Designer has its own toolbar containing commands and menus to facilitate the creation of Collaboration Definitions (see [Business Rules Designer Toolbar Icons](#) on page 257). These icons also appear in the context menu for each rule in the Business Rules Editor.

**Figure 200** Business Rules Designer



Input and output instances of all included OTDs are shown in the left and right panes, respectively. You can perform data transformation and mapping graphically on the mapping canvas, between the input and output JCD trees.

You can add new OTDs to an existing Collaboration Definition by right-clicking the Collaboration in Project Explorer to display the Collaboration Definition Properties dialog. You can edit the properties of an existing method or variable in a Collaboration by right-clicking the method or variable and modifying the properties in the resulting dialog.

**Note:** When specific rules are selected, the output tree changes to show only the variable or condition associated with the rule. This is done to simplify the display for those rules. Creating a new rule restores the full output tree.














**Note:** Static fields are not displayed in the JCD output tree.

### Mapping Between Repeating Nodes


When graphically creating a link between source and destination OTD nodes, index values are automatically generated as if the trees are symmetrical (that is, the source and destination trees contain the same number of repeating nodes). If the trees are asymmetrical (that is, if the source and destination trees contain a different number of repeating nodes), the automatically-generated index values may not be appropriate to the specific situation, and you will need to modify them manually.

## Business Rules Designer Toolbar Icons

**Table 54** Business Rules Designer Toolbar Icons

Icon	Command	Function
	Class Browser	Invokes the Class Browser (see <a href="#">Using the Class Browser</a> on page 282).
	New Array	Presents a dialog with which you can add an array to the Collaboration Definition (see <a href="#">Creating Arrays</a> on page 338).
	Encoding Converter	Presents the <i>Encoding Converter Methods</i> dialog (see <a href="#">Defining Encoding Converter Methods</a> on page 362). Icon is shown only if you enable the extended language options (see <a href="#">Language</a> on page 54).
	Auto Layout	Updates the layout of fields, literals, and methods that you have added to the middle column of the Business Rules Designer and vertically aligns method boxes.
	Expand All Methods	Displays the title and contents of fields, literals, and methods.
	Collapse All Methods	Displays the title of fields, literals, and methods only.
	Boolean	Displays the <i>Boolean</i> methods menu.
	Comparison	Displays the <i>Comparison</i> methods menu.
	Math	Displays the <i>Math</i> methods menu.
	Object	Displays the <i>Object</i> methods menu.
	String	Displays the <i>String</i> methods menu.
	Array	Displays the <i>Array</i> methods menu.
	Operator	Displays the <i>Operator</i> methods menu.

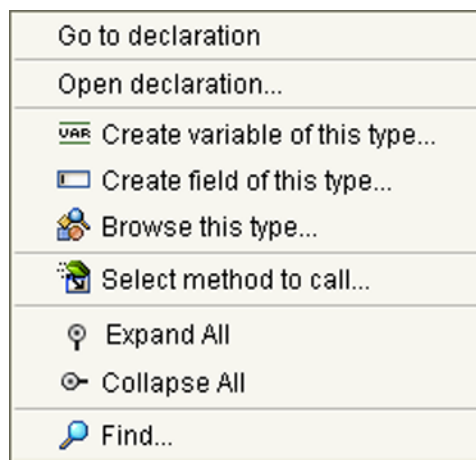
**Table 54** Business Rules Designer Toolbar Icons

Icon	Command	Function
	Assignment	Displays the <i>Assignment</i> methods menu.

## Node Menu

Right-clicking a node in the source or destination tree displays the menu shown in Figure 201, which duplicates several toolbar commands for the selected node.

**Figure 201** Node Menu



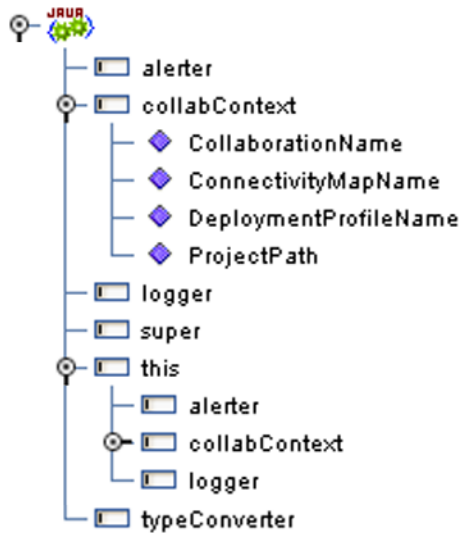
**Table 55** Business Rules Designer Node Menu Options

Option	Description
Go to Declaration	Takes you to the declaration in the Business Rules editor.
Open Declaration	Opens a dialog showing the parameters for the selected node.
Create variable of this type	Presents the <i>Create Variable</i> dialog (see <a href="#">Creating and Modifying Variables</a> on page 340).
Create field of this type	Presents the <i>Create Field</i> dialog (see <a href="#">Creating Fields</a> on page 336).
Browse this type	Presents the <i>Class Browser</i> dialog (see <a href="#">Using the Class Browser</a> on page 282).
Select method to call	Presents the method dialog, listing methods appropriate to the selected node (see <a href="#">Using the Method Dialog</a> on page 287).
Expand All	Expands the sub-tree under the selected node.
Collapse All	Collapses the sub-tree under the selected node.
Find	Presents a dialog with which you can perform a text search within the OTD tree.

## Method-Access Nodes

When you create a Java-based Collaboration, several method-access nodes are added automatically in the Business Rules Designer (see Figure 202).

**Figure 202** Method-Access Nodes



To call one of these methods, right-click on the node to display the node context menu and click **Select a method to call** (see previous section).

**Table 56** Method Access Nodes

Node	Subnode	Description
alerter		Allows access to <b>alert</b> methods, as described in <a href="#">Generating Alerts</a> on page 345.
collabContext	CollaborationName	Returns the name of the selected Collaboration, as shown in the Connectivity Map.
	ConnectivityMapName	Returns the name of the Connectivity Map in which the selected Collaboration appears.
	DeploymentProfileName	Returns the name of the Deployment in which the selected Collaboration appears.
	ProjectPath	Returns the name of the Project in which the selected Collaboration appears.
logger		Allows access to <b>logging</b> methods, as described in <a href="#">Creating Log Entries</a> on page 349.
super		Allows you to assign the class to another Java class (superclass).
this		Allows you to call other methods defined for <i>this</i> Collaboration without having to edit the Java code. Subnodes duplicate the nodes described above.

**Table 56** Method Access Nodes

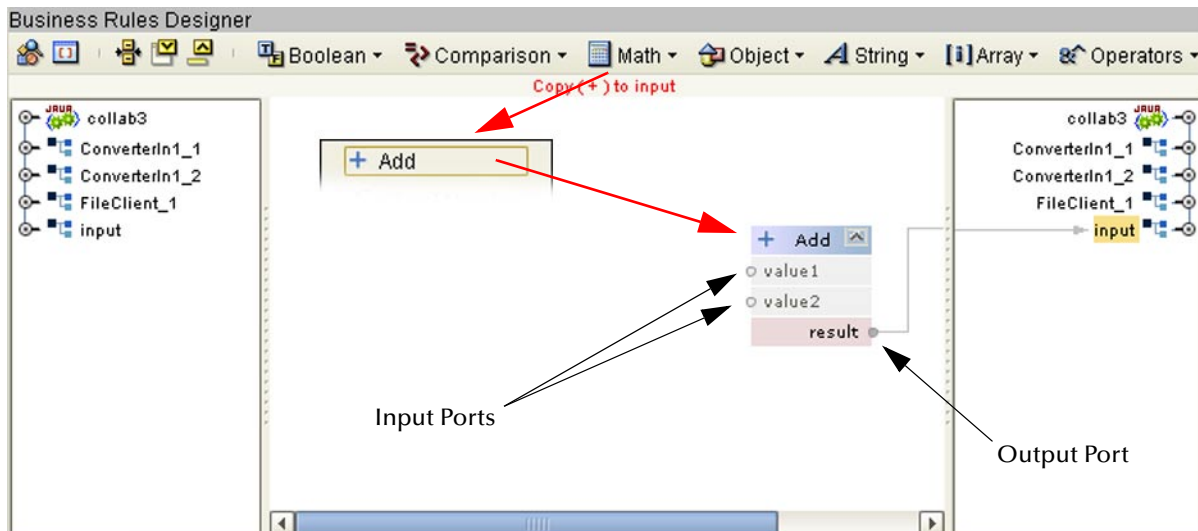
Node	Subnode	Description
typeConverter		Allows access to <b>datatype conversion</b> methods, as described in <a href="#">Auto Type Conversion</a> on page 290.

## Collaboration Method Menus

The Collaboration Method menus list commonly-used methods for use in Java-based Collaboration Definitions. Using these methods, you can create your Collaboration Definition graphically, without having to enter Java code (see [Graphical Collaboration Methods \(Java\)](#) on page 263). Clicking a category in the toolbar displays a list of methods for that category. Clicking an individual method in the list places the corresponding, predefined method box onto the Business Rules Designer mapping canvas (for alternatives, see [Placing Collaboration Methods](#) on page 262).

As an example, clicking **Math** in the toolbar displays a menu listing mathematical methods. Clicking **Add** in the menu places an **Add** method box on the mapping canvas. The method box contains input and output ports which you connect to the appropriate nodes in the OTD tree structures by dragging the cursor.

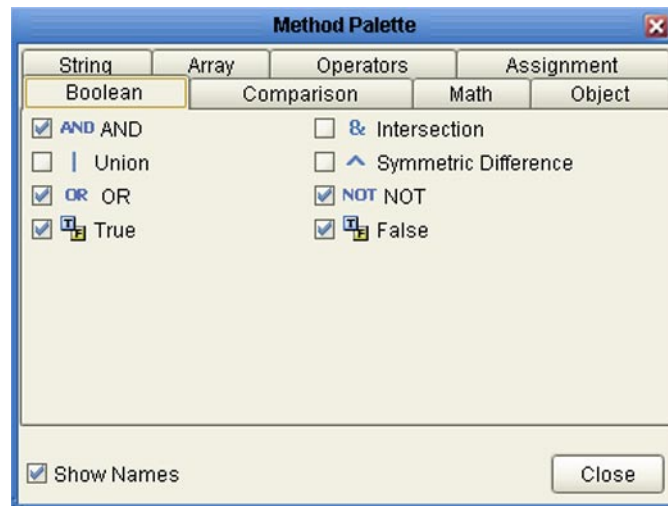
**Figure 203** Business Rules Designer: Method Menus and Boxes



Clicking the **Settings** option in any menu displays the Method Palette dialog shown in Figure 204, which allows you to select the methods that appear in the drop-down menus. Select a check box to add the method to the menu; clear a check box to remove the method from the menu.



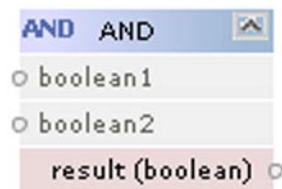
**Figure 204** Collaboration Method Palette (Java)



## Collaboration Method Boxes

The method boxes are placed on the mapping canvas of the Business Rules Designer by clicking the method in the drop-down method menu. As shown in [Figure 203 on page 260](#), the method boxes typically have input and output ports that you link to fields in the left and right panels, respectively. The method boxes are expanded by default (see Figure 205); you can collapse them (see Figure 206) by clicking the caret (^) in the upper right corner of the box. Clicking the now-inverted caret expands the box. Some boxes (such as addition, multiplication, and concatenation) expand further as needed to provide additional arguments.

**Figure 205** Expanded Method Box



**Figure 206** Collapsed Method Box



**Note:** *Tooltips, derived from the associated Javadocs, appear when you hover over a field in a method box.*

## Placing Collaboration Methods

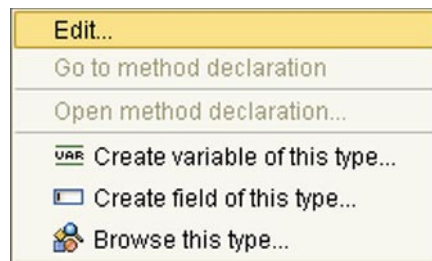
Collaboration Method Boxes are placed on the mapping canvas in any of three ways — pick the one that best suits the occasion and your operating style:

- Right-click a node in either the source or destination tree to display the node menu and click **Select a method to call** (see [Node Menu](#) on page 258).
- Click a node (or a method box port) and drag the cursor onto the mapping canvas. A method dialog appears when you release the cursor; selecting the appropriate method from the displayed list places the method box at that location (see [Using the Method Dialog](#) on page 287).
- Click a method category in the toolbar to display the corresponding menu, and click the desired option (see [Collaboration Method Menus](#) on page 260). The menus access a subset of all available methods.

When input and output values are directly mapped in the Business Rules Designer and the destination node is repeating or optional, the repeating/optional element in which the field is set is specified by an undefined index. This allows you to set a field in a specific element by defining the index.

## Editing or Adding Literals

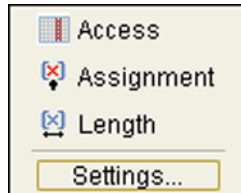
**Figure 207** Method Box Context Menu - Literal



## 9.4 Graphical Collaboration Methods (Java)

### 9.4.1 Array Methods

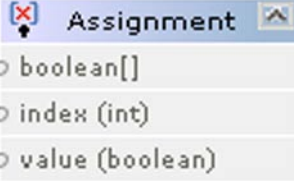
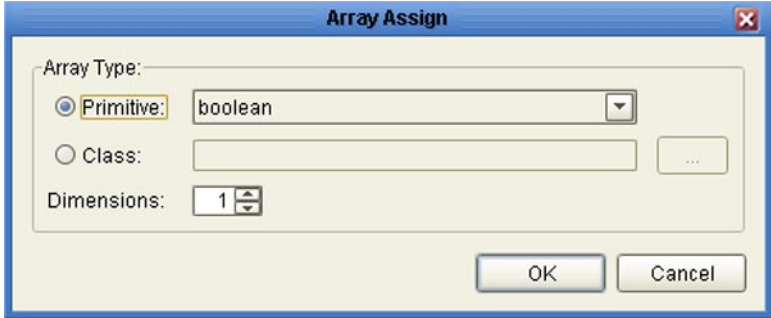
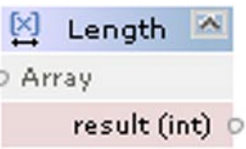
**Figure 208** Array Methods Menu



**Table 57** Array Operation Collaboration Methods (Java)

Method Box	Description/Usage
	<p>Selecting <b>Access</b> first presents the <i>Array Access</i> dialog, in which you specify the desired array type (can be either <i>primitive</i> or <i>class</i>) and dimensions. Clicking <b>OK</b> places the <b>Access</b> method box on the mapping canvas. At run time, the <b>Access</b> method returns the element located at the specified index within the array, where the index represents the number of elements from the beginning of the array.</p>

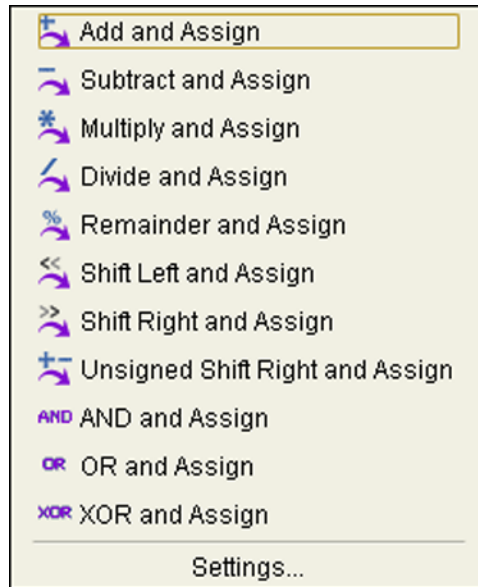
**Table 57** Array Operation Collaboration Methods (Java)

Method Box	Description/Usage
 <p>The <b>Assignment</b> method box is shown with a red 'X' icon and a blue 'A' icon. It contains three options: <code>boolean[]</code>, <code>index (int)</code>, and <code>value (boolean)</code>.</p>	<p>Selecting <b>Assignment</b> first presents the <i>Array Assign</i> dialog, in which you specify the desired array type (can be either <i>primitive</i> or <i>class</i>) and dimension. Clicking <b>OK</b> places the <b>Assignment</b> method box on the mapping canvas. At run time, the <b>Assignment</b> method sets the element located at the specified index within the array, where the index represents the number of elements from the beginning of the array.</p>  <p>The <i>Array Assign</i> dialog box has a title bar with a red 'X' icon. It contains the following fields:         <ul style="list-style-type: none"> <li><b>Array Type:</b> A radio button labeled <b>Primitive:</b> is selected, followed by a dropdown menu showing <code>boolean</code>.</li> <li>A radio button labeled <b>Class:</b> is unselected, followed by an empty text field and a button with three dots.</li> <li><b>Dimensions:</b> A text field containing the number <code>1</code> and a spinner control.</li> </ul>         At the bottom right are <b>OK</b> and <b>Cancel</b> buttons.       </p>
 <p>The <b>Length</b> method box is shown with a blue 'L' icon and a blue 'A' icon. It contains two options: <code>Array</code> and <code>result (int)</code>.</p>	<p>At run time, the <b>Length</b> method returns the length (number of elements) of the array.</p>

## 9.4.2 Assignment

The Assignment methods perform the indicated operation and then assign the result back into the variable attribute.

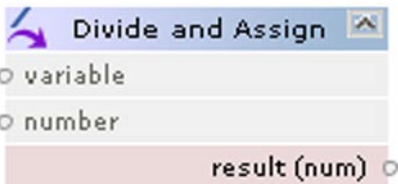
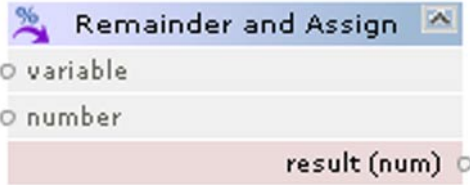
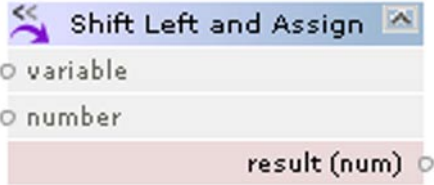
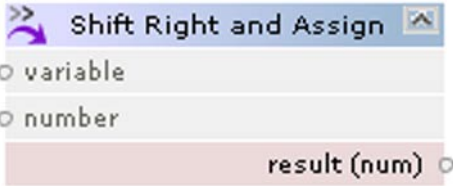
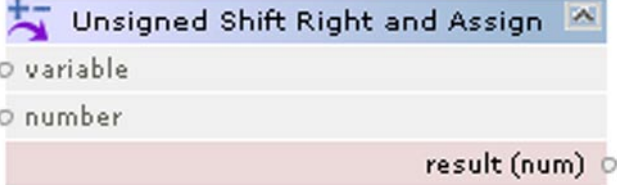
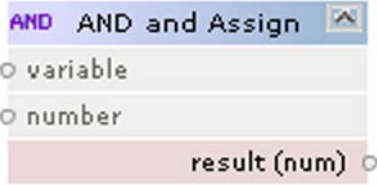
**Figure 209** Assignment Menu



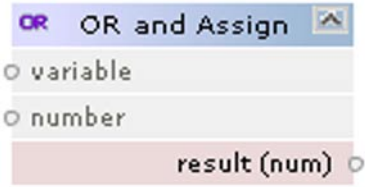
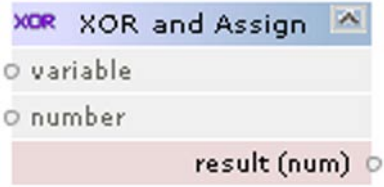
**Table 58** Assignment (Java)

Method Box	Description/Usage
	At run time, the <b>Add and Assign</b> method adds a value to a numerical or string variable, and then assigns the sum back into the variable.
	At run time, the <b>Subtract and Assign</b> method subtracts the numerical value of <i>number</i> from the numerical value of <i>variable</i> , and then assigns the difference back into <i>variable</i> .
	At run time, the <b>Multiply and Assign</b> method multiplies the numerical value of <i>number</i> and the numerical value of <i>variable</i> , and then assigns the product back into <i>variable</i> .

**Table 58** Assignment (Java)

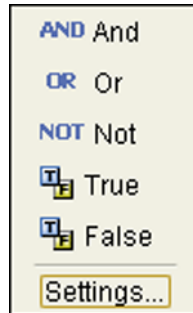
Method Box	Description/Usage
 <p><b>Divide and Assign</b></p> <ul style="list-style-type: none"> <li>variable</li> <li>number</li> <li>result (num)</li> </ul>	At run time, the <b>Divide and Assign</b> method divides the numerical value of <i>variable</i> by the numerical value of <i>number</i> , and then assigns the quotient back into <i>variable</i> .
 <p><b>Remainder and Assign</b></p> <ul style="list-style-type: none"> <li>variable</li> <li>number</li> <li>result (num)</li> </ul>	At run time, the <b>Remainder and Assign</b> method divides the numerical value of <i>variable</i> by the numerical value of <i>number</i> , and then assigns the remainder back into <i>variable</i> .
 <p><b>Shift Left and Assign</b></p> <ul style="list-style-type: none"> <li>variable</li> <li>number</li> <li>result (num)</li> </ul>	At run time, the <b>Shift Left and Assign</b> method shifts the value of <i>variable</i> to the left by <i>number</i> of places, and then assigns the result back into <i>variable</i> . (This operation is equivalent to multiplication by power of two.)
 <p><b>Shift Right and Assign</b></p> <ul style="list-style-type: none"> <li>variable</li> <li>number</li> <li>result (num)</li> </ul>	At run time, the <b>Shift Right and Assign</b> method shifts the value of <i>variable</i> to the right by <i>number</i> of places, and then assigns the result back into <i>variable</i> . New left bits are filled with the value of the high-order bit so that the algebraic sign is retained.
 <p><b>Unsigned Shift Right and Assign</b></p> <ul style="list-style-type: none"> <li>variable</li> <li>number</li> <li>result (num)</li> </ul>	At run time, the <b>Unsigned Shift Right and Assign</b> method shifts the value of <i>variable</i> to the right by <i>number</i> of places, and then assigns the result back into <i>variable</i> . New left bits are filled with zeros. (For non-negative integers, an unsigned right shift is equivalent to dividing the left operator by the number two raised to the power of the right operator.)
 <p><b>AND and Assign</b></p> <ul style="list-style-type: none"> <li>variable</li> <li>number</li> <li>result (num)</li> </ul>	At run time, the <b>AND and Assign</b> method performs a bitwise AND between <i>variable</i> and <i>number</i> , and then assigns the result back into <i>variable</i> . This operation returns Boolean true if both operands evaluate to true, and false if both operands evaluate to false.

**Table 58** Assignment (Java)

Method Box	Description/Usage
	<p>At run time, the <b>OR and Assign</b> method performs a bitwise OR between <i>variable</i> and <i>number</i>, and then assigns the result back into <i>variable</i>. This operation returns Boolean true if either or both operands evaluate to true; otherwise, returns Boolean false.</p>
	<p>At run time, the <b>XOR and Assign</b> method performs a bitwise XOR between <i>variable</i> and <i>number</i>, and then assigns the result back into <i>variable</i>. This operation returns Boolean true if both operands are different; otherwise, returns Boolean false.</p>

### 9.4.3 Boolean Methods

**Figure 210** Boolean Methods Menu



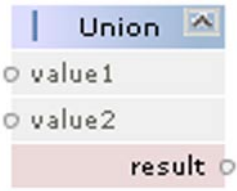
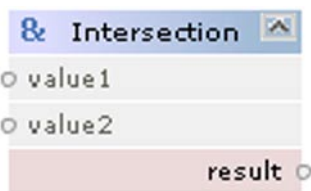
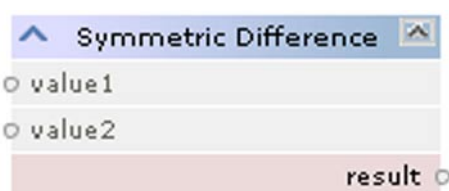
**Table 59** Boolean Collaboration Methods (Java)

Method Box	Description/Usage
	At run time, the <b>AND</b> method returns Boolean true if both <i>boolean1</i> and <i>boolean2</i> are true; otherwise, returns Boolean false.
	At run time, the <b>OR</b> method returns Boolean false if both <i>boolean1</i> and <i>boolean2</i> are false; otherwise, returns Boolean true.
	At run time, the <b>NOT</b> method returns the inverse of <i>boolean</i> .
	At run time, the <b>True</b> method always returns Boolean true.
	At run time, the <b>False</b> method always returns Boolean false.



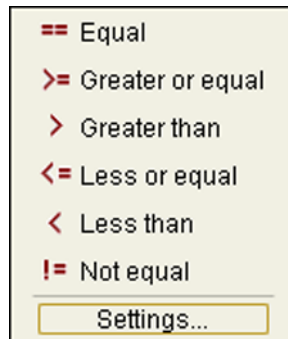
The following methods are available from both the *Boolean* menu and the *Operators* menu.

**Table 60** Additional Boolean Collaboration Methods (Java)

Method Box	Description/Usage
 <p>The Union operator method box features a blue header with the text 'Union' and a small icon. Below the header are two input fields labeled 'value1' and 'value2', each with a circular selection icon to its left. At the bottom is a red output field labeled 'result' with a circular selection icon to its right.</p>	At run time, the <b>Union</b> operator computes the union of the input values (not shown by default).
 <p>The Intersection operator method box features a blue header with the text '&amp; Intersection' and a small icon. Below the header are two input fields labeled 'value1' and 'value2', each with a circular selection icon to its left. At the bottom is a red output field labeled 'result' with a circular selection icon to its right.</p>	At run time, the <b>Intersection</b> operator computes the set intersection of bit or Boolean values (not shown by default).
 <p>The Symmetric Difference operator method box features a blue header with the text '^ Symmetric Difference' and a small icon. Below the header are two input fields labeled 'value1' and 'value2', each with a circular selection icon to its left. At the bottom is a red output field labeled 'result' with a circular selection icon to its right.</p>	At run time, the <b>Symmetric Difference</b> operator computes the set symmetric difference of bit or Boolean values (not shown by default).

## 9.4.4 Comparison Methods

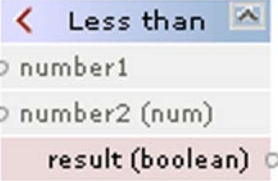
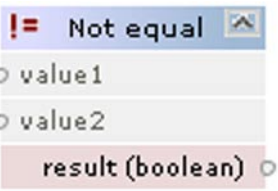
**Figure 211** Comparison Methods Menu



**Table 61** Comparison Collaboration Methods (Java)

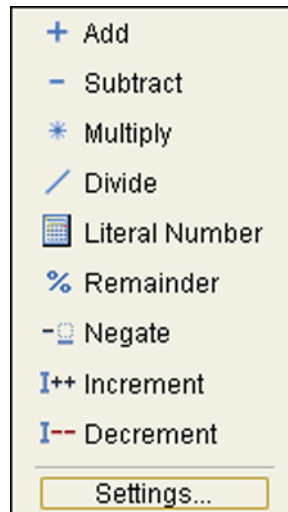
Method Box	Description/Usage
	At run time, the <b>Equal</b> method returns Boolean true if the value of <i>number1</i> is equal to the value of <i>number2</i> ; otherwise, returns Boolean false.
	At run time, the <b>Greater or equal</b> method returns Boolean true if the value of <i>number1</i> is greater than or equal to the value of <i>number2</i> ; otherwise, returns Boolean false.
	At run time, the <b>Greater than</b> method returns Boolean true if the value of <i>number1</i> is greater than the value of <i>number2</i> ; otherwise, returns Boolean false.
	At run time, the <b>Less or equal</b> method returns Boolean true if the value of <i>number1</i> is less than or equal to the value of <i>number2</i> ; otherwise, returns Boolean false.

**Table 61** Comparison Collaboration Methods (Java)

Method Box	Description/Usage
 <p>The 'Less than' method box features a blue header with a red less-than sign icon and the text 'Less than' followed by a small icon. Below the header are two input fields: 'number1' and 'number2 (num)'. At the bottom is a red output field labeled 'result (boolean)' with a small icon to its right.</p>	<p>At run time, the <b>Less than</b> method returns Boolean true if the value of <i>number1</i> is less than the value of <i>number2</i>; otherwise, returns Boolean false.</p>
 <p>The 'Not equal' method box features a blue header with a red '!= ' icon and the text 'Not equal' followed by a small icon. Below the header are two input fields: 'value1' and 'value2'. At the bottom is a red output field labeled 'result (boolean)' with a small icon to its right.</p>	<p>At run time, the <b>Not equal</b> method returns Boolean true if the value of <i>number1</i> is not equal to the value of <i>number2</i>; otherwise, returns Boolean false.</p>

## 9.4.5 Math Methods

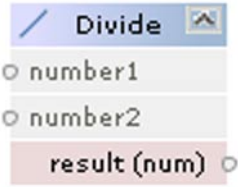
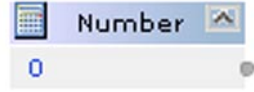
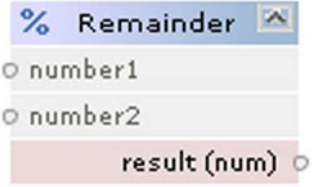
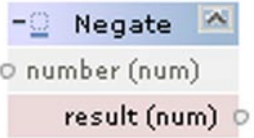
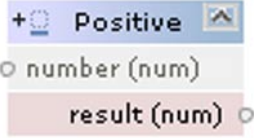
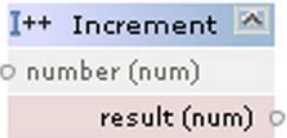
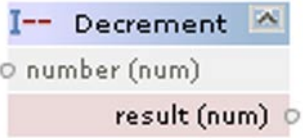

**Figure 212** Math Methods Menu



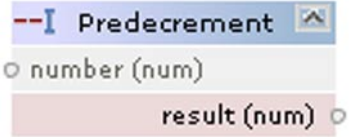
**Table 62** Math Collaboration Methods (Java)

Method Box	Description/Usage
<p>The Add method box has a blue header with a plus sign and the text "Add". Below the header are two input fields labeled "value1" and "value2". At the bottom is a red output field labeled "result".</p>	At run time, the <b>Add</b> method returns the sum of <i>value1</i> and <i>value2</i> .
<p>The Subtract method box has a blue header with a minus sign and the text "Subtract". Below the header are two input fields labeled "number1" and "number2". At the bottom is a red output field labeled "result (num)".</p>	At run time, the <b>Subtract</b> method subtracts the numerical value of <i>number2</i> from the numerical value of <i>number1</i> , returns the difference.
<p>The Multiply method box has a blue header with an asterisk and the text "Multiply". Below the header are two input fields labeled "number1" and "number2". At the bottom is a red output field labeled "result (num)".</p>	At run time, the <b>Multiply</b> method returns the product of the numerical value of <i>number1</i> and the numerical value of <i>number2</i> .

**Table 62** Math Collaboration Methods (Java)

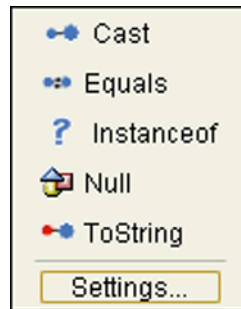
Method Box	Description/Usage
	At run time, the <b>Divide</b> method returns the quotient of the numerical value of <i>number1</i> divided by the numerical value of <i>number2</i> .
	At run time, the literal <b>Number</b> method returns the specified numeric value. Double-clicking in the value field enables the field for editing. The default value is zero ( <b>0</b> ).
	At run time, the <b>Remainder</b> method divides the numerical value of <i>number1</i> by the numerical value of <i>number2</i> , returns the remainder.
	At run time, the <b>Negate</b> method returns the input number as a negative value.
	The <b>Positive</b> method returns the input number as a positive value. By default, this method does not appear in the menu — you can enable it from the method palette
	At run time, the <b>Increment</b> method increments the input number by one, following other operations.
	At run time, the <b>Decrement</b> method decrements the input number by one, following other operations.
	At run time, the <b>Preincrement</b> method increments the input number by one, preceding other operations. By default, this method does not appear in the menu — you can enable it from the method palette

**Table 62** Math Collaboration Methods (Java)

Method Box	Description/Usage
	<p>At run time, the <b>Predecrement</b> method decrements the input number by one, preceding other operations. By default, this method does not appear in the menu — you can enable it from the method palette.</p>

## 9.4.6 Object Methods


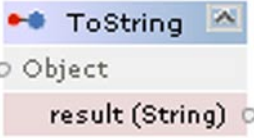
**Figure 213** Object Methods Menu



**Table 63** Object Collaboration Methods (Java)

Method Box	Description/Usage
<p>The image shows a method box for 'Cast (boolean)'. It has a blue header with the text 'Cast (boolean)' and a small icon. Below the header, there is a list of inputs: 'any' (selected) and 'result (boolean)' (output).</p>	<p>Selecting <b>Cast</b> first presents the <i>Cast</i> dialog, in which you specify the desired type (can be either <i>primitive</i> or <i>class</i>) and dimension (if it is an array). Clicking <b>OK</b> places the <b>Cast</b> method box on the mapping canvas. At run time, the <b>Cast</b> method converts the given type (<i>any</i>) to the type specified in the dialog (<b>boolean</b> is shown as an example).</p> <p>The image shows a dialog box titled 'Cast'. It has a 'Cast To Type' section with two radio buttons: 'Primitive' (selected) and 'Class'. The 'Primitive' option has a dropdown menu showing 'boolean'. There is also a 'Is Array' checkbox and an 'Array Dimensions' spinner set to '1'. At the bottom are 'OK' and 'Cancel' buttons.</p>
<p>The image shows a method box for 'Equals'. It has a blue header with the text 'Equals' and a small icon. Below the header, there is a list of inputs: 'Object' (selected), 'obj (Object)', and 'result (boolean)' (output).</p>	<p>At run time, the <b>Equals</b> method returns Boolean true if <i>Object</i> is logically equal to <i>obj</i>, otherwise, returns Boolean false.</p>
<p>The image shows a method box for 'instanceof Date'. It has a blue header with the text 'instanceof Date' and a small icon. Below the header, there is a list of inputs: 'Object' (selected) and 'result (boolean)' (output).</p>	<p>Selecting <b>Instanceof</b> first presents the <i>Class Browser</i> dialog, in which you specify the desired class (see <a href="#">Using the Class Browser</a> on page 282). Clicking <b>OK</b> places the <b>instanceof</b> method box on the mapping canvas. At run time, the <b>Instanceof</b> method returns Boolean true if <i>Object</i> is of the specified type; otherwise, returns Boolean false. <b>Date</b> is shown as an example.</p>

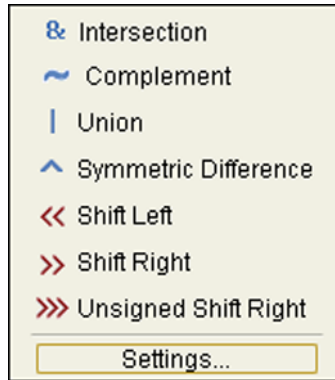
**Table 63** Object Collaboration Methods (Java)

Method Box	Description/Usage
 <p>The image shows a method box for the <code>Object</code> class. It has a blue header with the text "Object" and a small icon on the left. Below the header, the word "null" is displayed in a light blue box.</p>	At run time, the <b>Null</b> method always returns a null value.
 <p>The image shows a method box for the <code>Object</code> class. It has a blue header with the text "ToString" and a small icon on the left. Below the header, the text "Object" is displayed in a light blue box. At the bottom, the text "result (String)" is displayed in a light red box.</p>	At run time, the <b>ToString</b> method converts <i>Object</i> into a string.



## 9.4.7 Operators

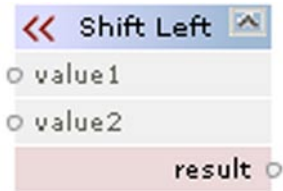
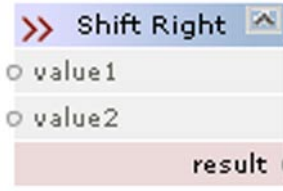
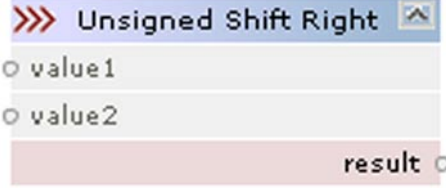
**Figure 214** Operators Menu



**Table 64** Operators (Java)

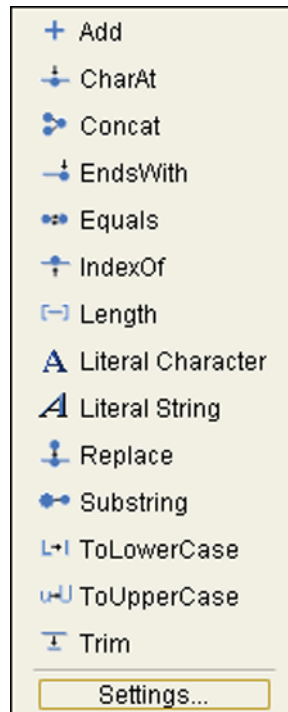
Operator Box	Description/Usage
<p>The box shows the 'Intersection' operator with inputs 'value1' and 'value2', and an output 'result'.</p>	At run time, the <b>Intersection</b> operator computes the set intersection of bit or Boolean values. (Also listed on the Boolean menu, but not shown by default.)
<p>The box shows the 'Complement' operator with input 'number (num)' and an output 'result (num)'.</p>	At run time, the <b>Complement</b> operator computes the complement of a bit value by inverting each bit of the input number.
<p>The box shows the 'Union' operator with inputs 'value1' and 'value2', and an output 'result'.</p>	At run time, the <b>Union</b> operator computes the union of the input values. (Also listed on the Boolean menu, but not shown by default.)
<p>The box shows the 'Symmetric Difference' operator with inputs 'value1' and 'value2', and an output 'result'.</p>	At run time, the <b>Symmetric Difference</b> operator computes the set symmetric difference of bit or Boolean values. (Also listed on the Boolean menu, but not shown by default.)

**Table 64** Operators (Java)

Operator Box	Description/Usage
	At run time, the <b>Shift Left</b> operator shifts bits to the left; equivalent to multiplication by power of two.
	At run time, the <b>Shift Right</b> operator shifts bits to the right; new left bits are filled with the value of the high-order bit so the sign is retained.
	At run time, the <b>Unsigned Shift Right</b> operator shifts bits to the right, based on the value of the right operand; new left bits are filled with zeros. For non-negative integers, an unsigned right shift is equivalent to dividing the left operator by the number two raised to the power of the right operator.

## 9.4.8 String Methods


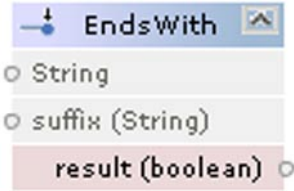
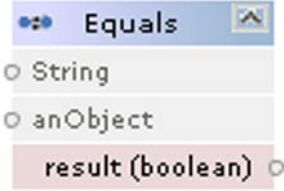
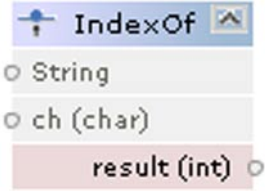


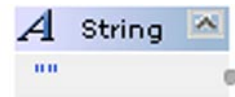
**Figure 215** String Methods Menu



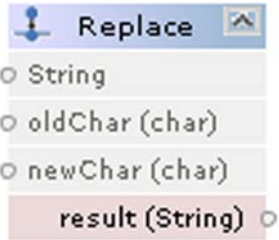
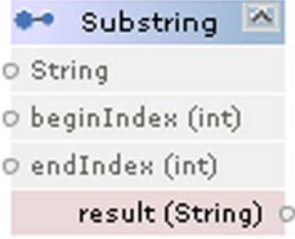
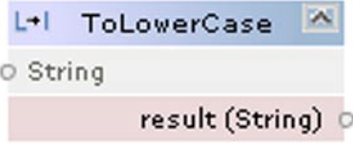
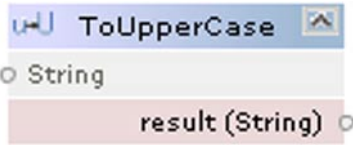

**Table 65** String Collaboration Methods (Java)

Method Box	Description/Usage
	At run time, if <i>value1</i> is a string, the <b>Add</b> method converts <i>value2</i> to a string and concatenates it to <i>value1</i> to produce the <i>result</i> .
	At run time, the <b>CharAt</b> method returns the character at the specified index, where <i>index</i> is the number of characters from the beginning of <i>String</i> .

**Table 65** String Collaboration Methods (Java)

Method Box	Description/Usage
	At run time, the <b>Concat</b> method returns the string created by concatenating <i>str</i> to the end of <i>String</i> .
	At run time, the <b>EndsWith</b> method returns Boolean true if <i>String</i> ends with the string <i>suffix</i> ; otherwise, returns Boolean false.
	At run time, the <b>Equals</b> method compares <i>String</i> with <i>anObject</i> and returns Boolean true if and only if they both represent the same sequence of characters and the argument is not null. Otherwise, returns Boolean false.
	At run time, the <b>IndexOf</b> method returns the index within <i>String</i> corresponding to the location of the specified character ( <i>ch</i> ), where the index represents the number of characters from the beginning of <i>String</i> .
	At run time, the <b>Length</b> method returns the length (number of characters) of <i>String</i> .
	At run time, the <b>Literal Character</b> method returns the specified character. Double-clicking in the value field enables the field for editing. The default value is a space character (' '), as shown.
	At run time, the <b>Literal String</b> method returns the specified string. Double-clicking in the value field enables the field for editing. The default value is a null string(''), as shown. <b>Note:</b> Use <i>Ctrl+Enter</i> to insert a line break.

**Table 65** String Collaboration Methods (Java)

Method Box	Description/Usage
 <p>Replace</p> <ul style="list-style-type: none"> <li>String</li> <li>oldChar (char)</li> <li>newChar (char)</li> <li>result (String)</li> </ul>	At run time, the <b>Replace</b> method returns a new string in which all occurrences of <i>oldChar</i> are replaced with <i>newChar</i> .
 <p>Substring</p> <ul style="list-style-type: none"> <li>String</li> <li>beginIndex (int)</li> <li>endIndex (int)</li> <li>result (String)</li> </ul>	At run time, the <b>Substring</b> method returns a string that is a substring of <i>String</i> , beginning from <i>beginIndex</i> (inclusive) and ending at <i>endIndex</i> (exclusive). The indices represent the number of characters from the beginning of <i>String</i> .
 <p>ToLowerCase</p> <ul style="list-style-type: none"> <li>String</li> <li>result (String)</li> </ul>	At run time, the <b>ToLowerCase</b> method converts all characters in <i>String</i> to lower case, using the rules of the default locale.
 <p>ToUpperCase</p> <ul style="list-style-type: none"> <li>String</li> <li>result (String)</li> </ul>	At run time, the <b>ToUpperCase</b> method converts all characters in <i>String</i> to upper case, using the rules of the default locale.
 <p>Trim</p> <ul style="list-style-type: none"> <li>String</li> <li>result (String)</li> </ul>	At run time, the <b>Trim</b> method trims leading and trailing whitespace from <i>String</i> .

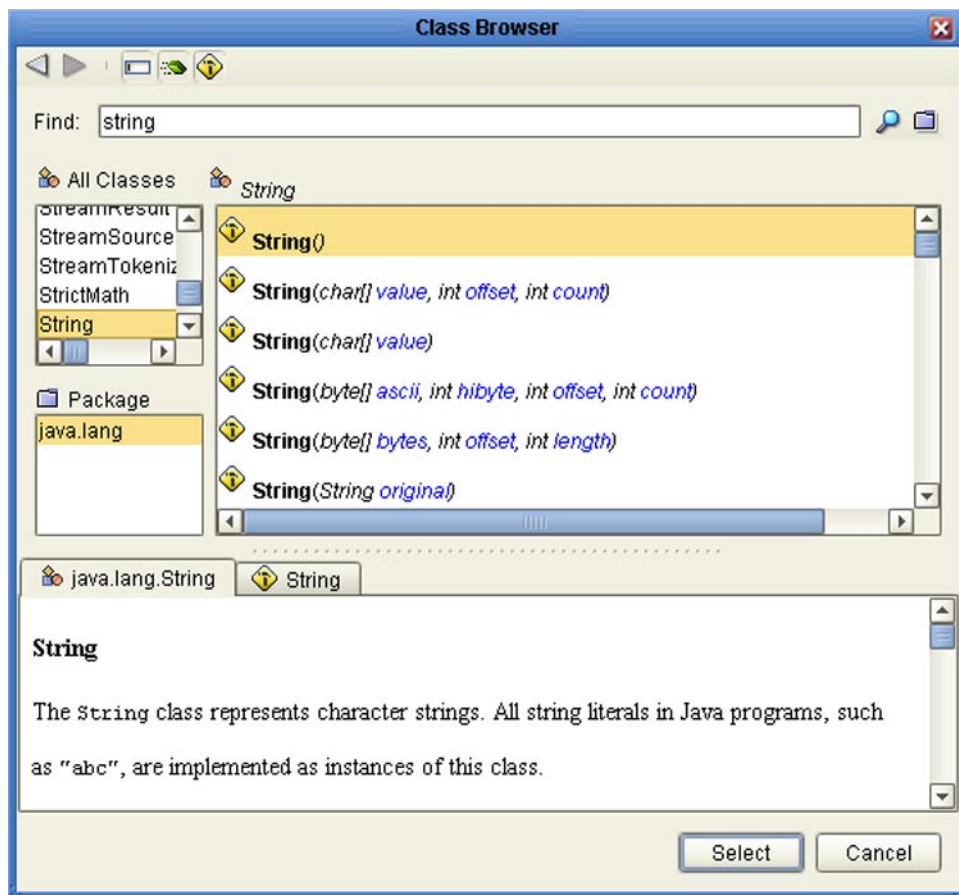
## 9.5 Using the Class Browser

Clicking the Class Browser icon in the Business Rules Designer toolbar invokes the Class Browser, which you can use to search for Java classes and methods. The browser displays all available classes, including any third-party classes that have been uploaded.

Figure 216 Class Browser Icon










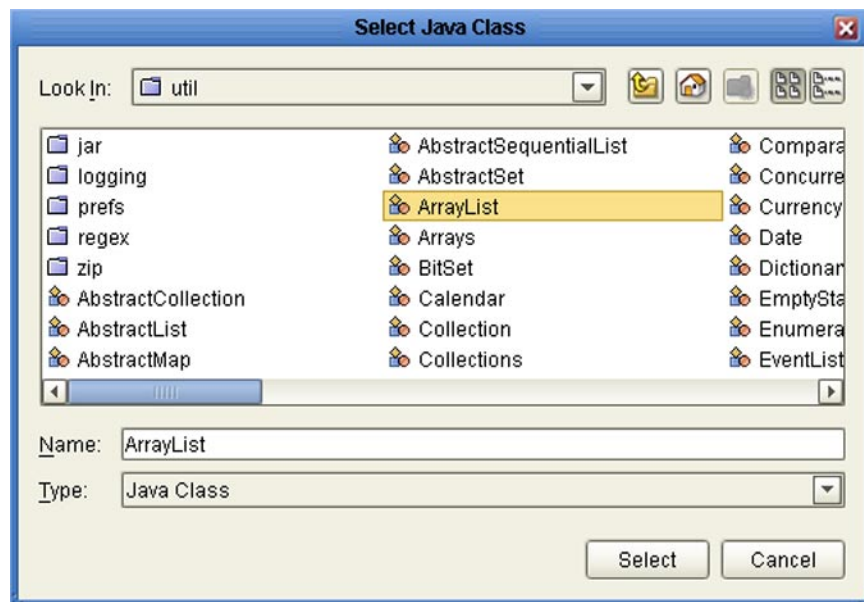
Figure 217 Class/Method Browser



## 9.5.1 Class Browser Toolbar Icons

**Table 66** Method Browser Toolbar Icons

Icon	Command	Function
	Step Back	Steps back in your usage history for the current session.
	Step Forward	Steps forward in your usage history for the current session.
	Show Fields	Displays or hides all fields in the selected class. By default, all fields are displayed; you must click the icon to remove them from the list.
	Show Methods	Displays or hides all methods in the selected class. By default, all methods are displayed; you must click the icon to remove them from the list.
	Show Constructors	Displays or hides all constructors in the selected class. By default, all constructors are displayed; you must click the icon to remove them from the list.
	Search	Executes search on an entry in <b>Find</b> text box.
	Browse Java Package Hierarchy	Presents the <i>Select Java Class</i> dialog, with which you can locate classes contained in another installed package. Clicking <b>Select</b> displays the contents of the package in the Class Browser.

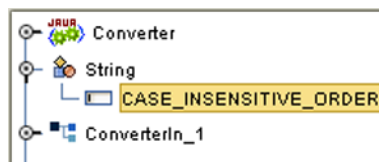


You can locate a class by scrolling through the class list or by typing the first few letters of a class name into the text box and clicking the **Search** icon. Scrolling through the method list displays all methods, constructors, and fields defined for that class. If the desired class is not listed, you can search for it in other installed packages by using the **Browse Java Package Hierarchy** icon.

The action resulting from clicking **Select** depends upon how you are using the browser.

- If you are adding a class or method instance to the Collaboration, it adds a constructor or method box to the Business Rules Designer mapping canvas (see [To add a class instance using the Class Browser](#) on page 285).
- If you are adding a field to the Collaboration, it adds the class and field to the source OTD, as shown in Figure 218.

**Figure 218** Class and Field



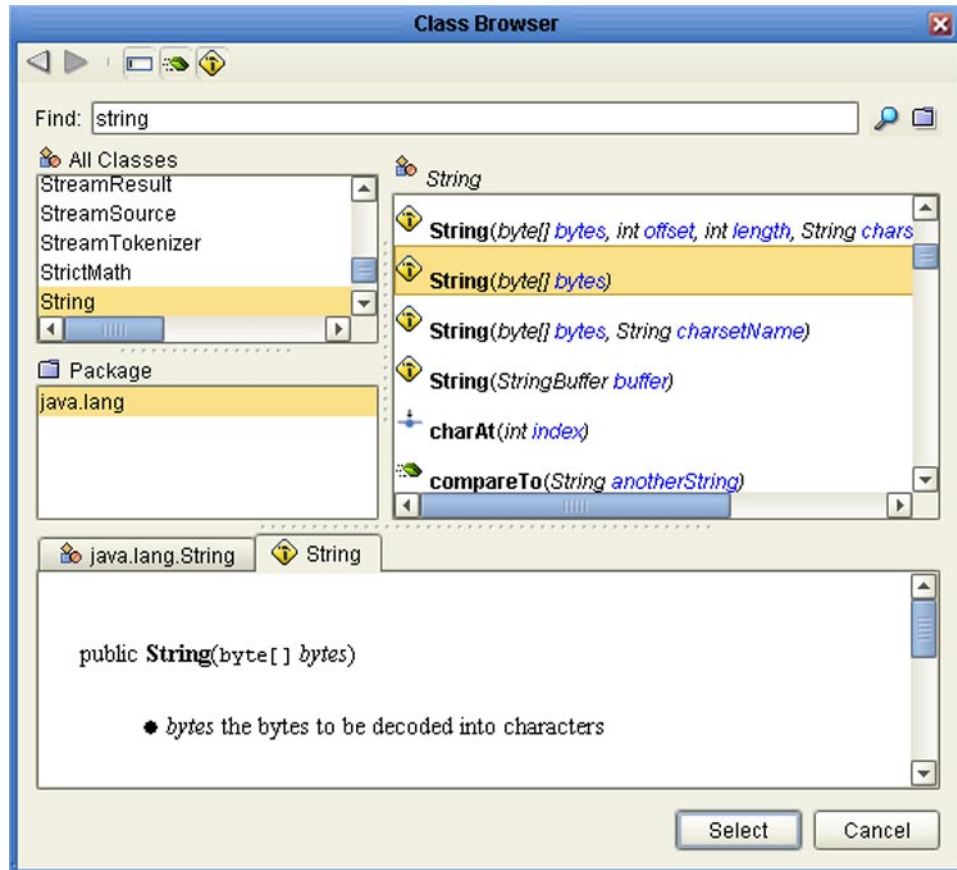
- If you are creating a variable, it adds the variable to the tree in the Business Rules editor (see [Creating and Modifying Variables](#) on page 340).



### To add a class instance using the Class Browser

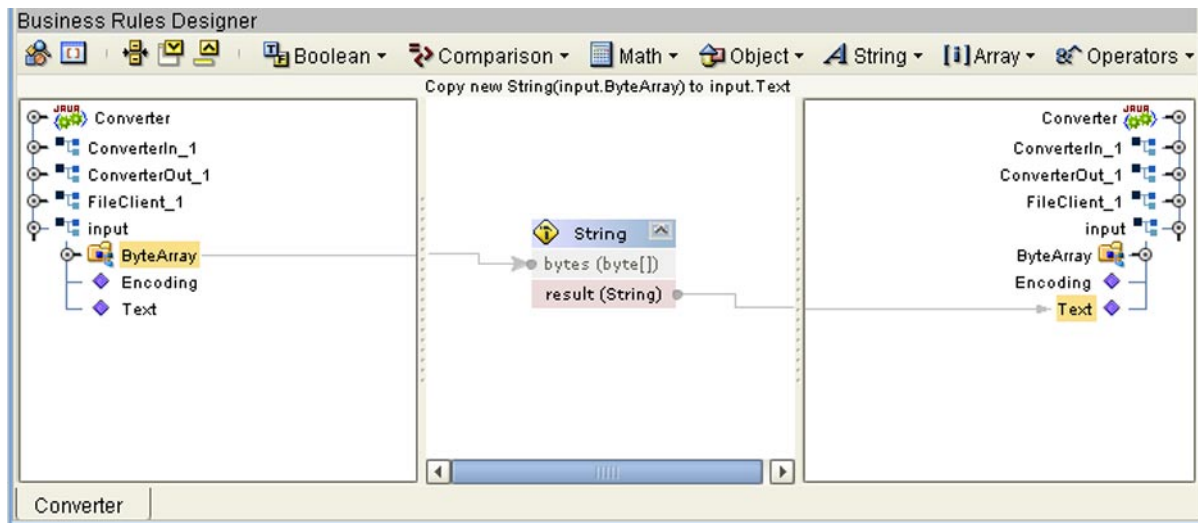
- 1 Click the **Class Browser** icon in the Business Rules Designer toolbar to display the Class Browser.
- 2 In the Class Browser, search for and select the desired class and constructor (see Figure 219).

**Figure 219** Class Browser - Select Constructor



- 3 Click **Select** to place the constructor box on the Business Rules Designer mapping canvas as shown in Figure 220.

**Figure 220** Constructor Box on Mapping Canvas



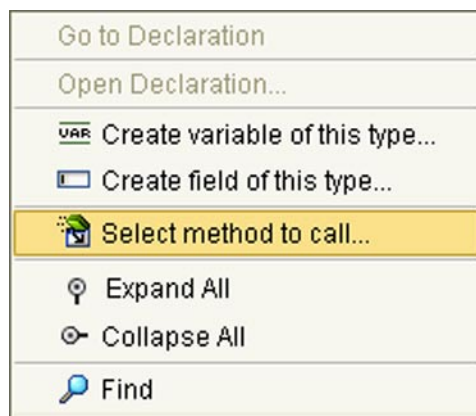
- 4 Link the constructor box ports to the appropriate OTD nodes, as shown.
- 5 **Save** to the Repository.

## 9.6 Using the Method Dialog

The method dialog provides an alternative to the Class Browser for selecting methods and constructors, and also provides for creating literals. To display the method dialog, select a node in either the left or right panel of the Business Rules Designer (or a method box port) and:

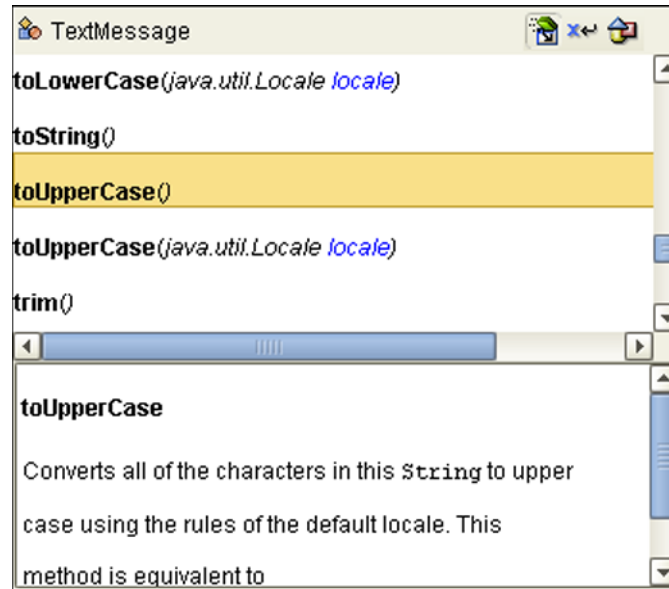
- Right-click on the node to display the context menu and click **Select a method to call** (see Figure 221), or
- Left-click on the node and drag the cursor to the mapping panel.

**Figure 221** Node Context Menu



Either action will display the method list for that specific node type, such as that shown in Figure 222 (using the second option allows you to select the position of the resulting method box, which can save time). A description of the method appears in the lower panel of the method dialog.

**Figure 222** Method Dialog (for type TextMessage)



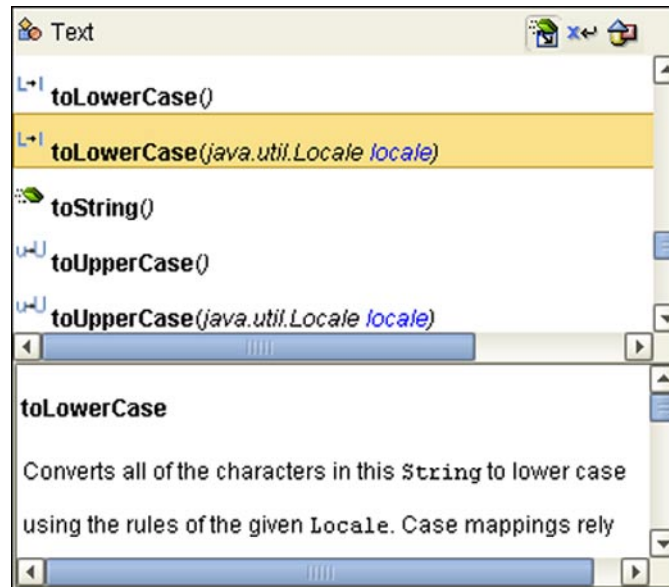
**Table 67** Method Dialog Toolbar Icons

Icon	Command	Function
	Show Methods	Displays a list of methods <b>having</b> the same type as the selected node. This is the default display when the dialog is launched from the <i>left</i> panel of the Business Rules Designer or a method box <i>output</i> port.
	Return	Displays a list of methods <b>returning</b> the same type as the selected node. This is the default display when the dialog is launched from the <i>right</i> panel of the Business Rules Designer or a method box <i>input</i> port.
OR		Creates a literal of the same type as the as the selected node (object or string), placing the literal box on the mapping canvas for linking. See <a href="#">Creating and Modifying Literals</a> on page 344.

### To add a method instance using the method dialog

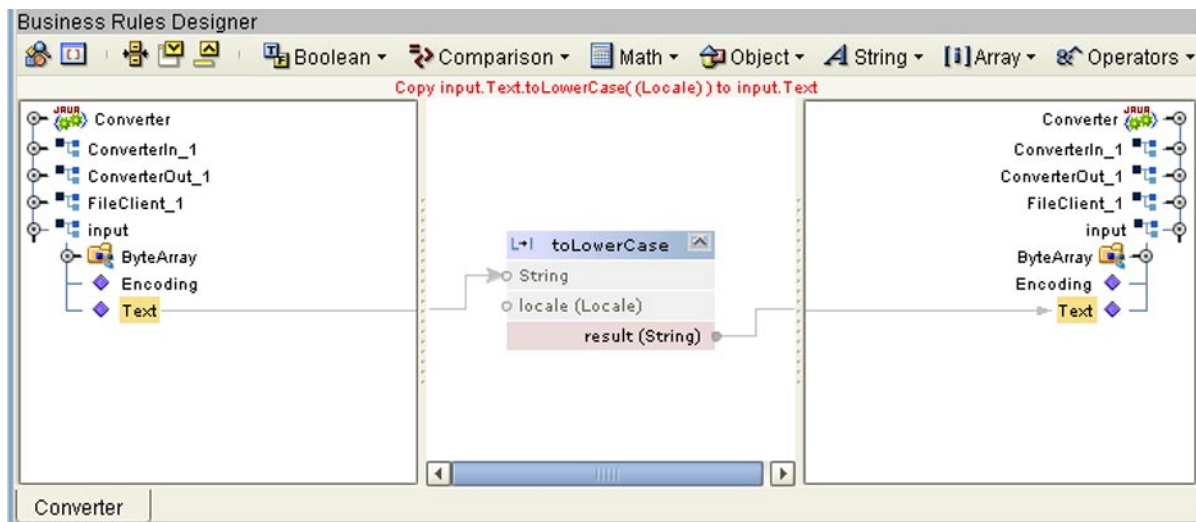
- 1 To display a list of methods of the same type as a node (see Figure 223), either right-click on the node (and release) or left-click on the node and drag to the mapping panel.

**Figure 223** Method List for Text



- 2 Scroll to the desired method and double-click on the method to place the method box in the Business Rules Designer as shown in Figure 224, then link the method ports and **Save** to the Repository.

**Figure 224** Method Box in Mapping Panel



## 9.7 Auto Type Conversion

When you map between OTD nodes having different data types, if a conversion method exists for that pair of data types, a dialog is automatically presented in which you can specify certain properties associated with the conversion. The primary content of the dialog is specific to the conversion, examples of which are described in following sections. Two properties that are common to all dialogs deal with exception handling and whether or not the dialog is displayed for each conversion.

### Exception handling

Select the method for handling exceptions, either *Use Default Value* or *Throw Exception*, from the drop-down menu (see Figure 225). In general,

- If you select **Use Default Value**, the output will be based on the unquoted string in the *Default Value* box if the desired conversion cannot be performed. See the individual conversion type descriptions for specific details.
- If you select **Throw Exception**, then the code should use a *try/catch* block to catch the exception should it occur at run-time.

**Figure 225** Exception Handling Options




### Automatic conversion

If you want your specification to be used for all subsequent conversions between the same data types, select the check box beside **Please convert automatically next time** (it is selected by default — see Figure 226). If this option is enabled, all subsequent conversions between those same data types will use the same settings, and the dialog will not be displayed.

**Note:** *The dialog will appear again the next time you open the Collaboration Definition Editor — you can reset the option at that time.*

**Figure 226** Automatic Conversion Option



If you want to specify each conversion individually, clear the check box. The dialog will then be presented for each conversion.

## Data types

Not all data types can be converted into other specific data types — conversions for which type conversion methods exist are listed in Table 68. You can also access these methods from the **typeConverter** node (see [Method-Access Nodes](#) on page 259). Attempting to map between data types for which no type conversion method exists displays an error notice.

**Table 68** Supported Datatype Conversions

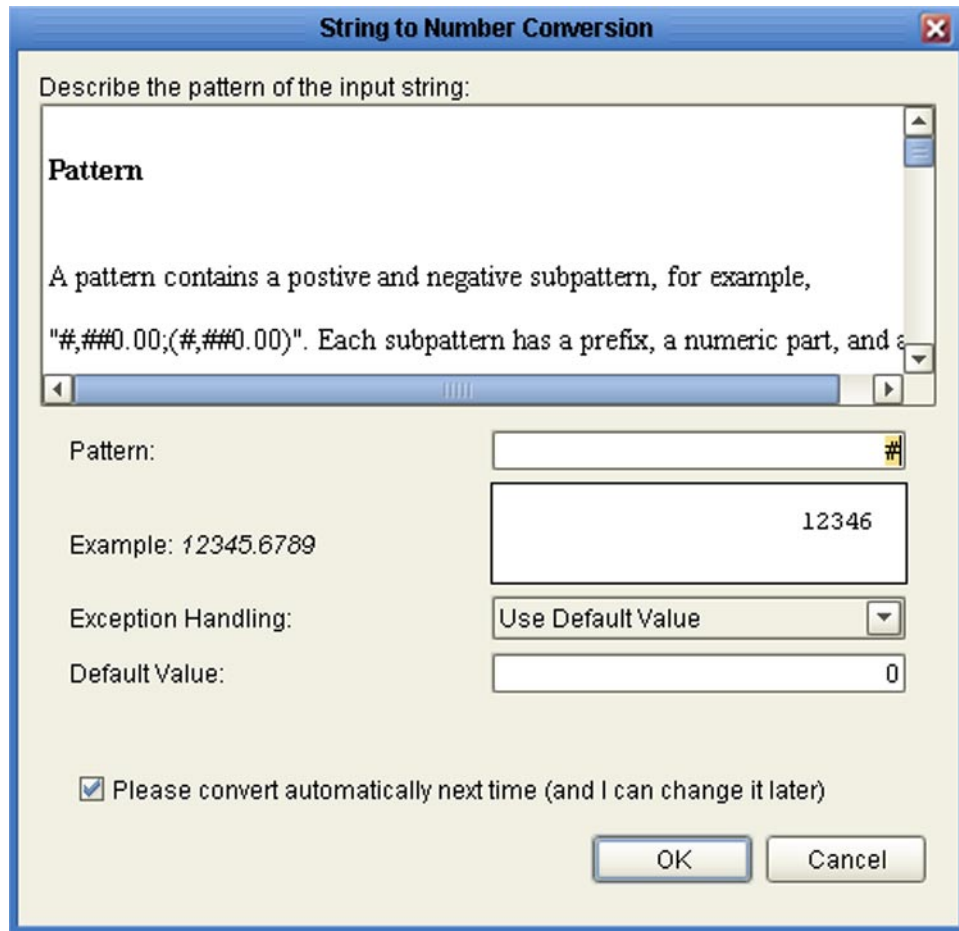
String to Byte Array	Byte Array to String
String to Date	Date to String
String to DateTime	DateTime to String
String to Double	Double to String
String to Float	Float to String
String to Int	Int to String
String to Long	Long to String
String to Short	Short to String
String to SQL Date	String to Timestamp

## 9.7.1 Character Conversion

### String to Number Conversion

Mapping between a *Text* node and an *int* node automatically presents the dialog shown in Figure 227, where you can specify the pattern of the input string.

**Figure 227** Type Conversion - String to Number



#### String Pattern

You specify the pattern by entering characters in the **Pattern** text box, and an example illustrating the resulting output appears in the field below. A comprehensive explanation of how this works is provided in the scrollable text box above. For your convenience, the following table lists special characters that are frequently used.



**Table 69** Special Characters

Symbol	Location	Description or Use	Use Quotes for Literal?
0	Number	Digit.	Yes
1-9	Number	'1' - '9' indicate rounding.	Yes
@	Number	Significant digit.	Yes
#	Number	Digit (zero shows as absent).	Yes
.	Number	Decimal separator.	Yes
-	Number	Minus sign.	Yes
,	Number	Grouping separator.	Yes
E	Number	Separates mantissa and exponent in scientific notation.	No
+	Exponent	Prefix positive exponents with localized plus sign.	Yes
;	Subpattern boundary	Separates positive and negative subpatterns.	Yes
%	Prefix or suffix	Multiply by 100 and show as percentage.	Yes
/u2030	Prefix or suffix	Multiply by 1000 and show as per mile.	Yes
'	Prefix or suffix	Used to quote special characters in a prefix or suffix to appear as literal.	See below
*	Prefix or suffix boundary	Pad escape, precedes pad character.	Yes

To be entered as literals, special characters generally must be entered within quotation marks, but there are exceptions to this rule, as noted in the table. To have a single quote (') appear as a literal (as an apostrophe, for example), precede it with another single quote.

### Exception Handling

This parameter allows you to specify that when the desired conversion cannot be performed, either:

- ♦ An exception should be thrown, or
- ♦ The output should be set to the value of the default string specified as the *Default Value* parameter.

### Default Value

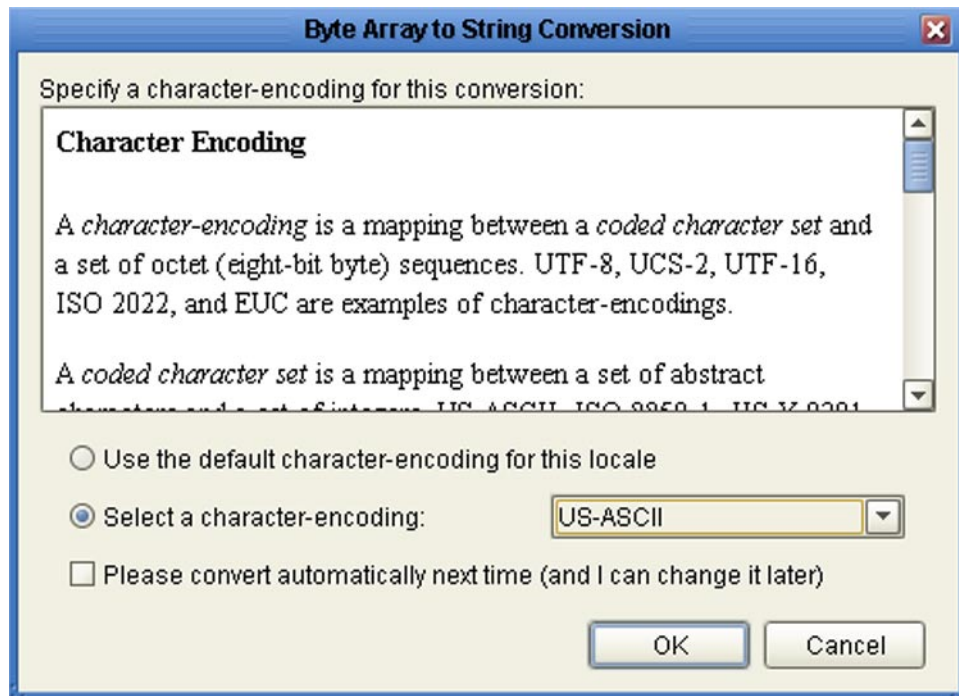
If you have selected *Default Value* for the exception handling, specify an unquoted character string. The output string will be set to this string value.

## 9.7.2 Character Encoding

Mapping between a *ByteArray* node and a *Text* node automatically presents a dialog such as that shown in Figure 228, where you can specify the character encoding.

**Note:** This encoding is independent of the encoding specified for the OTD.

**Figure 228** Type Conversion - Character Encoding



## Byte-to-String Conversion

### Conversion Options

When converting a byte array to a Java string, the resulting string is defined by the character encoding. Please select from the following options:

- **Use the default character-encoding for this locale**

Selected by default, this option makes use of the default locale encoding, as set in the Java Virtual Machine containing the Logical host. You should select this option if you are dealing with a single encoding for all your data, and your Java Virtual Machine encoding is known to be correct.

- **Select a character-encoding**

Select this option to manually specify the encoding of your (known) byte[] data from the drop-down list of encodings supported by this version of Java.

## String-to-Byte Conversion

### Conversion Options

When converting a Java string to a byte array, the byte value for each character is defined by the character encoding used. Please select from the following options:

- **Use the default character-encoding for this locale**

Selected by default, this option makes use of the default locale encoding, as set in the Java Virtual Machine containing the Logical host. You should select this option if you are dealing with a single encoding for all your data, and your Java Virtual Machine encoding is known to be correct.

- **Select a character-encoding**

Select this option to manually specify the encoding expected by your application from the drop-down list of encodings supported by this version of Java.

**Note:** For an understanding of character encoding in general, please refer to <http://java.sun.com/j2se/1.4.2/docs/guide/intl/index.html>. Also, *CJVK Information* by Ken Lunde (O'Reilly) provides a detailed explanation of character encoding.

## 9.8 Using the Collaboration Tester (Java)

The Collaboration Tester (Java) supports testing of both marshalable and unmarshalable OTDs and, combined with the Java Source Editor, allows line-by-line debugging of the Java code.

**Note:** See also [Using the Java Debugger](#) on page 303 for a description of the Java Debugger, which works with deployed Collaborations.

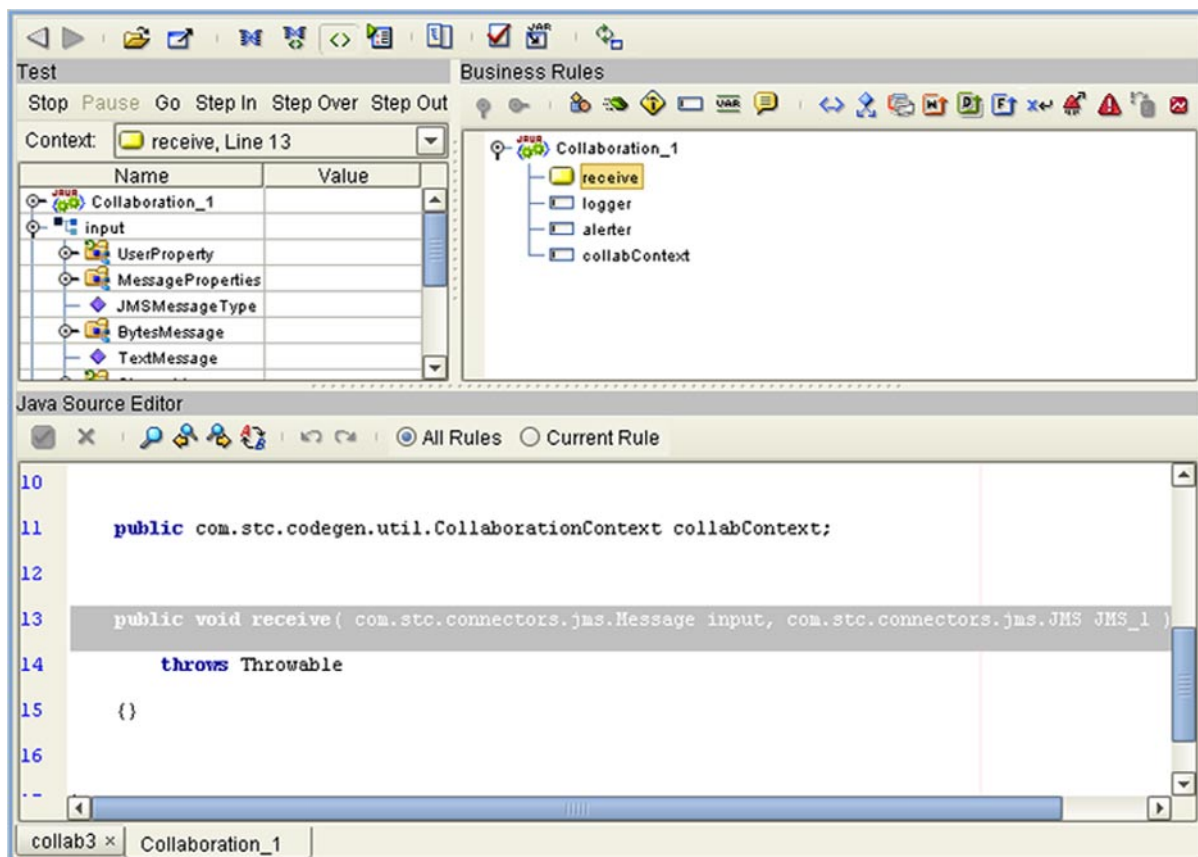
Selecting the **Test Mode** icon in the main Collaboration Editor toolbar (see Figure 229) displays the *Test* panel, which appears at the left of the Business Rules editor (see Figure 230).

**Figure 229** Test Mode Icon



**Note:** The function of this icon toggles between *display* and *hide*, depending upon the current state of the tester.

**Figure 230** Collaboration Tester (Java)



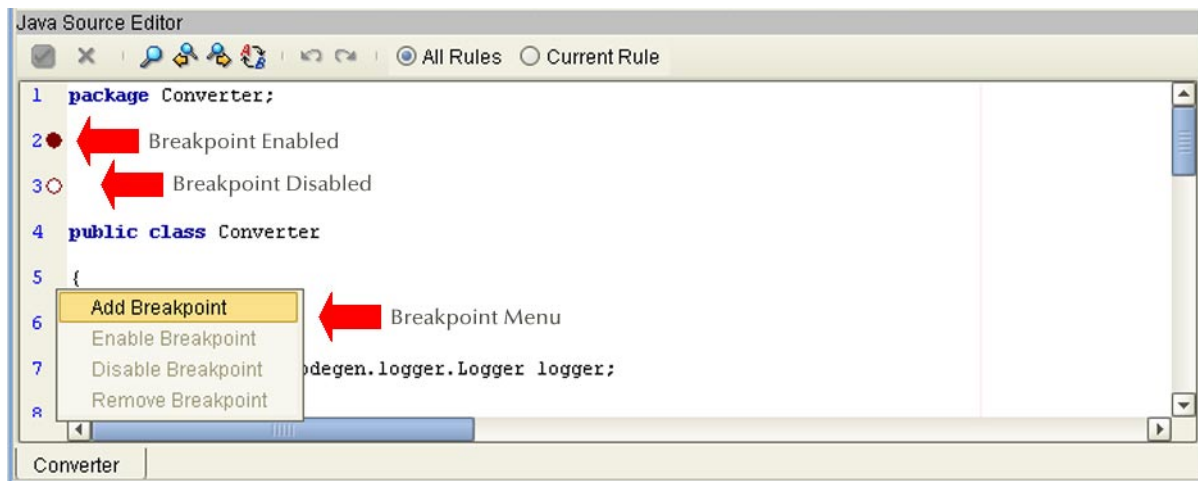
**Table 70** Collaboration Tester (Java) Commands

Command	Description
<b>Start / Stop</b>	Toggles to start or stop test procedure. After stopping, clicking <b>Start</b> refreshes the tree.
<b>Pause</b>	Pauses test procedure on current line.
<b>Go</b>	Proceed to next step.
<b>Step In</b>	Steps into Java code by one hierarchical level.
<b>Step Over</b>	Steps to next line of code, at same hierarchical level.
<b>Step Out</b>	Steps out of Java code by one hierarchical level.

### 9.8.1 Breakpoints

You can set breakpoints in the Java code where you want execution to stop during testing. A single click next to a line number in the Java Source Editor adds and enables a breakpoint, placing a red dot marker in the left column. Clicking on a red dot disables the breakpoint, changing the dot to white. Clicking on a white dot removes the breakpoint. The markers are illustrated in Figure 231.

**Figure 231** Breakpoints in Java Code



Alternatively, right-clicking next to a line number displays a breakpoint menu from which you can select the desired action (see Figure 232). In the absence of an existing breakpoint, the menu appears as shown in Figure 231. Right-clicking on an existing breakpoint enables the other options, as appropriate.

**Figure 232** Breakpoint Menu



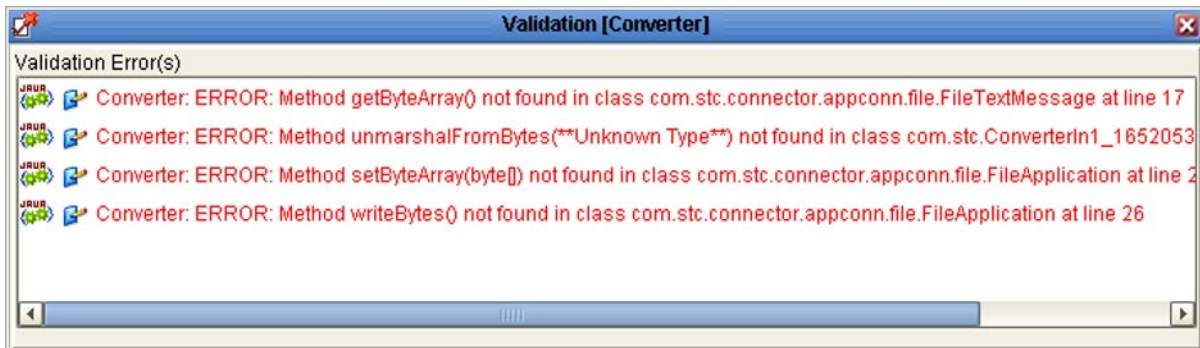
**Table 71** Breakpoint Menu Options

Option	Action
Add Breakpoint	Adds and enables a breakpoint at the selected line.
Enable Breakpoint	Enables a disabled breakpoint.
Disable Breakpoint	Disables an enabled breakpoint.
Remove Breakpoint	Removes the breakpoint.

## 9.8.2 Code Validation

The first step in testing is to validate the Java code. If errors are found, a *Validate* panel is displayed listing all errors (seeFigure 233). The test will not proceed until these errors are resolved.

**Figure 233** Validation Panel



### To test an OTD using the Collaboration Tester (Java)

- 1 Open a Collaboration to display the Collaboration Editor.
- 2 Click the **Source Code Mode** icon to display the Source Code Editor.

**Figure 234** Source Code Mode Icon



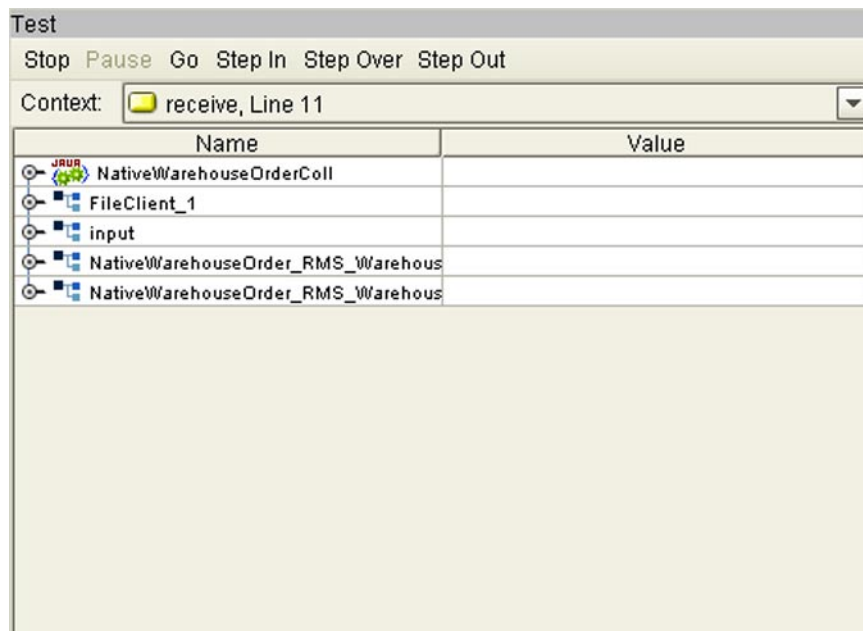
- 3 Click the **Test Mode** icon to display the Collaboration Tester.

**Figure 235** Test Mode Icon



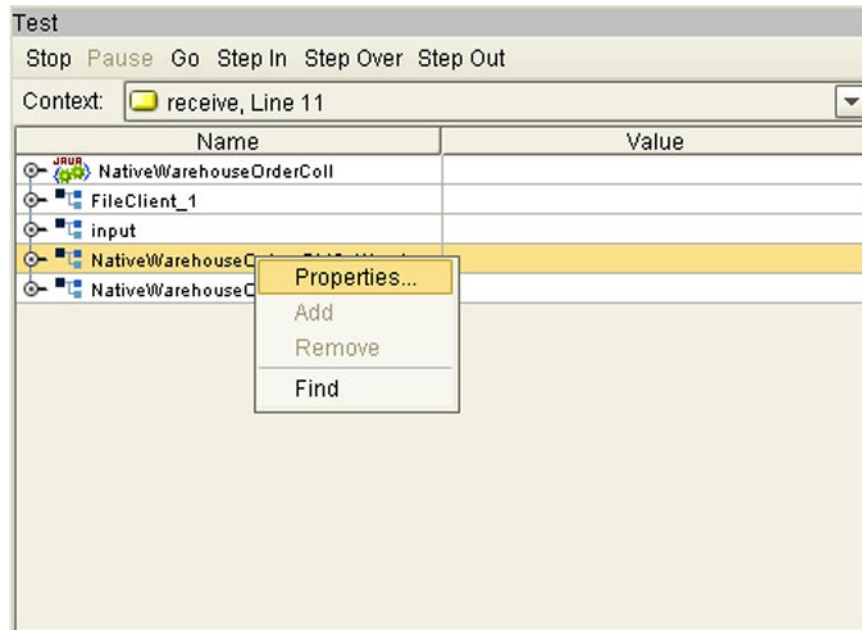
- 4 In the *Test* panel, click **Start** to initialize the Tester. The Collaboration tree will be displayed (see Figure 236), and the Tester will advance to the first rule (highlighted in the *Source Code* panel).

**Figure 236** Initial Collaboration Tree



- 5 In the tester, right-click an OTD root node to display its context menu, and select **Properties** (see Figure 237).

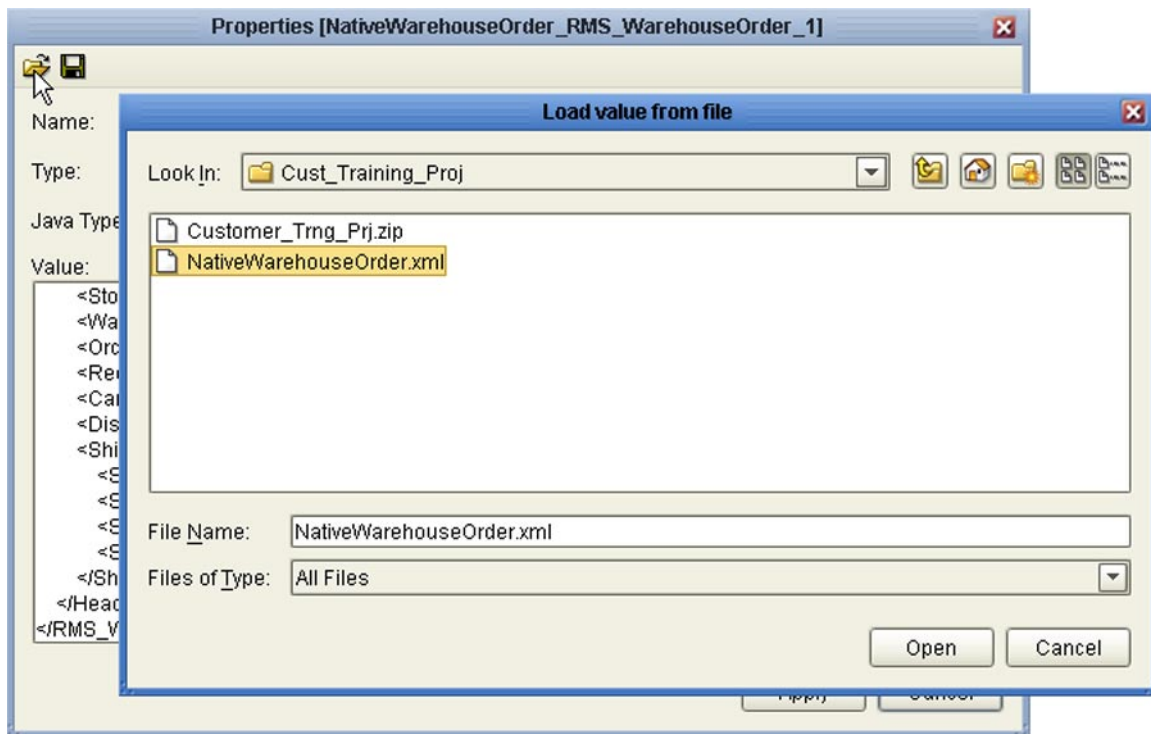
**Figure 237** Root Node Context Menu



- 6 In the *Properties* dialog, invoke the File Browser (see Figure 238).



**Figure 238** Selecting Input Data File



- 7 In the browser, select the desired input data file and click **Open**.
- 8 In the *Properties* dialog, click **Apply**. The *Value* fields will be populated in the Test panel (see Figure 239).

**Figure 239** Populated Value Fields

The screenshot shows the 'Test' window of the Collaboration Tester. At the top, there are buttons: Stop, Pause, Go, Step In, Step Over, and Step Out. Below these is a 'Context:' dropdown menu showing 'receive, Line 11'. The main part of the window is a table with two columns: 'Name' and 'Value'. The table lists various components and their associated values. The 'NativeWarehouseOrder\_RMS\_Warehouse' component is highlighted in yellow. Below it, the 'Header' component is expanded, showing a list of fields with their corresponding values. The 'Shipping\_Detail' and 'Detail' components are also listed but have no values. The 'NativeWarehouseOrder\_RMS\_Warehouse' component is listed again at the bottom of the table.

Name	Value
NativeWarehouseOrderColl	
FileClient_1	
input	
NativeWarehouseOrder_RMS_Warehouse	
Header	
PurchaseID	x000001
Order_Status	New
Store_Number	Store1
Warehouse_Number	Warehouse1
Order_Date	2/19/2003
Required_Date	3/1/2003
Cancel_Date	
Disposition_Time	
Shipping_Detail	
Detail	
NativeWarehouseOrder_RMS_Warehouse	

- ♦ Alternatively, you can manually type test data into the Value fields (double-quotes are not necessary for string values).
- 9 Click **Go** to run the tester through all the rules, or **Step Over** to test the rules one by one. (In this example all the rules are at a single level, so *Step In* and *Step Out* are not applicable.)
- 10 Click **Stop** to end the test.
- 11 Click **Start** to clear the Value fields and reset the Tester.

## 9.9 Using the Java Debugger

The Java Debugger enables you to debug Java-based Collaboration Definitions as deployed within an integration server on a Logical Host. As such, it offers an alternative to creating logs or warnings in an Java-based Collaboration and subsequently inspecting them via the Enterprise Manager.

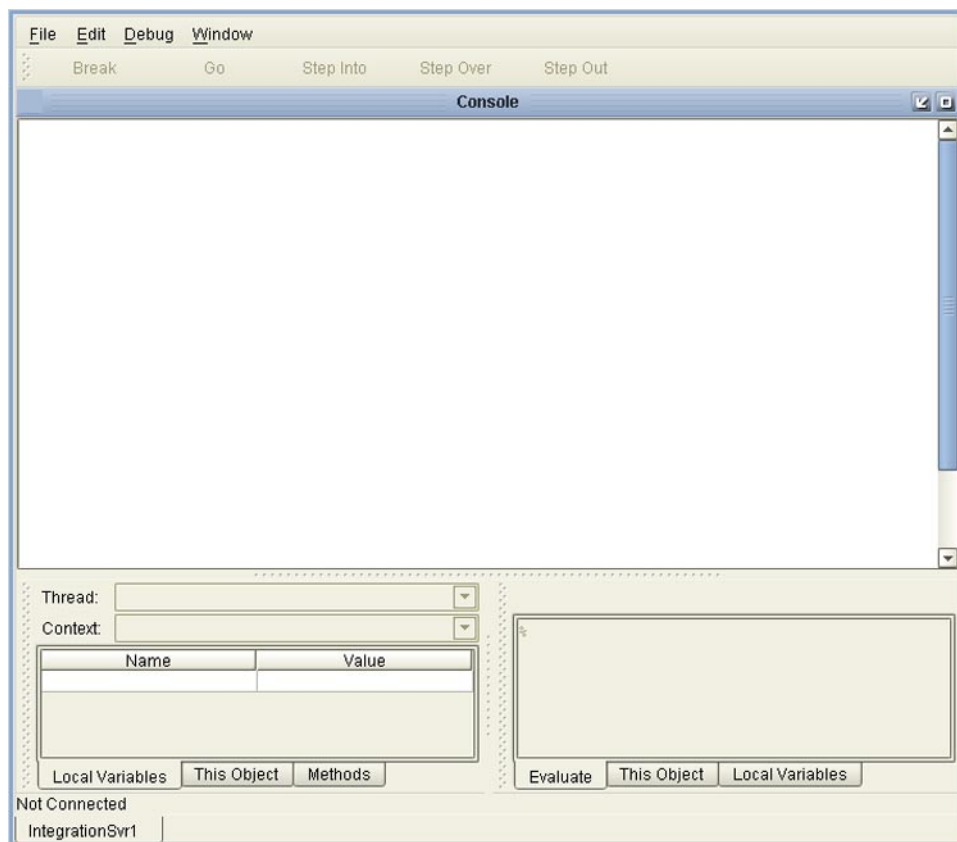
The Java Debugger functions in much the same way as the Collaboration Tester (see [Using the Collaboration Tester \(Java\)](#) on page 296), but operates in a run-time Environment. Both are displayed within the Collaboration Editor (Java).

### 9.9.1 Starting the Java Debugger

To start the Java Debugger

- 1 In Enterprise Explorer, select the appropriate integration server.
- 2 Right-click to display the context menu and click **Java Debugger**.
- 3 The Java Debugger appears in the Enterprise Designer Editor panel (see Figure 240).

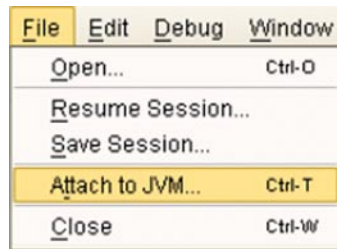
**Figure 240** Java Debugger



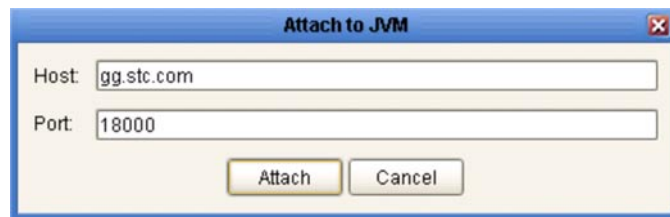
**Note:** The Java Debugger appears whether or not the connection was successful. If there is no **Connected to** message, try the following procedure:

- A Select **Attach to JVM** from the File menu (see Figure 241), which presents the *Attach to JVM* dialog (see Figure 242).

**Figure 241** File Menu

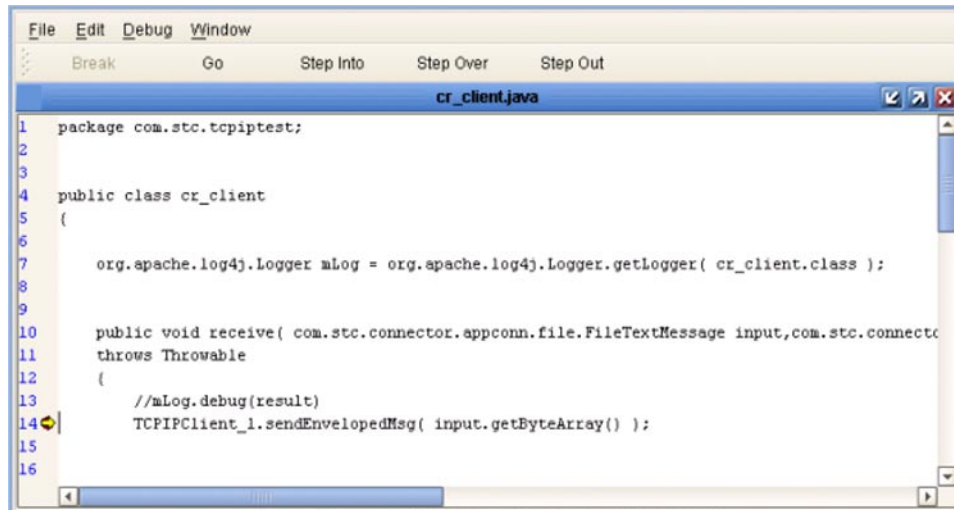


**Figure 242** Attach to JVM Dialog



- B Enter the integration server's host name and port number into the text boxes and click **Attach**. The debugger then re-attempts to connect to the integration server.
- 4 Once the Java Debugger is running, Java source code is displayed as soon as a Java-based Collaboration executes (see Figure 243).

**Figure 243** Java Source Code Display



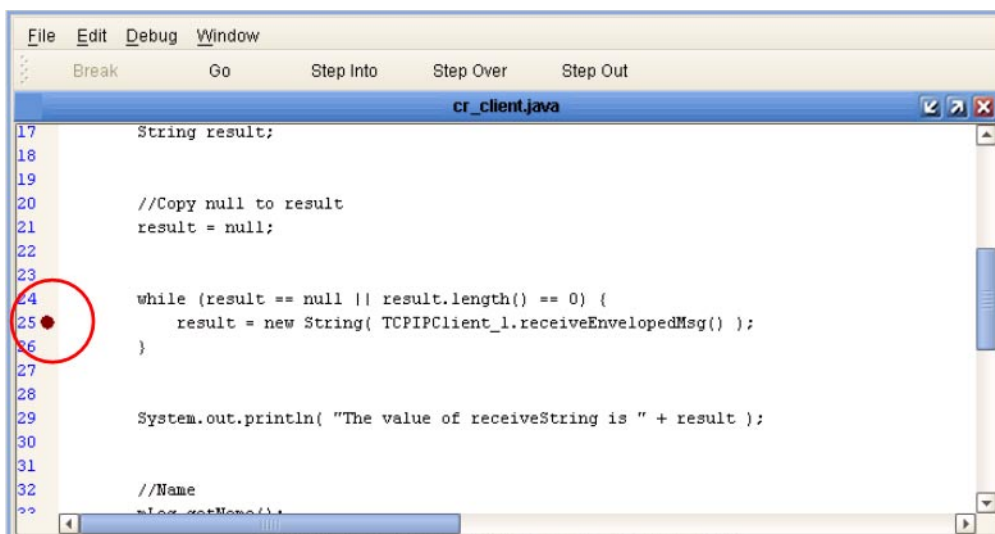
5 You can now set breakpoints to assist in examining and debugging the code.

## 9.9.2 Setting Breakpoints

To set a breakpoint

- 1 Click next to a line number in the executed source code. A red dot is displayed as a marker (see Figure 244).

**Figure 244** Breakpoint Example



- 2 Alternatively, you can set stops in a specific class or method, or have the debugger break on an exception. These options are available from the **Debug** menu.

- A To set a stop in a method, for example, select **Stop in Method** from the Debug menu (see Figure 245). A dialog is presented with which you can select the desired method (see Figure 246).

Figure 245 Debug Menu

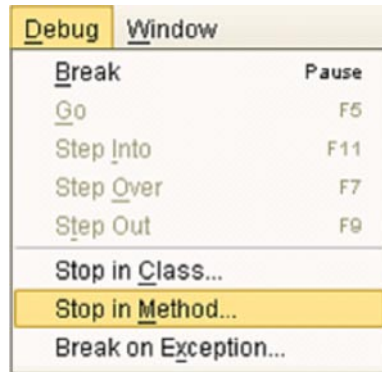
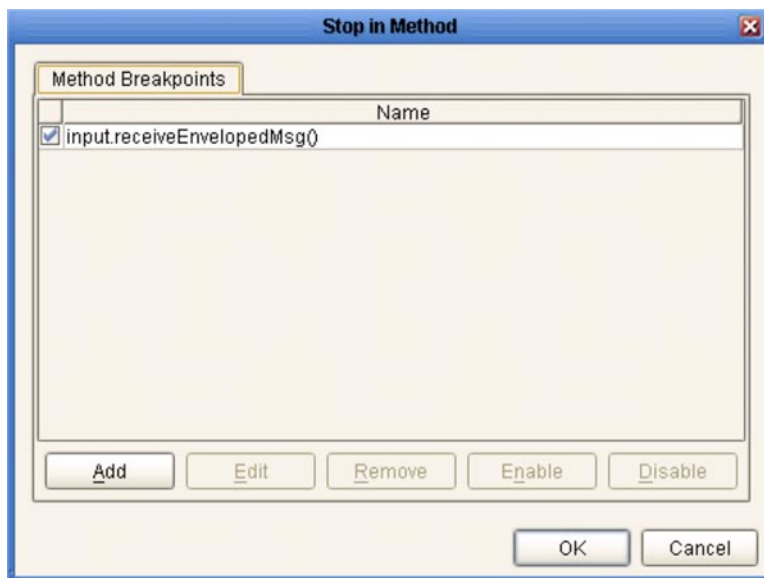
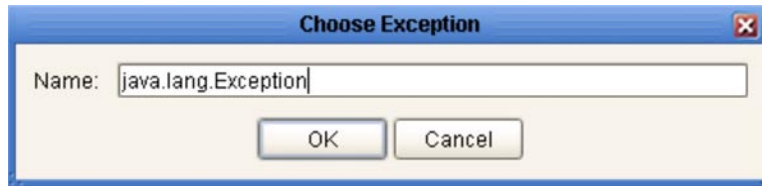


Figure 246 Stop in Method Dialog



- B To break on an exception, select **Break on Exception** from the Debug menu, which presents a *Choose Exception* dialog (see Figure 247). All occurrences of the specified exception are then trapped and reported (see Figure 248).

**Figure 247** Choose Exception Dialog



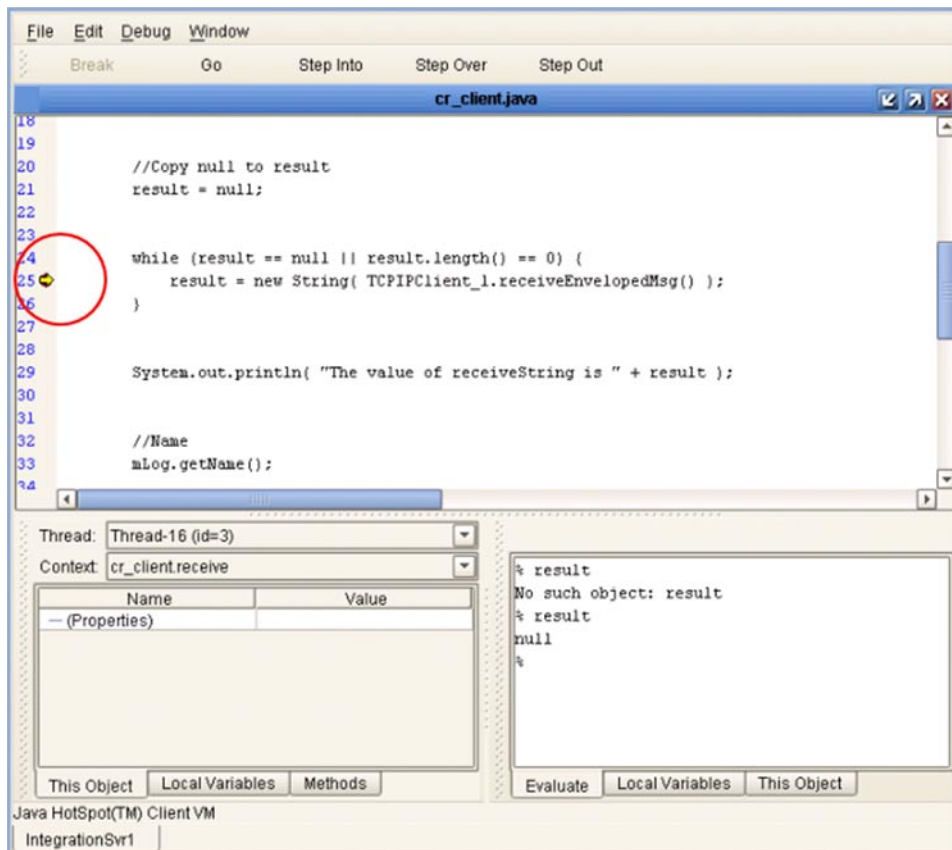
**Figure 248** Break on Exception Dialog



### 9.9.3 Inspecting and Editing the Source Code

As soon as the execution of the Java-based Collaboration arrives at a set breakpoint, it stops executing and displays an right arrow indicator next to the line number in the source code (see Figure 249).

**Figure 249** Breakpoint Indicator



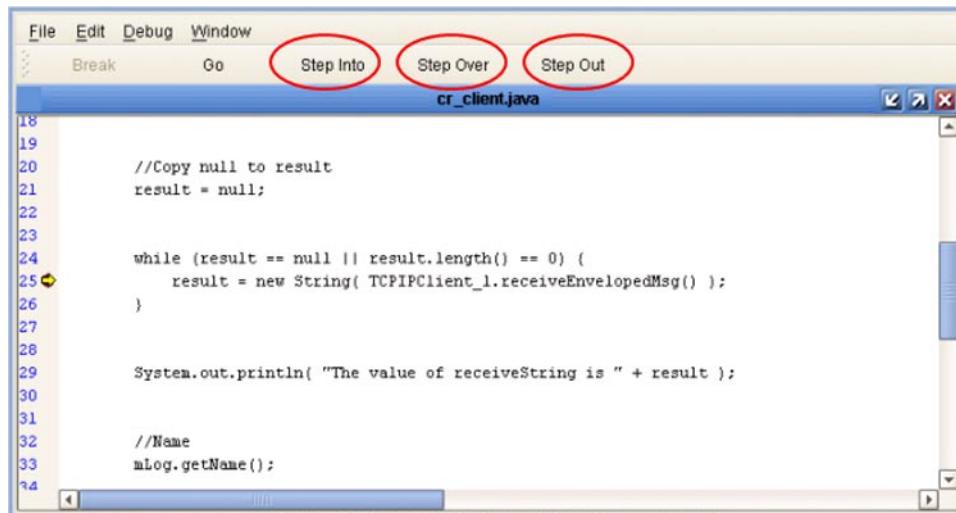
At this point, you can continue by (for example):

- ♦ Stepping Into
- ♦ Stepping Over
- ♦ Stepping Out
- ♦ Inspecting a local variable
- ♦ Setting a local variable



## Stepping Into, Over, or Out

**Figure 250** Stepping Into, Over, and Out Commands

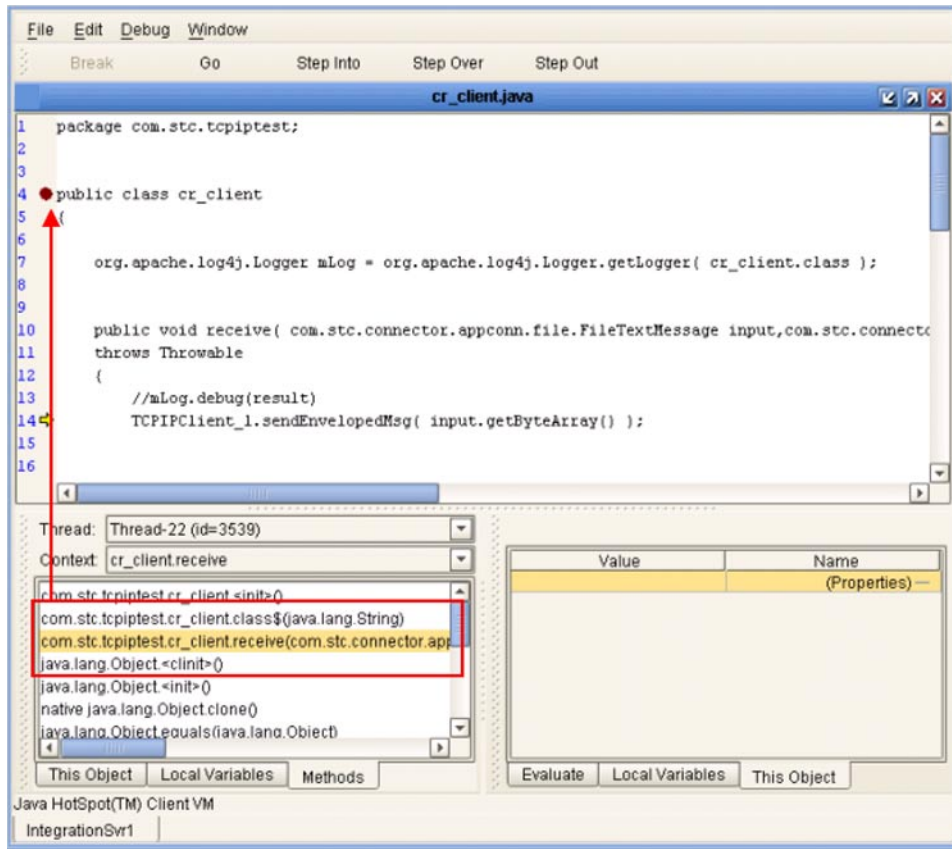


- By selecting the **Step Into** option (see Figure 250), the breakpoint is lifted and execution of the Collaboration will continue, *including* the line of code at the breakpoint.
- By selecting the **Step Over** option, the breakpoint is lifted and execution of the Collaboration will continue, *ignoring* the line of code at the breakpoint.
- By selecting the **Step Out** option, execution of the Collaboration is terminated.

## Inspecting Java Threads and Methods

Selecting the **Methods** tab in the left bottom panel of the debugger displays the currently executed Java thread and method (see Figure 251).

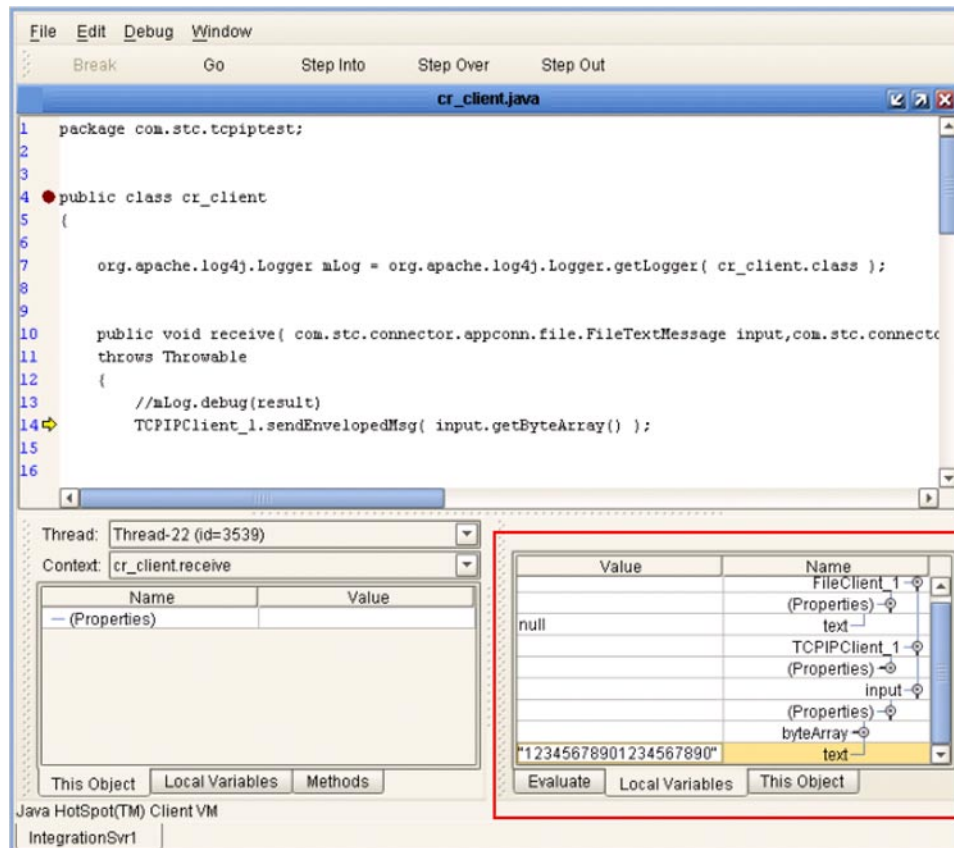
**Figure 251** Java Thread and Method Display



## Inspecting a Local Variable or Method

You can inspect a local variable by selecting the **Local Variables** tab in the right bottom panel of the debugger (see Figure 252). All nodes of the currently executed Java class are displayed here, and you can expand or collapse certain nodes to search for the value of the desired variable.

**Figure 252** Local Variables Tab

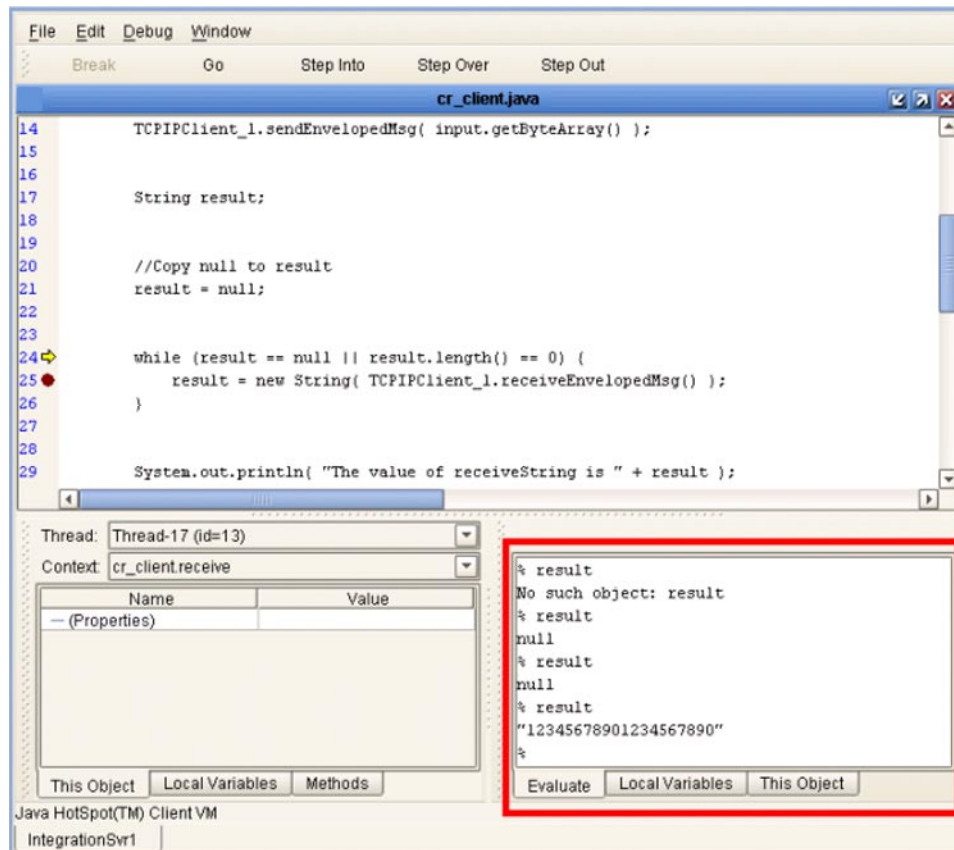


You also can inspect a local variable by selecting the **Evaluate** tab and entering the variable name in the panel using the following syntax:

```
% <variable_name>
```

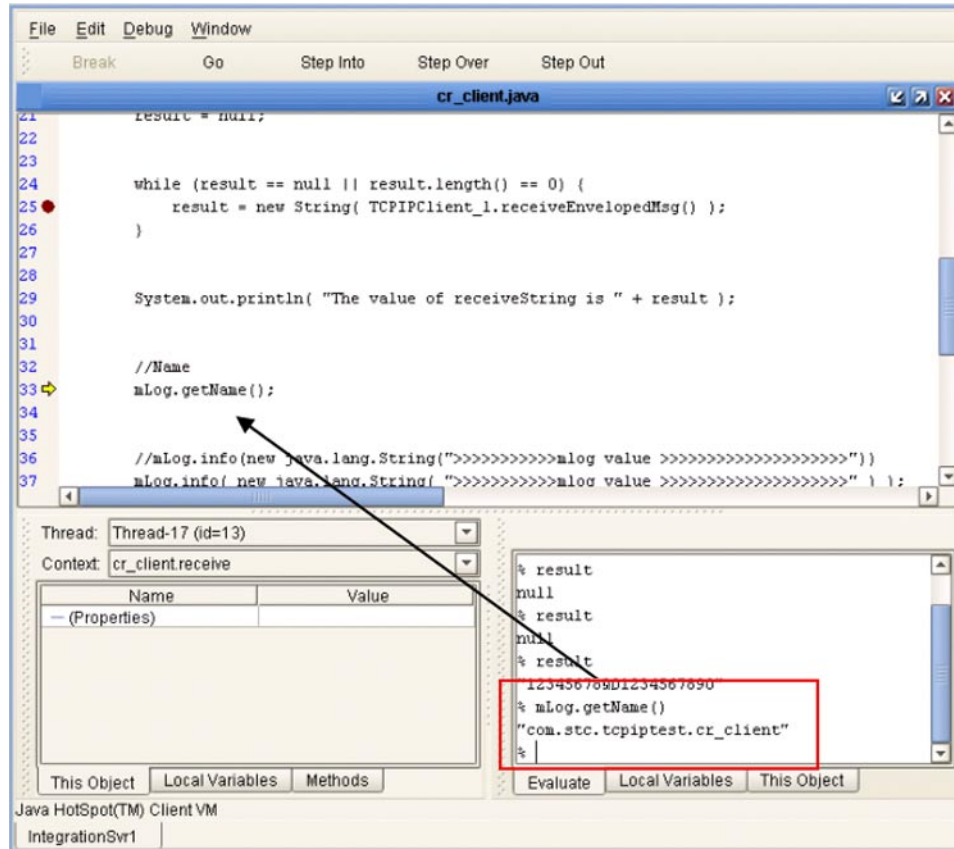
If the code has not initialized a variable, *No such object* is displayed; otherwise, the current value of the variable is displayed (see Figure 253).

**Figure 253** Evaluate Local Variable



You can also inspect the result of a method by entering the method name, following the same syntax format (see Figure 254).

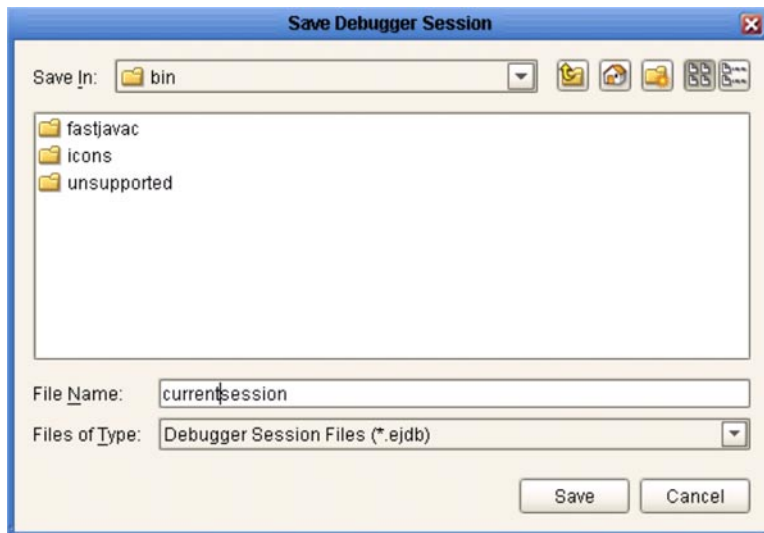
**Figure 254** Evaluate Method



## Saving and Resuming Debug Sessions

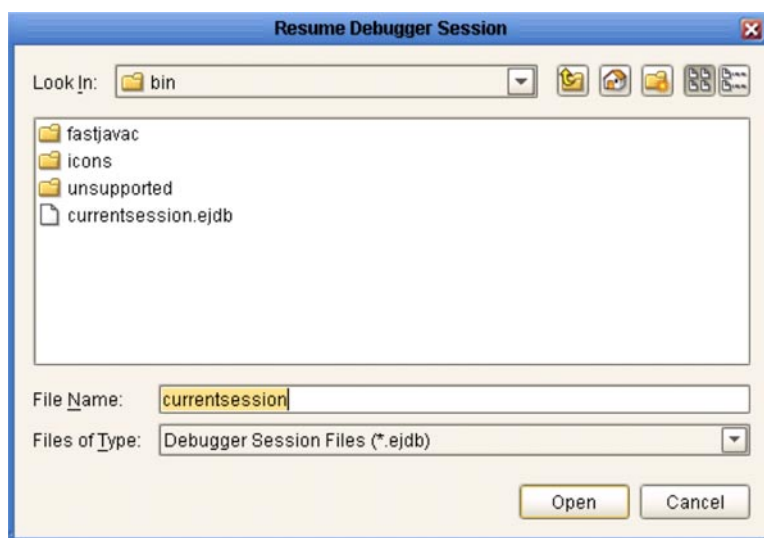
You can pause the debugging process by saving the session to a file. Selecting **Save Session** from the debugger File menu presents the *Save Debugger Session* dialog (see Figure 255), in which you specify the file name and location.

**Figure 255** Save Debugger Session Dialog



You can continue a debugging process that was paused by saving to a file. Selecting **Resume Session** from the debugger File menu presents the *Resume Debugger Session* dialog (see Figure 256), in which you specify the file name and location.

**Figure 256** Resume Debugger Session Dialog



# Collaboration Definitions (Java) II

This chapter outlines procedures for building Java-based Collaboration Definitions.

## What's in This Chapter

- [Creating a Java-based Collaboration Definition](#) on page 316
- [Editing Collaboration Definition Properties](#) on page 324
- [Creating Classes](#) on page 327
- [Creating Methods](#) on page 329
- [Creating Constructors](#) on page 333
- [Creating Fields](#) on page 336
- [Creating Arrays](#) on page 338
- [Creating and Modifying Variables](#) on page 340
- [Creating and Modifying Literals](#) on page 344
- [Generating Alerts](#) on page 345
- [Creating Log Entries](#) on page 349
- [Importing Third-Party Java Classes](#) on page 352
- [Adding Java classes to Collaboration Definitions](#) on page 354
- [Copying OTD Nodes](#) on page 356
- [Using Try-Catch](#) on page 358
- [Using Byte-Array Converters](#) on page 361
- [Validating Java-based Collaboration Definitions](#) on page 376

## 10.1 Creating a Java-based Collaboration Definition

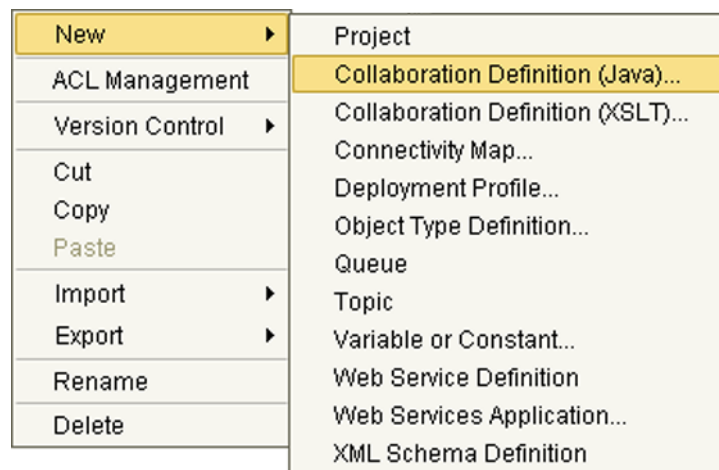
The Collaboration Definition Wizard (Java) guides you through the initial phases of creating a Java-based Collaboration Definition, and then invokes the Collaboration Editor (Java).

**Note:** Any changes made to the names of the Collaborations should be done when the Collaboration is created. If the name is changed later, the Collaboration should be opened, regenerated if applicable, and saved again. To be safe, this should also be done before creating the Connectivity Map and Deployment Profile.

### To create a Java-based Collaboration Definition

- 1 Right-click on a Project in the Enterprise Explorer to display the Project context menu (see Figure 257).

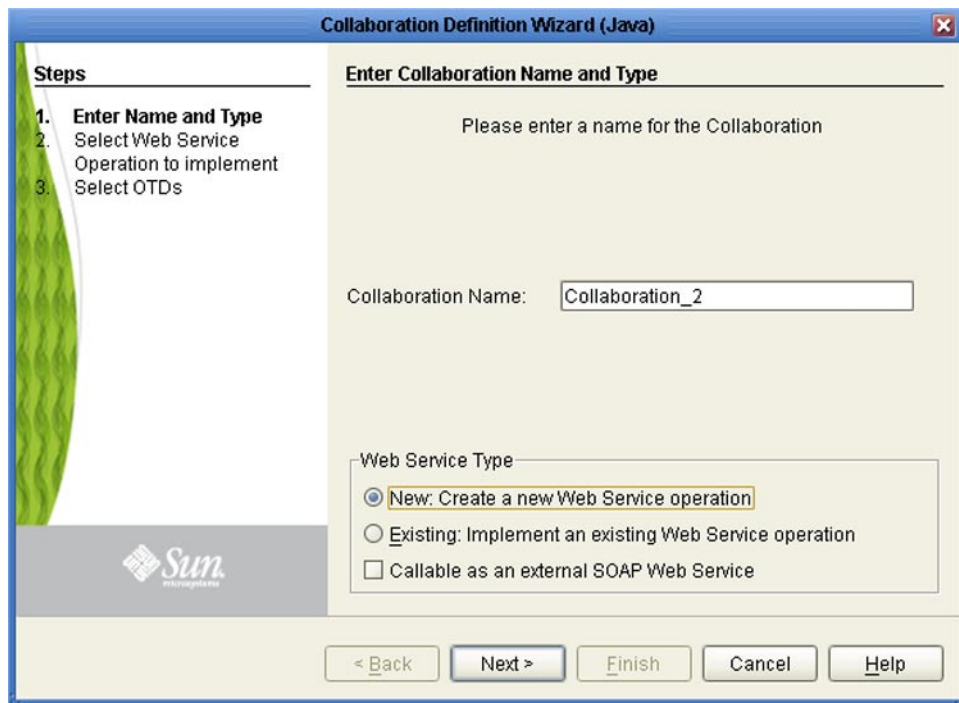
**Figure 257** Project Context Menu - New Web Service Definition



- 2 Select **New > Collaboration Definition (Java)** to invoke the Collaboration Definition Wizard (Java).
- 3 Enter a **Name** for your Collaboration, as shown in Figure 258.



**Figure 258** Initial Wizard Dialog



- 4 Select a Web Service Type, which can be either:
  - ♦ A **new** web service.
  - ♦ An **existing** web service (for example, an eInsight business process or an OTD).
- 5 If you want to expose the Collaboration Definition as a web service, check the box entitled **Callable as an external SOAP Web Service** (you must have selected a *new* web service).

**Note:** All XSD OTDs that are used in SOAP-callable Java Collaboration Definitions must be derived from XML schemas that have explicitly declared target namespaces.

- 6 Click **Next** to proceed to the next Wizard dialog.

The dialog sequence that appears next depends on which Web Service Type you selected in step 4.

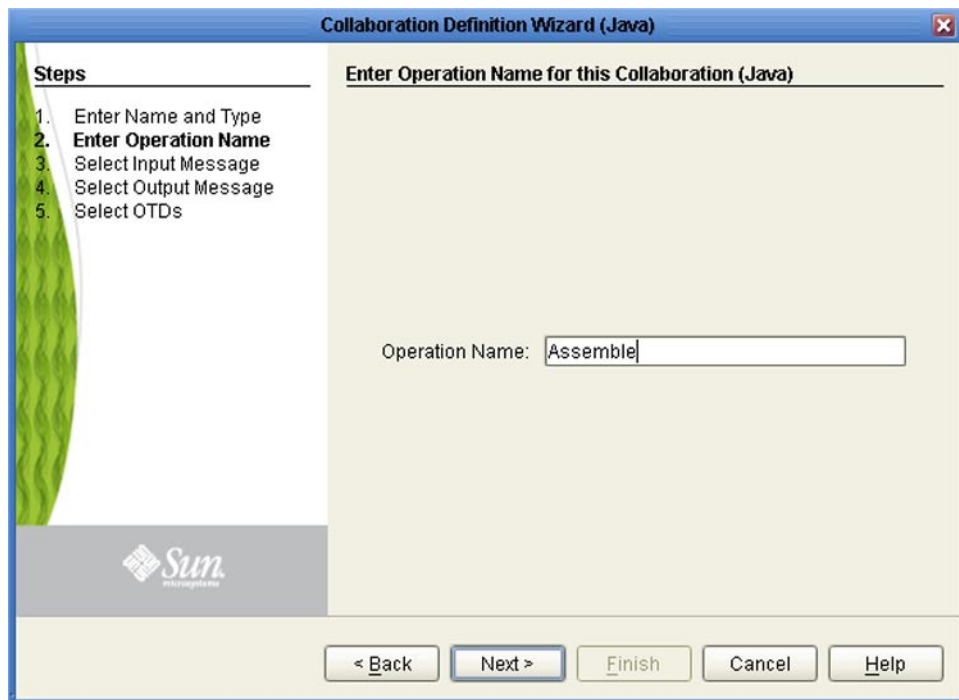
### 10.1.1 New Web Service

If you selected a new web service, you will be presented with the Wizard dialog shown in Figure 259.

To create a new web service

- 1 Enter an operation name, as shown in Figure 259. This will become the *method* that can be used to invoke the Java-based Collaboration as a web service.

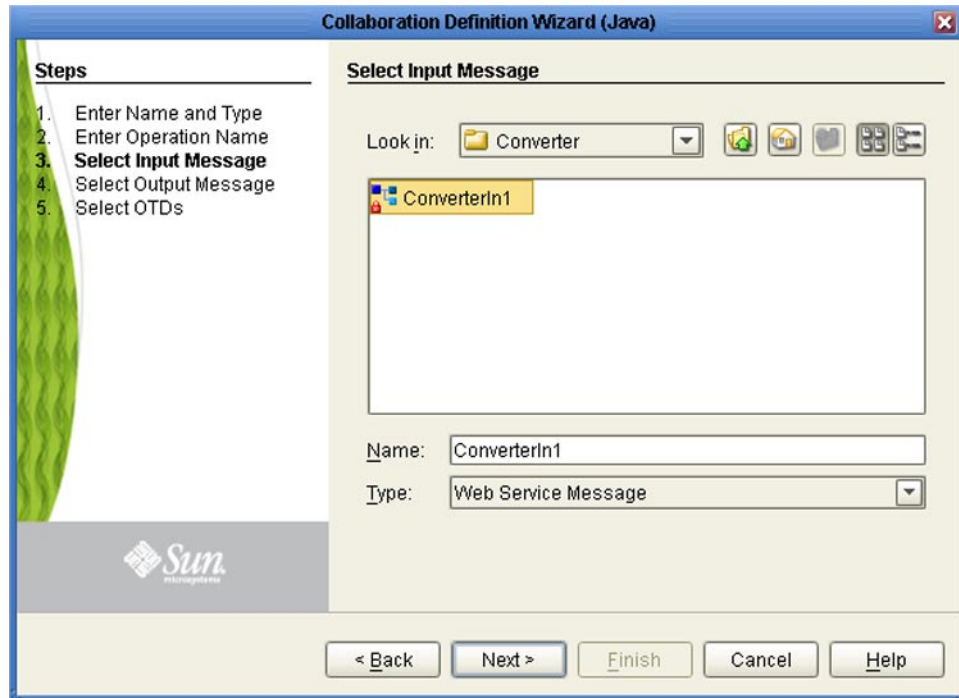
**Figure 259** New Web Service: Operation Name



- 2 Click **Next** to proceed to the next Wizard dialog.

- 3 Select the input web service message, (for example, *ConverterIn1* - shown in Figure 260).

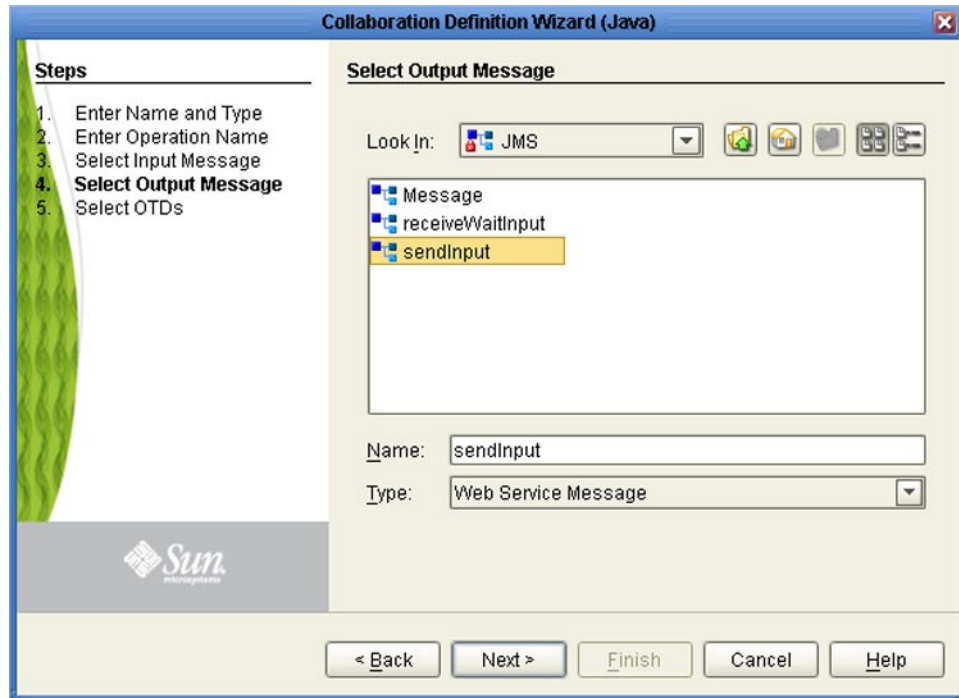
**Figure 260** New Web Service: Input Message



- 4 Click **Next** to proceed to the next Wizard dialog.

- 5 Select the output web service message, (for example, JMS *sendinput* - shown in Figure 261).

**Figure 261** New Web Service: Output Message

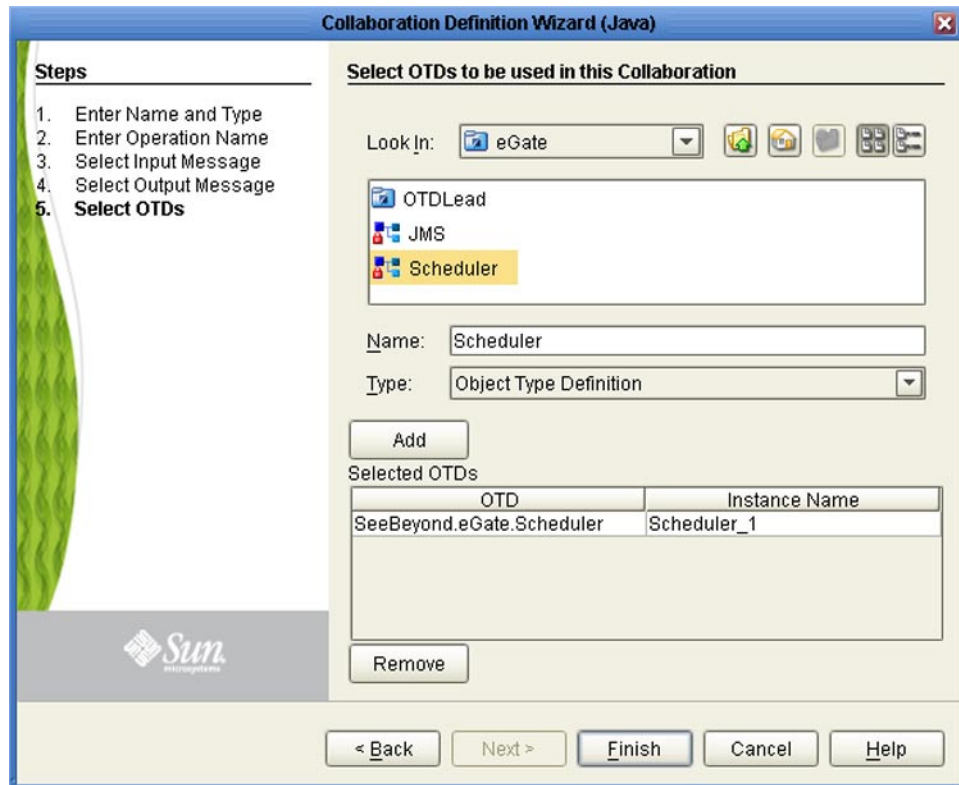


- 6 Click **Next** to proceed to the next Wizard dialog.

- 7 Select an auxiliary OTD, as shown in Figure 262. This step is optional, and is intended to support additional functionality. For example, if the Collaboration is triggered by a Scheduler adapter, you need to select the Scheduler OTD.

**Note:** Use caution here, since you may already have OTDs selected in the preceding steps.

**Figure 262** New Web Service: Auxiliary OTD



- 8 Click **Add** to add the OTD to the Collaboration Definition.
- 9 Click **Next** to proceed to the Collaboration Editor (Java).

## 10.1.2 Existing Web Service

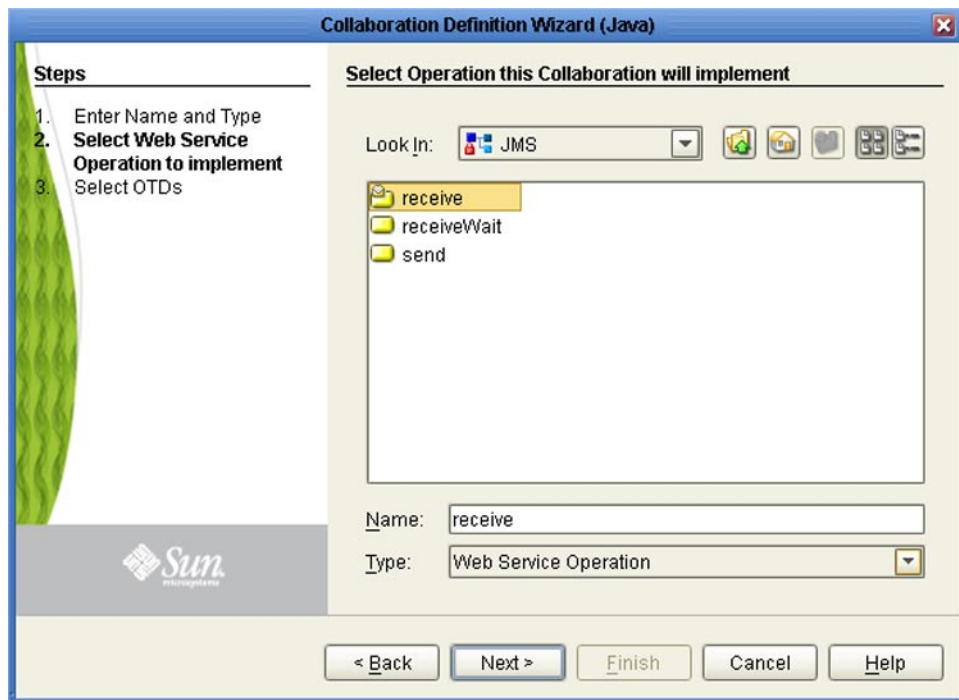
If you selected an existing web service, you will be presented with the Wizard dialog shown in Figure 263.

**Note:** The option to expose the Collaboration Definition as a web service is not available in this situation. You must create a new web service operation to do so.

To use an existing web service

- 1 Select a web service operation, which must be an installed web service. For example, if the input to the Collaboration is from an eWay, you need to select JMS *receive* (see Figure 263).

**Figure 263** Existing Web Service: Select Operation

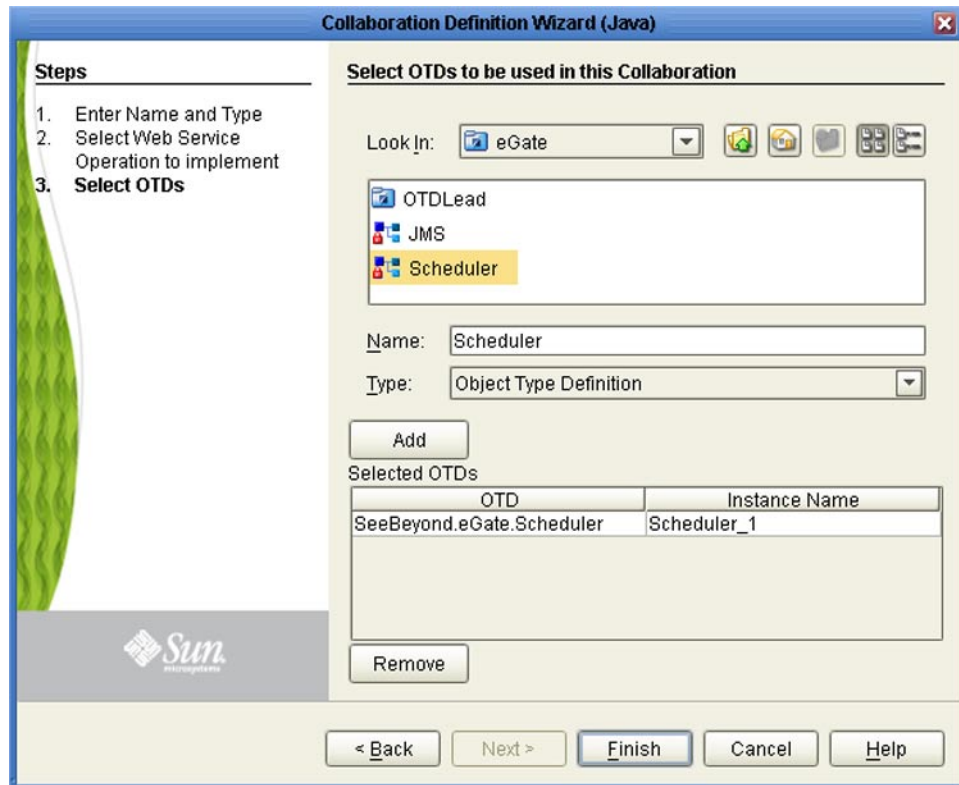


- 2 Click **Next** to proceed to the next Wizard dialog.

- 3 Select an auxiliary OTD, as shown in Figure 264. This step is optional, and is intended to support additional functionality. For example, if the Collaboration is triggered by a Scheduler adapter, you need to select the Scheduler OTD.

**Note:** Use caution here, since you may already have OTDs selected in the preceding steps.

**Figure 264** Existing Web Service: Select OTD



- 4 Click **Add** to add the OTD to the Collaboration Definition.
- 5 Click **Next** to proceed to the Collaboration Editor (Java).

## 10.2 Editing Collaboration Definition Properties

Right-clicking a Collaboration Definition (Java) in the Project Explorer displays the context menu shown in [Using the Collaboration Definition Context Menu](#) on page 244. Selecting **Properties** from the menu presents the *Collaboration Definition (Java) Properties* dialog for the selected Collaboration Definition (see Figure 265).

This dialog is similar to the Collaboration Definition Wizard, and shows the property values that were set previously. By default, the Operation Configuration field is set to **Keep Current Operation**; to edit the property values, you must select another command. The options are:

- **Keep Current Operation**, which only allows you to add or remove auxiliary OTDs.
- **Choose an Existing Operation**, which allows you to reset the properties for the current operation, or select a different (existing) operation.
- **Create a New Operation**, which allows you to create a new operation to replace the existing one.

**Figure 265** Collaboration Definition (Java) Properties

**Collaboration Definition (Java) Properties**

Collaboration Name:

Operation Configuration:

Operation Name:  ...

Input Message:  ...

☐ Output Message:  ...

☐ Callable as an external SOAP Web Service

Object Type Definitions:

OTD	Instance Name
DemoProject.emp_input_Employee	emp_input_Employee_1



**Important:** If you change any Java-based Collaboration that implements existing web services, be sure to reset the web service in the Collaboration Definition (Java) Properties dialog, as described in the following procedure.

To reset the configuration properties for an existing operation

- 1 In the Explorer, right-click the Collaboration Definition that has been changed.
- 2 Select **Properties** from the Collaboration Definition context menu (see [Using the Collaboration Definition Context Menu](#) on page 244).
- 3 Select **Choose an Existing Operation** from the Operation Configuration list (see Figure 266).

**Figure 266** Collaboration Definition (Java) Properties

OTD	Instance Name
DemoProject.emp_input_Employee	emp_input_Employee_1

- 4 Select the same web service operation as before in **Operation Name** (click the ellipsis [...] button, as shown in Figure 266, to display a dialog for selecting the operation).

**Note:** The input and output messages are automatically selected for existing operations.

**Note:** The option to expose the Collaboration Definition as a web service is not available in this situation. You must create a new web service operation to do so.

- 5 Add or remove any auxiliary OTDs, if desired. These OTDs are optional, and are to support additional functionality such as a database lookup.
- 6 Click **OK** to finish.

**To set the configuration properties for a new operation**

- 1 In the Explorer, right-click the Collaboration Definition.
- 2 Select **Properties** from the Collaboration Definition context menu (see [Using the Collaboration Definition Context Menu](#) on page 244).
- 3 Select **Create a New Operation** from the Operation Configuration list (see Figure 267).

**Figure 267** Collaboration Definition (Java) Properties

Collaboration Name:

Operation Configuration:

Operation Name:

Input Message:

☐ Output Message:

☐ Callable as an external SOAP Web Service

Object Type Definitions:

OTD	Instance Name
DemoProject.emp_input_Employee	emp_input_Employee_1

- 4 Type in the desired web service operation in **Operation Name**.
- 5 Select the desired input message and (optional) output message (click the ellipsis [...] button, as shown in Figure 267, to display a dialog for selecting the message).
- 6 If you want to expose the Collaboration Definition as a web service, check the box entitled **Callable as an external SOAP Web Service**.
- 7 Add or remove any auxiliary OTDs, if desired. These OTDs are optional, and are to support additional functionality such as a database lookup.
- 8 Click **OK** to finish.

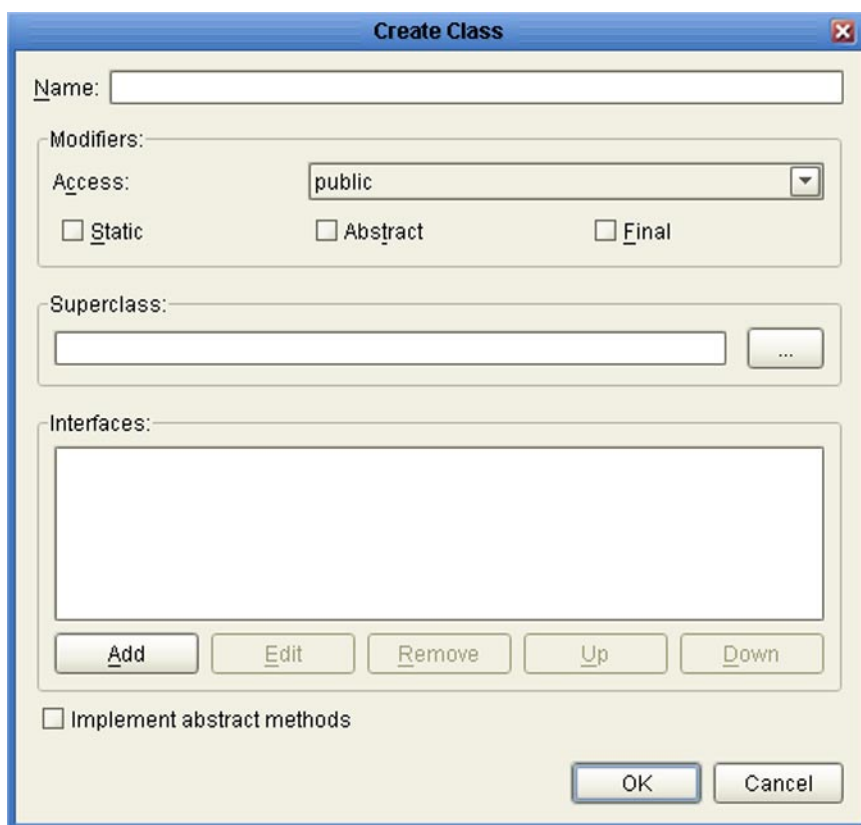
## 10.3 Creating Classes

Clicking the **Class** icon in the Business Rules Editor toolbar (see Figure 268) presents the *Create Class* dialog (see Figure 277).

**Figure 268** Class Icon



**Figure 269** Create Class Dialog



The **Create Class** dialog box contains the following fields and controls:

- Name:** A text input field.
- Modifiers:**
  - Access:** A dropdown menu currently showing **public**.
  - ☐ **Static**
  - ☐ **Abstract**
  - ☐ **Final**
- Superclass:** A text input field with a browse button (three dots) to its right.
- Interfaces:** A large empty list box with five buttons below it: **Add**, **Edit**, **Remove**, **Up**, and **Down**.
- ☐ **Implement abstract methods**
- OK** and **Cancel** buttons at the bottom right.

The various options are summarized in Table 72.

**Table 72** Class Options

Heading	Option/Command	Description
Name		Your name for the class.
Modifiers	Access	One of the following: <ul style="list-style-type: none"> <li>▪ public</li> <li>▪ protected</li> <li>▪ private</li> </ul>
	Static	Specifies that the class is static.
	Abstract	Specifies that the class is abstract.
	Final	Specifies that the class is final (no subclasses can be created).
Superclass		Assigns the class to another Java class. Clicking the ellipsis (...) button launches the <i>Class Browser</i> (see <a href="#">Using the Class Browser</a> on page 282).
Interfaces		Displays a list of interfaces for the method.
	Add	Invokes the <i>Class Browser</i> with which you can locate an interface to add (see <a href="#">Using the Class Browser</a> on page 282).
	Edit	Presents a dialog in which you can edit the interface.
	Remove	Deletes the selected interface.
	Up	Moves the selected interface up one level in the list.
	Down	Moves the selected interface down one level in the list.
Implement Abstract Methods		Permits the implementation of abstract methods within the class.

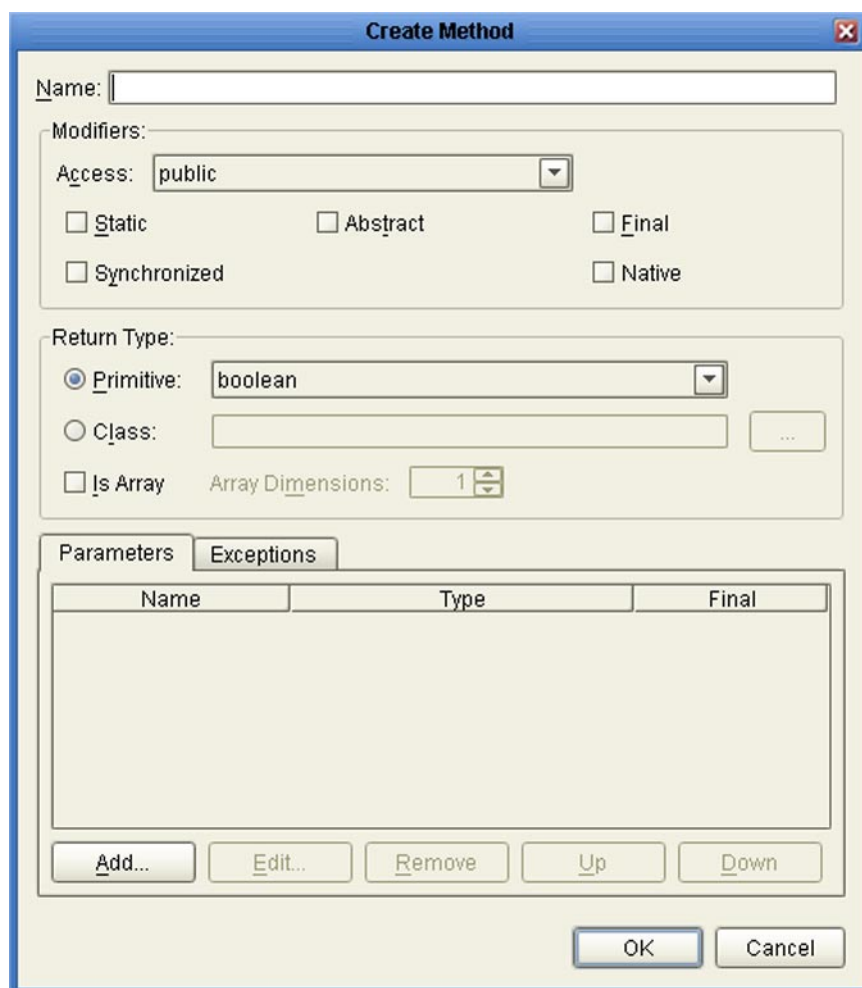
## 10.4 Creating Methods

Clicking the **Method** icon in the Business Rules Editor toolbar (see Figure 270) presents the *Create Method* dialog (see Figure 271).

**Figure 270** Method Icon



**Figure 271** Create Method Dialog



The **Create Method** dialog box is shown. It has a title bar with the text "Create Method" and a close button. The dialog is divided into several sections:

- Name:** A text field for entering the method name.
- Modifiers:** A section containing a dropdown menu for **Access** (currently set to "public") and five checkboxes: **Static**, **Abstract**, **Final**, **Synchronized**, and **Native**.
- Return Type:** A section with three options:
  - Primitive:** A radio button (selected) followed by a dropdown menu (currently set to "boolean").
  - Class:** A radio button followed by a text field and a button with three dots "...".
  - Is Array:** A checkbox followed by the text "Array Dimensions:" and a spinner box (currently set to "1").
- Parameters:** A tabbed section with two tabs: **Parameters** (selected) and **Exceptions**. The **Parameters** tab contains a table with three columns: **Name**, **Type**, and **Final**. The table is currently empty. Below the table are five buttons: **Add...**, **Edit...**, **Remove**, **Up**, and **Down**.
- Buttons:** At the bottom right are **OK** and **Cancel** buttons.

The various options are summarized in Table 73.

**Table 73** Method Options

Heading	Option/Command	Description
Name		Your name for the method.
Modifiers	Access	One of the following: <ul style="list-style-type: none"> <li>▪ public</li> <li>▪ protected</li> <li>▪ private</li> </ul>
	Static	Specifies that the method is static.
	Synchronized	Specifies that the method is synchronized.
	Abstract	Specifies that the method is abstract.
	Final	Specifies that the method is final.
	Native	Specifies that the method is native.
Return Type	Primitive	One of the following: <ul style="list-style-type: none"> <li>▪ boolean</li> <li>▪ byte</li> <li>▪ char</li> <li>▪ short</li> <li>▪ int</li> <li>▪ long</li> <li>▪ float</li> <li>▪ double</li> </ul>
	Class	Assigns the method to a Java class. Clicking the ellipsis (...) button launches the <i>Class Browser</i> (see <a href="#">Using the Class Browser</a> on page 282).
	Is Array	Specifies that the method returns an array. When checked, enables the Array Dimensions list box, where you can set the array dimensions.
Parameters		Displays a list of parameters for the method.
	Add	Presents the <i>Create Parameter</i> dialog, in which you can define a parameter (see <a href="#">Creating Parameters</a> on page 331).
	Edit	Presents the <i>Parameter</i> dialog (identical to the <i>Create Parameter</i> dialog), in which you can edit the selected parameter.
	Remove	Deletes the selected parameter.
	Up	Moves the selected parameter up one level in the list.
	Down	Moves the selected parameter down one level in the list.

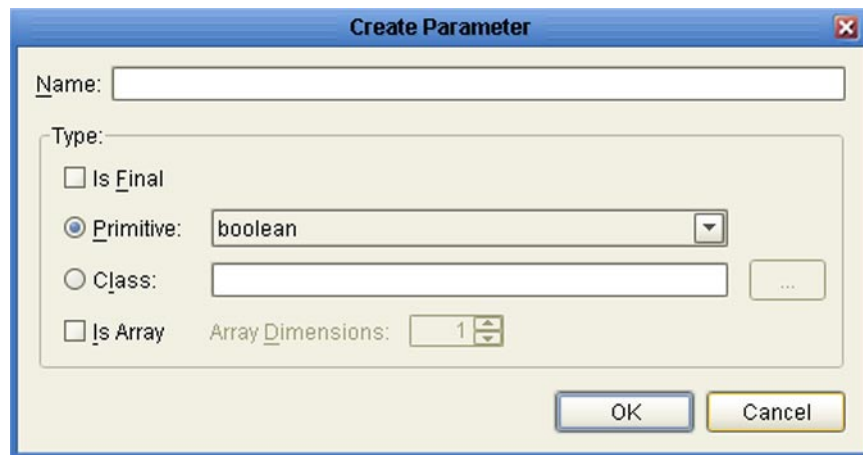
**Table 73** Method Options

Heading	Option/Command	Description
Exceptions		Displays a list of exception classes thrown by the method.
	Add	Invokes the <i>Class Browser</i> , with which you can locate an exception class (see <a href="#">Using the Class Browser</a> on page 282).
	Edit	Invokes the <i>Class Browser</i> , with which you can replace the selected class with another.
	Remove	Deletes the selected exception from the list.
	Up	Moves the selected exception class up one level in the list.
	Down	Moves the selected exception class down one level in the list.

### 10.4.1 Creating Parameters

Clicking **Add** in the *Parameters* panel of the *Create Method* dialog presents the *Create Parameter* dialog, shown in Figure 272.

**Figure 272** Create Parameter Dialog



The various options are summarized in Table 74.

**Table 74** Parameter Options

Heading	Option	Description
Name		Your name for the parameter.
Type	Is Final	Specifies that the parameter is final.
	Primitive	One of the following: <ul style="list-style-type: none"> <li>▪ boolean</li> <li>▪ byte</li> <li>▪ char</li> <li>▪ short</li> <li>▪ int</li> <li>▪ long</li> <li>▪ float</li> <li>▪ double</li> </ul>
	Class	Assigns the parameter to a Java class. Clicking the ellipsis (...) button launches the <i>Class Browser</i> (see <a href="#">Using the Class Browser</a> on page 282).
	Is Array	Specifies that the parameter is an array. When checked, enables the Array Dimensions field, where you set the array size (square).



## 10.5 Creating Constructors

Clicking the **Constructor** icon in the Business Rules Editor toolbar (see Figure 273) presents the *Create Constructor* dialog (see Figure 277).

**Figure 273** Constructor Icon



**Figure 274** Create Constructor Dialog

Name	Type	Final
------	------	-------

The various options are summarized in Table 75.

**Table 75** Constructor Options

Heading	Option/Command	Description
Modifiers	Access	One of the following: <ul style="list-style-type: none"> <li>▪ public</li> <li>▪ protected</li> <li>▪ private</li> </ul>
Parameters		Displays a list of parameters for the method.
	Add	Presents the <i>Create Parameter</i> dialog, in which you can define a parameter (see <a href="#">Creating Parameters</a> on page 335).
	Edit	Presents the <i>Parameter</i> dialog (identical to the <i>Create Parameter</i> dialog), in which you can edit the selected parameter.
	Remove	Deletes the selected parameter.
	Up	Moves the selected parameter up one level in the list.
	Down	Moves the selected parameter down one level in the list.
Exceptions		Displays a list of exception classes thrown by the method.
	Add	Invokes the <i>Class Browser</i> , with which you can locate an exception class (see <a href="#">Using the Class Browser</a> on page 282).
	Edit	Invokes the <i>Class Browser</i> , with which you can replace the selected class with another.
	Remove	Deletes the selected exception from the list.
	Up	Moves the selected exception class up one level in the list.
	Down	Moves the selected exception class down one level in the list.

## 10.5.1 Creating Parameters

Clicking **Add** in the Parameters panel of the *Create Constructor* dialog presents the *Create Parameter* dialog, shown in Figure 272.

**Figure 275** Create Parameter Dialog

The various options are summarized in Table 74.

**Table 76** Parameter Options

Heading	Option	Description
Name		Your name for the parameter.
Type	Is Final	Specifies that the parameter is final.
	Primitive	One of the following: <ul style="list-style-type: none"> <li>boolean</li> <li>byte</li> <li>char</li> <li>short</li> <li>int</li> <li>long</li> <li>float</li> <li>double</li> </ul>
	Class	Assigns the parameter to a Java class. Clicking the ellipsis (...) button launches the <i>Class Browser</i> (see <a href="#">Using the Class Browser</a> on page 282).
	Is Array	Specifies that the parameter is an array. When checked, enables the Array Dimensions field, where you set the array size (square).

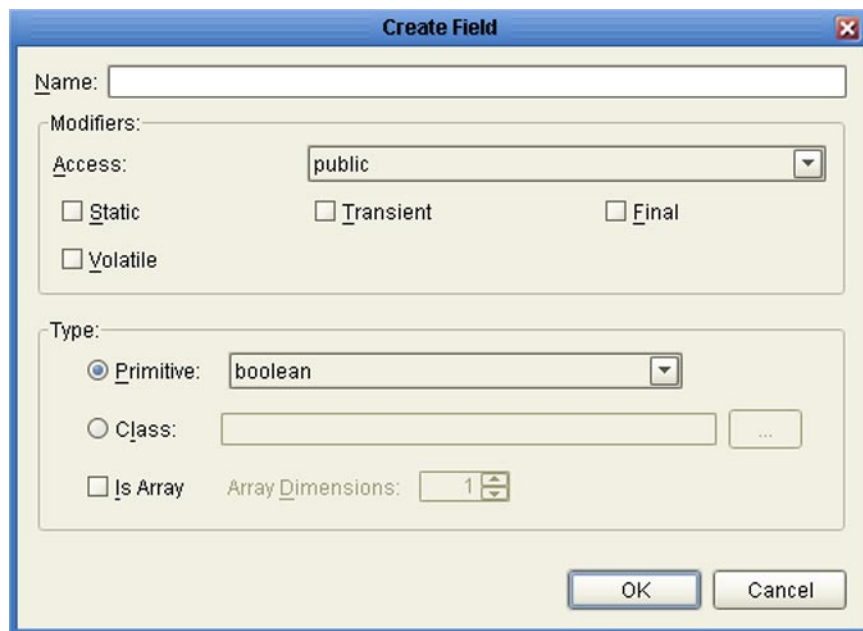
## 10.6 Creating Fields

Clicking the **Field** icon in the Business Rules Editor toolbar (see Figure 276) presents the **Create Field** dialog (see Figure 277).

**Figure 276** Field Icon



**Figure 277** Create Field Dialog



The **Create Field** dialog box is shown with the following fields and options:

- Name:** A text input field.
- Modifiers:**
  - Access:** A dropdown menu set to **public**.
  - ☐ **Static**
  - ☐ **Transient**
  - ☐ **Final**
  - ☐ **Volatile**
- Type:**
  - ☒ **Primitive:** A dropdown menu set to **boolean**.
  - ☐ **Class:** A text input field with a browse button (...).
  - ☐ **Is Array:** A checkbox.
  - Array Dimensions:** A spinner box set to **1**.
- Buttons:** **OK** and **Cancel** buttons at the bottom right.

The various options are summarized in Table 77.

**Table 77** Field Options

Heading	Option	Description
Name		Your name for the field.
Modifiers	Access	One of the following: <ul style="list-style-type: none"> <li>▪ public</li> <li>▪ protected</li> <li>▪ private</li> </ul>
	Static	Specifies that the field is static.
	Volatile	Specifies that the field is volatile.
	Transient	Specifies that the field is transient.
	Final	Specifies that the field is final.
Type	Primitive	One of the following: <ul style="list-style-type: none"> <li>▪ boolean</li> <li>▪ byte</li> <li>▪ char</li> <li>▪ short</li> <li>▪ int</li> <li>▪ long</li> <li>▪ float</li> <li>▪ double</li> </ul>
	Class	Assigns the field to a Java class. Clicking the ellipsis (...) button launches the <i>Class Browser</i> (see <a href="#">Using the Class Browser</a> on page 282).
	Is Array	Specifies that the variable is an array. When checked, enables the Array Dimensions list box, where you can set the array dimensions.

## 10.7 Creating Arrays

Clicking the **New Array** icon in the Business Rules Designer toolbar (see Figure 278) presents the **New Array** dialog (see Figure 279).

**Figure 278** New Array Icon



**Figure 279** New Array Dialog

The various options are summarized in Table 78.

**Table 78** Array Options

Heading	Option	Description
Array Type	Primitive	One of the following: <ul style="list-style-type: none"> <li>boolean</li> <li>byte</li> <li>char</li> <li>short</li> <li>int</li> <li>long</li> <li>float</li> <li>double</li> </ul>
	Class	Assigns the field to a Java class. Clicking the ellipsis (...) button launches the <i>Class Browser</i> (see <a href="#">Using the Class Browser</a> on page 282).

**Table 78** Array Options

Heading	Option	Description
Dimensions		Specifies the dimensionality of the array; that is, the number of arrays in the array. The default value is 1, which is the usual case.
Dimension		Identifies the array within the primary array.
Size		Specifies the number of elements in the corresponding array.

**Note:** *If you do not specify the size, a placeholder () is entered by default. You can then specify the size at a later time.*

## 10.8 Creating and Modifying Variables

Clicking the **Local Variable** icon in the Business Rules Editor toolbar (see Figure 280) presents the **Create Variable** dialog (see Figure 281).

**Figure 280** Local Variable Icon



**Figure 281** Create Variable Dialog

The various options are summarized in Table 79.

**Table 79** Variable Options

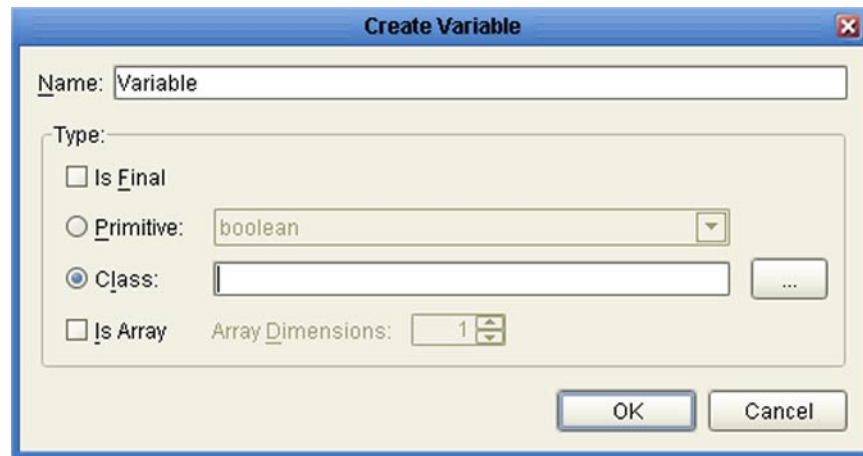
Heading	Option	Description
Variable Name		Your name for the variable.
Type	Is Final	Specifies that the variable is final.
	Primitive	One of the following: <ul style="list-style-type: none"> <li>boolean</li> <li>byte</li> <li>char</li> <li>short</li> <li>int</li> <li>long</li> <li>float</li> <li>double</li> </ul>
	Class	Assigns the variable to a Java class. Clicking the ellipsis (...) button launches the <i>Class Browser</i> (see <a href="#">Using the Class Browser</a> on page 282).
	Is Array	Specifies that the variable is an array. When checked, enables the Array Dimensions list box, where you can set the array dimensions.



To create a variable of a specific class type

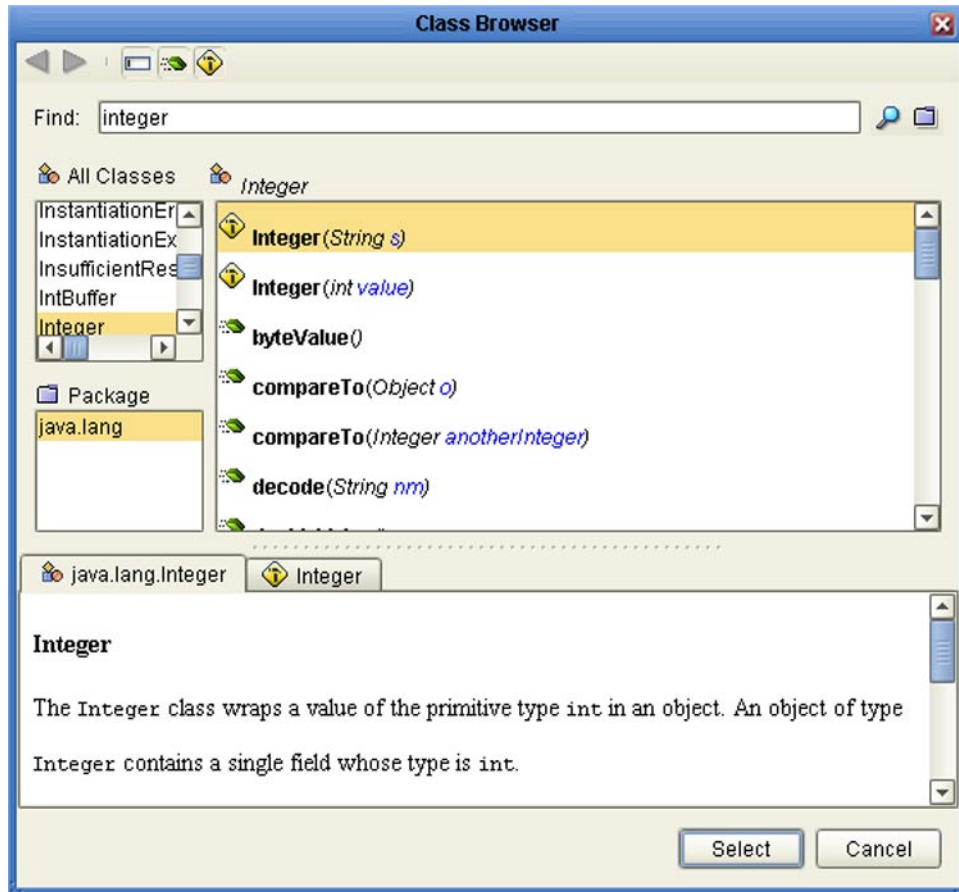
- 1 In the Business Rules editor, click the **Local Variable** button to display the dialog shown in Figure 282.

**Figure 282** Create Variable Dialog

The image shows a 'Create Variable' dialog box with a blue title bar and a standard Windows-style window. The 'Name' field contains the text 'Variable'. Under the 'Type' section, there are four options: 'Is Final' (unchecked checkbox), 'Primitive' (radio button), 'Class' (selected radio button), and 'Is Array' (unchecked checkbox). The 'Primitive' option has a dropdown menu showing 'boolean'. The 'Class' option has an empty text field and an ellipsis (...) button to its right. The 'Is Array' option has an 'Array Dimensions' field with the value '1' and a small up/down arrow. At the bottom right, there are 'OK' and 'Cancel' buttons.

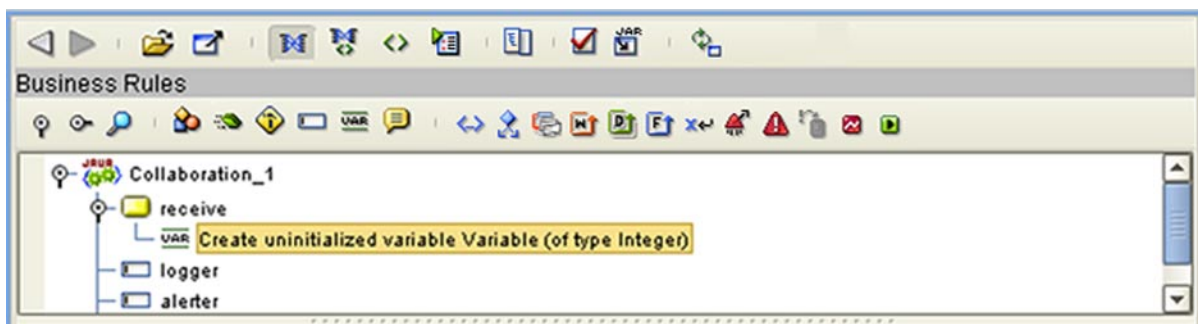
- 2 Enter a name for the variable in the **Name** text box.
- 3 Click either the **Primitive** or **Class** option button to select the variable type.
- 4 If you selected **Class**, click the ellipsis (...) button to display the **Class Browser** dialog shown in Figure 283.

**Figure 283** Class Browser - Select Class



- 5 Locate the desired class (for example, *Integer*) and click the **Select** button to add the variable to the Business Rules editor as shown in Figure 284.

**Figure 284** Business Rules with Variable

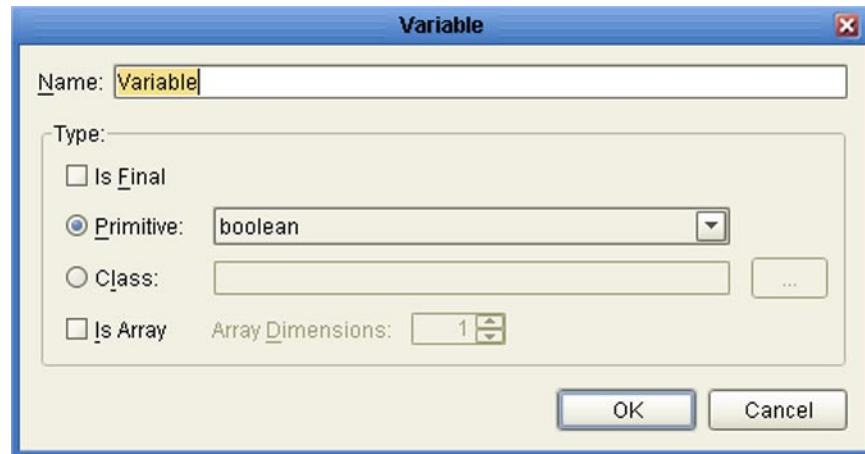


**Note:** Having the variable selected when the constructor is created initializes the variable. This also occurs when a literal is created, for those variables that can have literals assigned.

### To modify an existing variable

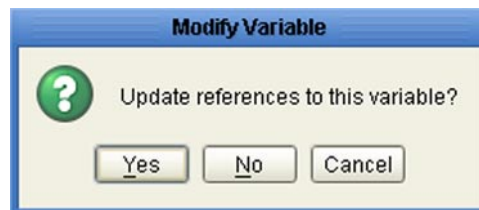
- 1 Right-click on the variable in the OTD tree to display its context menu.
- 2 Select **Open Declaration** to display the *Variable* dialog (see Figure 285).

**Figure 285** Variable Dialog



- 3 Make the desired changes to the variable properties.
- 4 Click **OK** to display the dialog shown in Figure 286.

**Figure 286** Modify Variable Dialog



- 5 Click the appropriate button, as follows:
  - ♦ **Yes** to save your modifications and update all references to the variable.
  - ♦ **No** to save your modifications without updating any references to the variable.
  - ♦ **Cancel** to return to the *Variable* dialog to change or cancel your modifications.

## 10.9 Creating and Modifying Literals

### Using the Toolbar Menus

The Business Rules Designer toolbar menus contain several literal methods, listed in Table 80, which are described in [Graphical Collaboration Methods \(Java\)](#) on page 263. Clicking the menu option places the corresponding method box on the mapping canvas.

**Table 80** Literals in Toolbar Menus

Menu	Literal
Boolean	False
	True
Math	Literal Number
Object	Null
String	Literal Character
	Literal String

### Using the Method Dialog

Launch the method dialog as described in [Using the Method Dialog](#) on page 287 and click the **Create Literal** icon in the toolbar. If you have launched the dialog from an Object, the icon depicts an Object (see Figure 287) and the **Null** method box is placed on the mapping canvas.

**Figure 287** Object Icon



If you have launched the icon from a String, the icon depicts a String (see Figure 288) and the **Literal String** method box is placed on the mapping canvas.

**Figure 288** String Icon



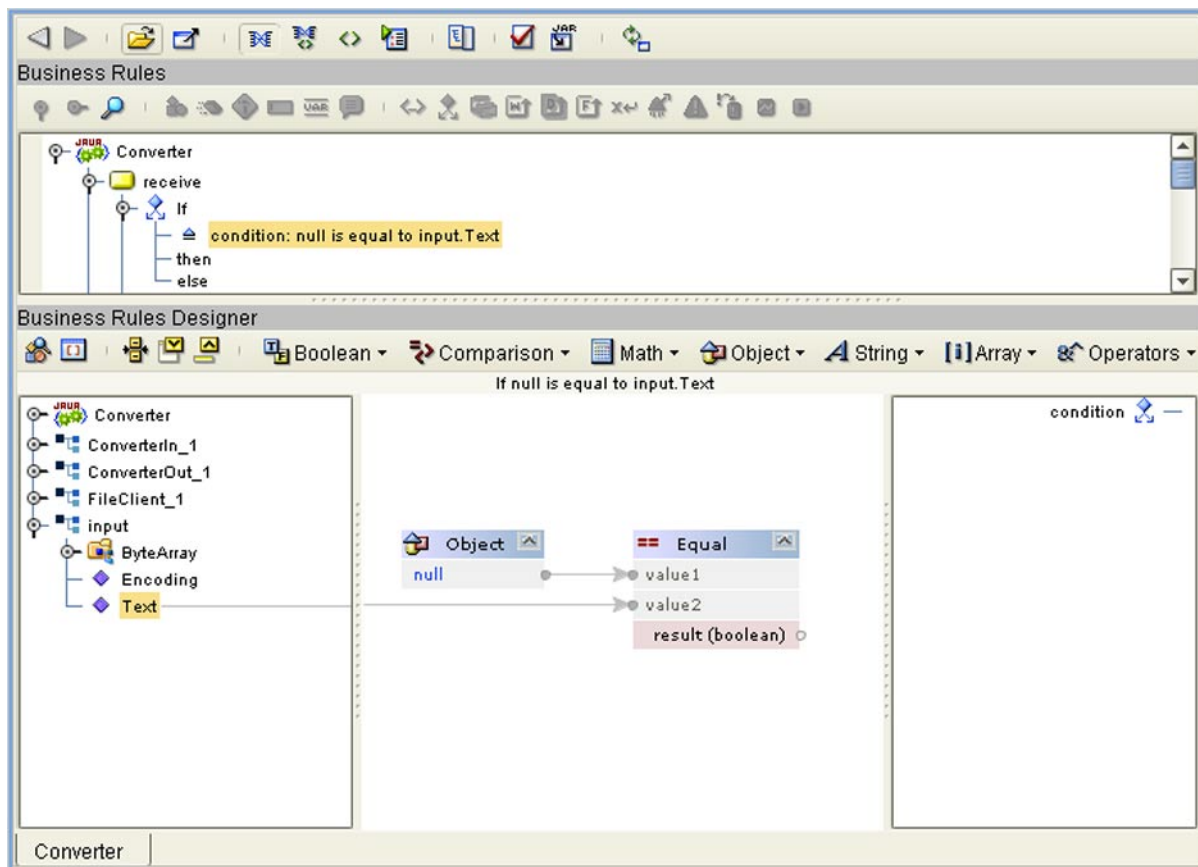
## 10.10 Generating Alerts

If you have *Sun SeeBeyond Alert Agent* installed on your system, you can incorporate alert generation into a Collaboration Definition (Java). As an example, you could perform a test to determine whether or not a file is empty, and raise an alert if it is, as described in this section. See the *Sun SeeBeyond Alert Agent User's Guide* for information.

### To create an alert using the Collaboration Editor (Java)

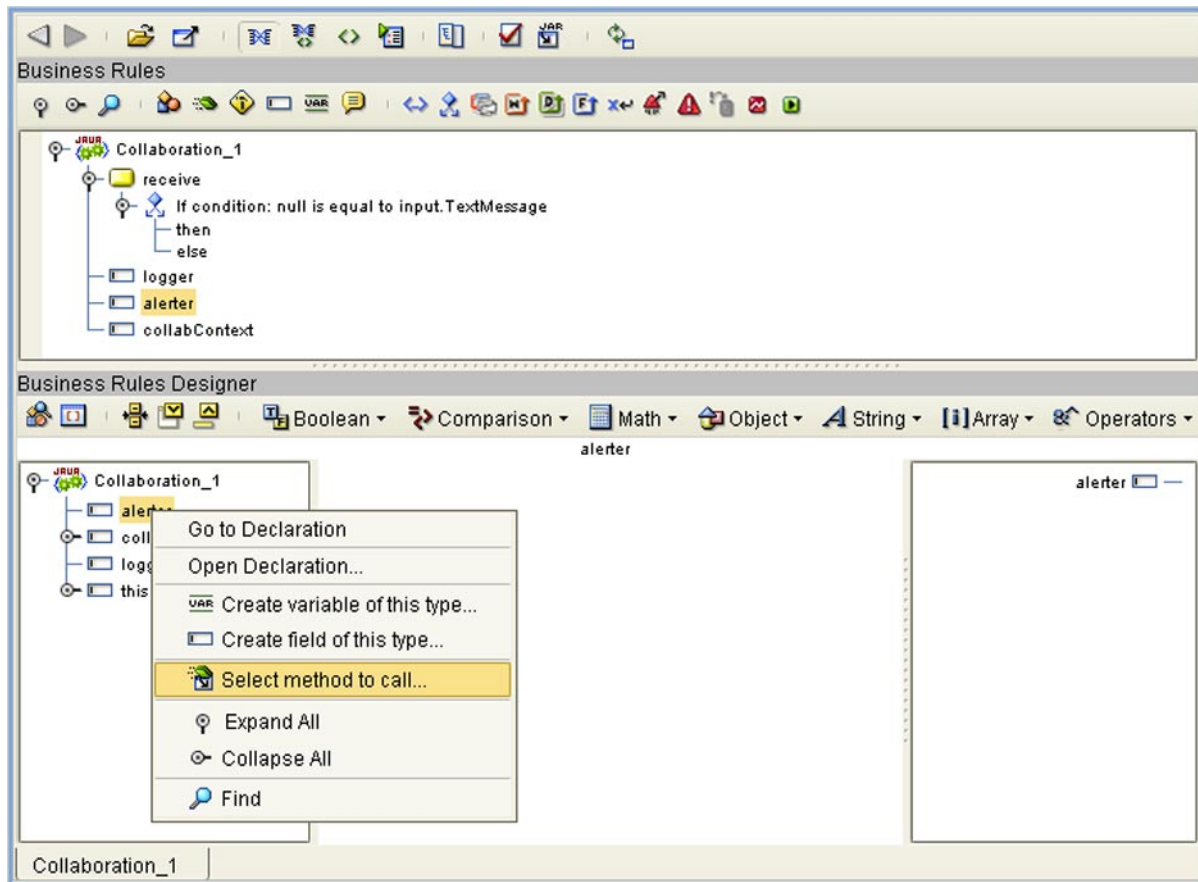
- 1 Add an **If** condition to the appropriate method in the Business Rules editor (see Figure 289).
- 2 Define the rule in the Business Rules Designer by placing an **equal** method box (*Comparison* menu) and a **literal null** method box (*Object* menu) in the mapping pane.
- 3 Link the desired input file to one input port of the **equal** method box, and the literal output port to the other **equal** input port, as shown.

**Figure 289** Empty File Test - If Condition



- 4 Initiate an alert object by clicking the **alerter** node of your Collaboration Definition in the Business Rules editor.
- 5 Right-click the **alerter** node of the Collaboration Definition in the left pane of the Business Rules Designer, which displays the menu shown in Figure 290.

**Figure 290** Empty File Test - Alerter Node Menu



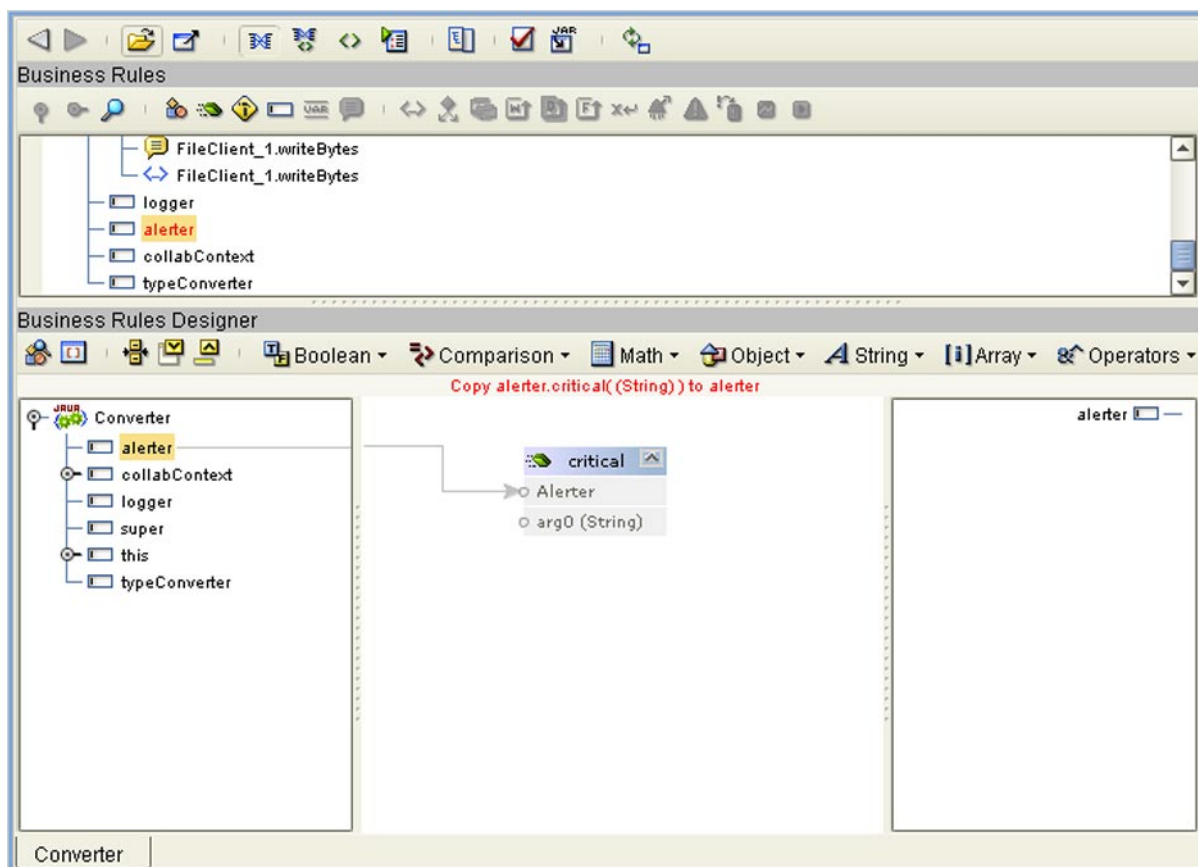
- 6 Click **Select a method to call**, which displays a method list box containing severity levels (see Figure 291).

**Figure 291** Alerter Severity Method List



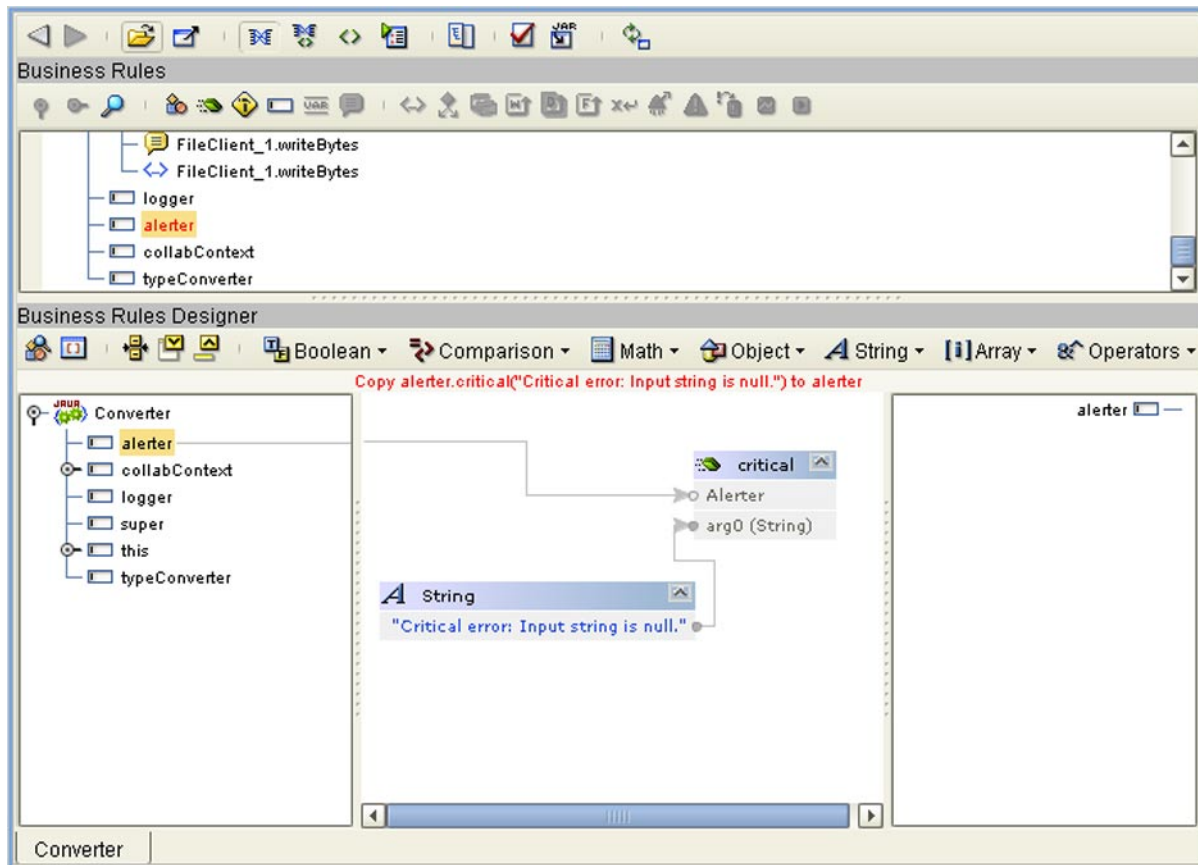
- 7 Select the desired severity for the alert from the method list box; for example, **critical**. A **critical** method box is placed in the mapping panel (see Figure 292).

**Figure 292** Empty File Test - critical Method



- 8 Create your alert message, which can be a literal, a constant, or an OTD field name. As an example, we show a **literal string**.
- 9 Pass the alert message to the alert event by linking the literal message output port to the argument of the alerter object (see Figure 293).

**Figure 293** Empty File Test - Pass Alert Message





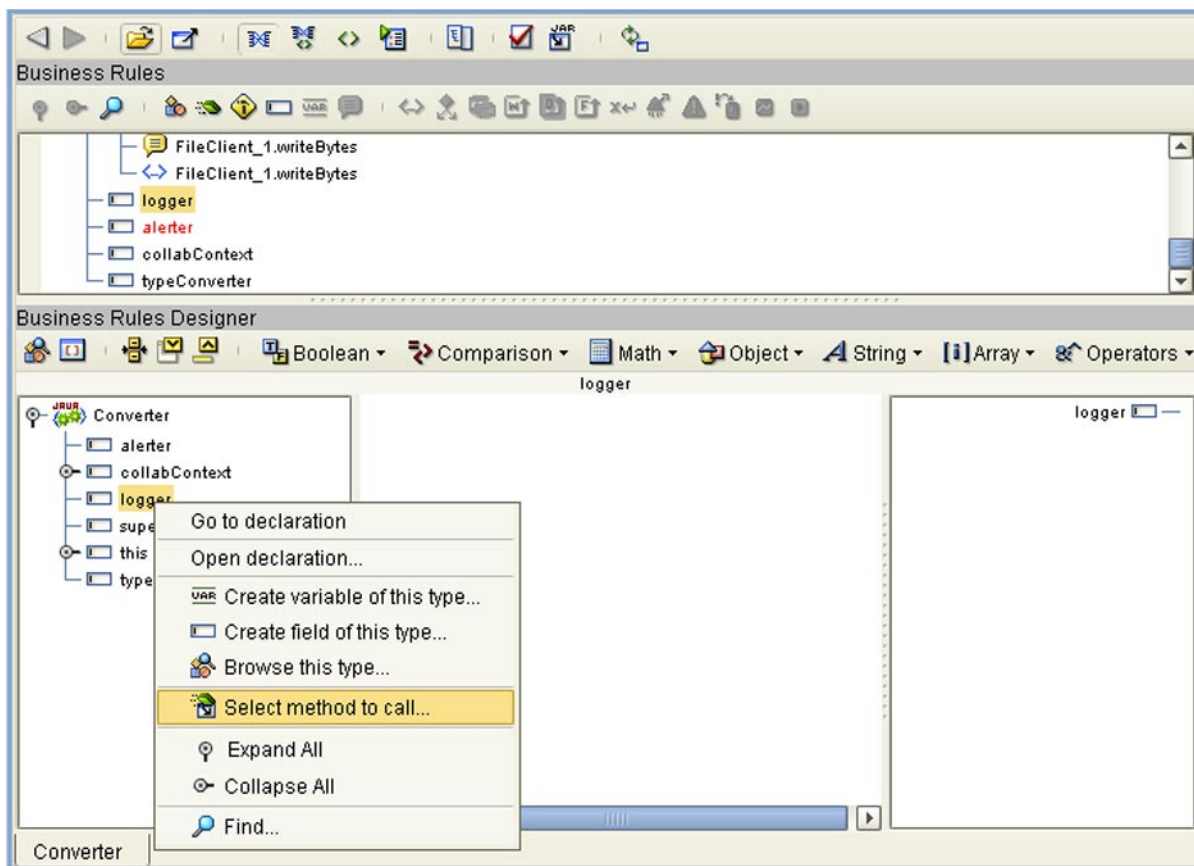
## 10.11 Creating Log Entries

You can initiate log entries from a Collaboration Definition (Java), as described in the following procedure.

To create a log entry using the Collaboration Editor (Java)

- 1 Initiate a logging event by clicking the **logger** node of your Collaboration Definition in the Business Rules editor.
- 2 Right-click the **logger** node for the Collaboration Definition in the left pane of the Business Rules Designer, which displays the menu shown in Figure 294.

**Figure 294** Logging Example - Logger Node Menu



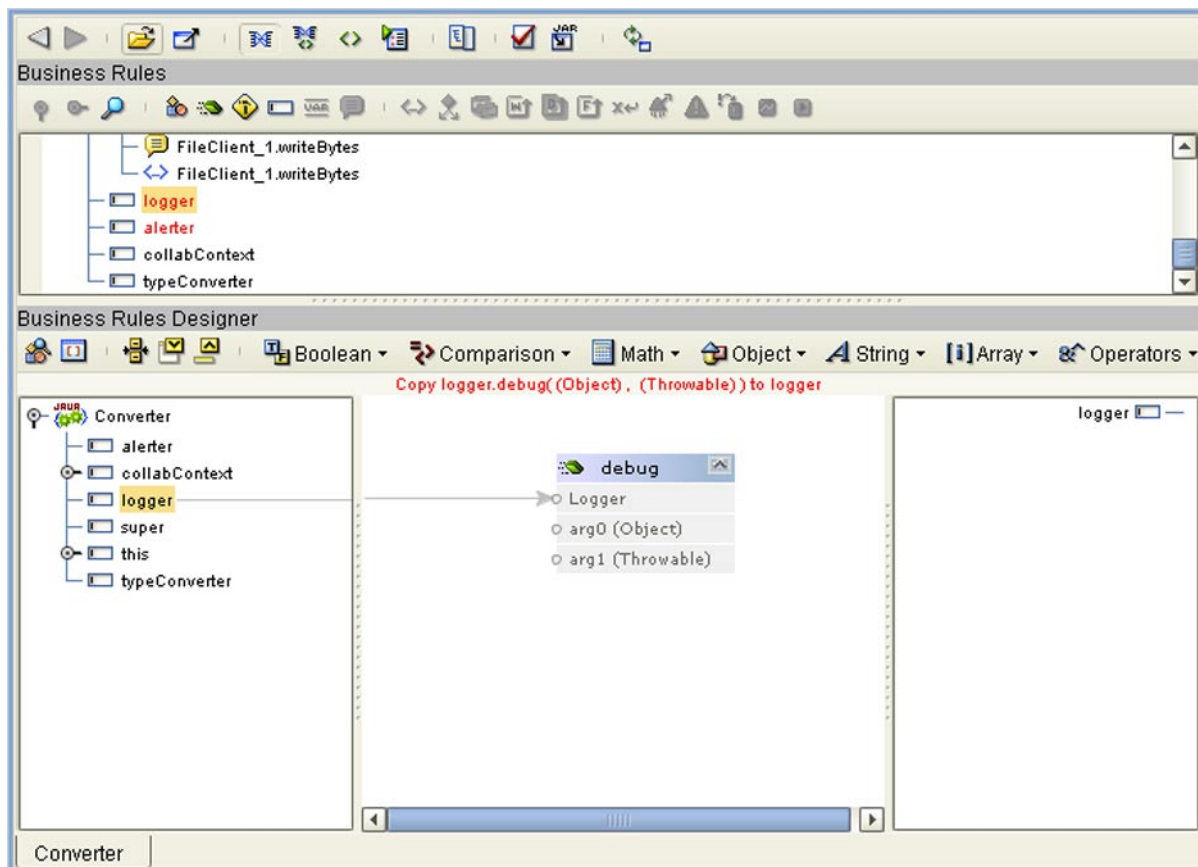
- 3 Click **Select a method to call**, which displays a method list box containing logging levels (see Figure 295).

**Figure 295** Logging Level Method List



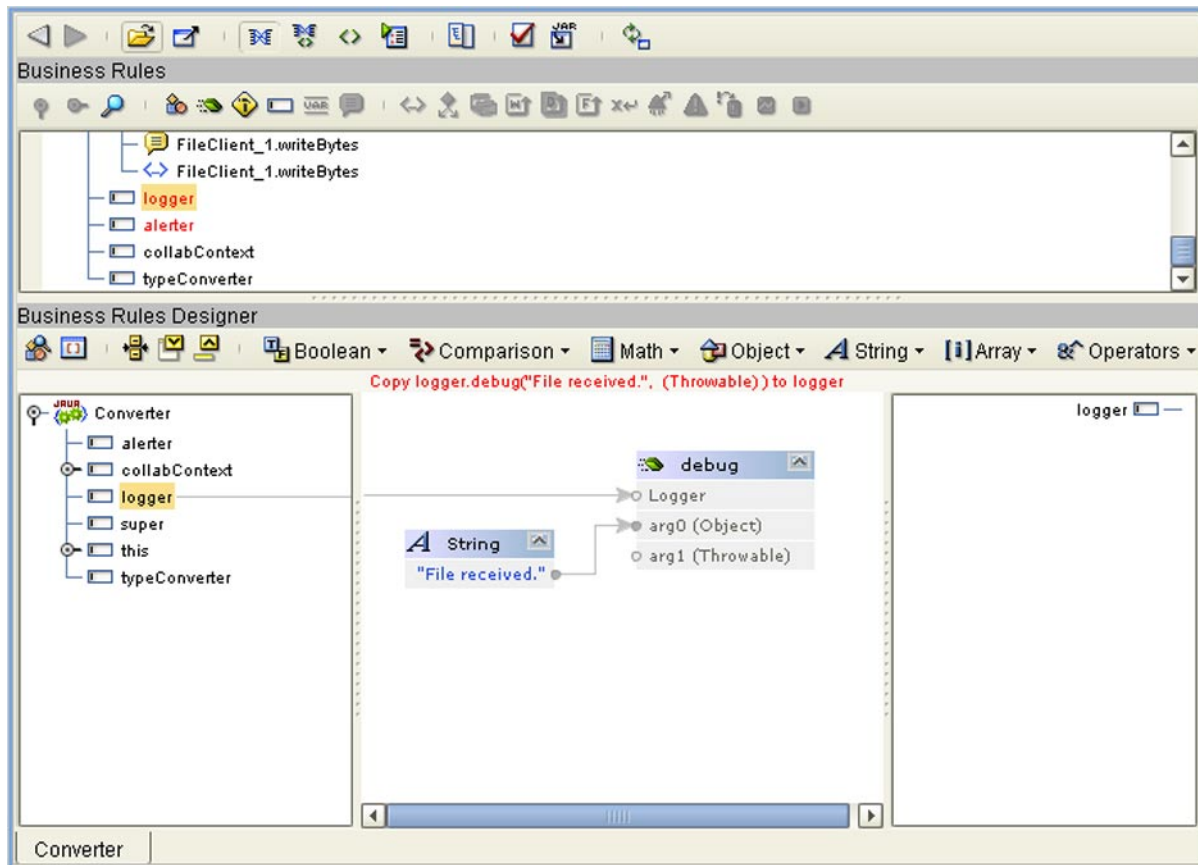
- 4 Select the desired logging level from the method list box; for example, **debug**. A **debug** method box is placed in the mapping panel (see Figure 296).

**Figure 296** Logging Example - debug Method Box



- 5 Create your log message, which can be a literal, a constant, or an OTD field name. As an example, we show a **literal string**.
- 6 Pass the log message to the logging event by dragging the message to the appropriate argument provided by the method box (see Figure 297).

**Figure 297** Logging Example - Pass Log Message



**Note:** Compiler errors are logged in the IDE log (\edesigner\usrdir\system\IDE.log), at the **warning** level. See the *Sun SeeBeyond eGate™ Integrator System Administration Guide* for more information regarding logging level settings.

## 10.12 Importing Third-Party Java Classes

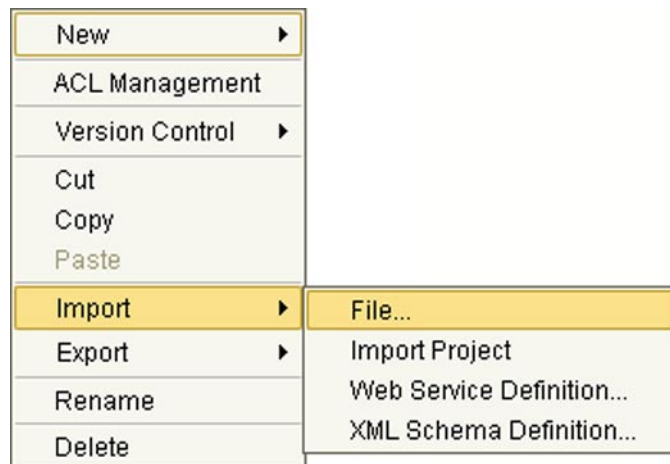
Adding third-party Java classes (.jar files) to a Collaboration Definition is a two-step process:

- 1 First, you must import the Java classes into your Project.
- 2 Second, you must add the Java classes to the specific Collaboration Definition.

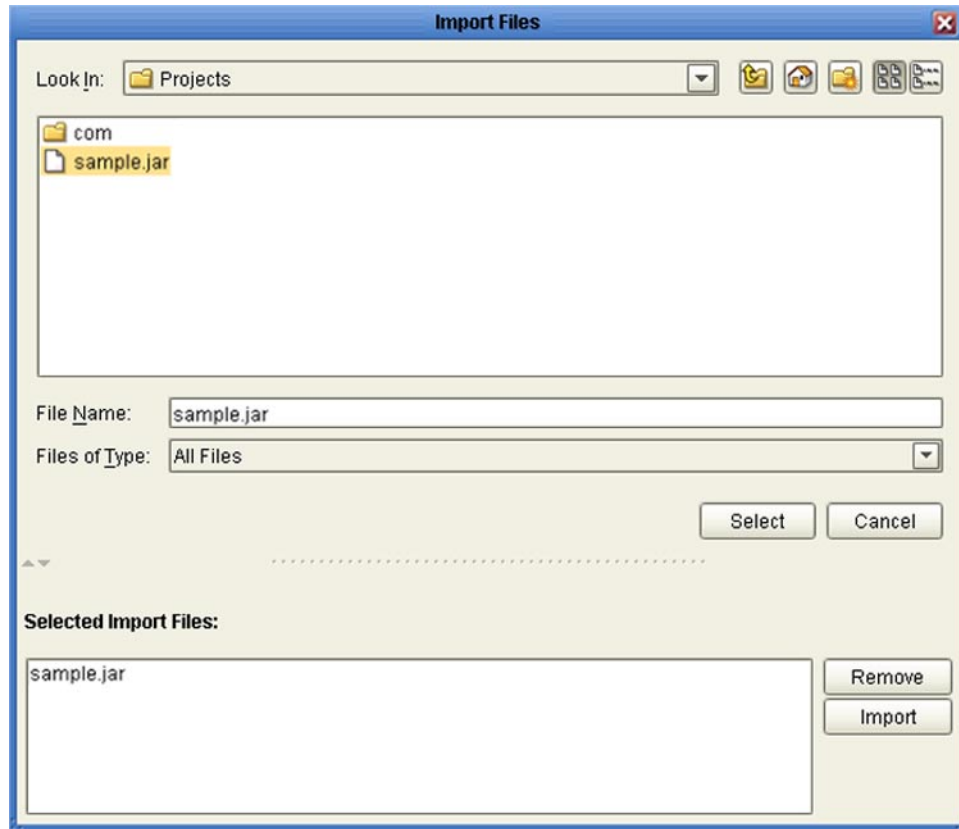
### To import third-party Java classes into a Project

- 1 Copy the third-party files to a convenient directory.
- 2 Right-click on a **Project** in the Project Explorer to display its context menu.
- 3 Select **Import > File** in the context menu (see Figure 298) to display the file import dialog (see Figure 299).

**Figure 298** Project Context Menu: Import File



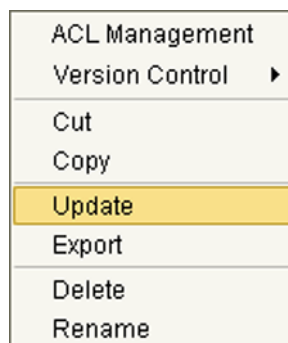
**Figure 299** File Import Dialog



- 4 Locate the desired **.jar** files and click **Select** for each.
- 5 When you have selected all desired files, click **Import** to import the entire list into the Project.
- 6 Add the Java classes to your Collaboration Definition following the procedure described in [Adding Java classes to Collaboration Definitions](#) on page 354.

**Note:** If the **.jar** file changes, you should update it using the **Update** command from the File context menu, rather than deleting and recreating or importing it (see Figure 300).

**Figure 300** File Context Menu - Update



## 10.13 Adding Java classes to Collaboration Definitions

You can use the Collaboration Editor to add Java classes to your Collaboration Definitions. These Java classes can be from within the current Project, or from another Project (such as the Sun SeeBeyond Project folder, for example).

To add Java classes to a Collaboration Definition:

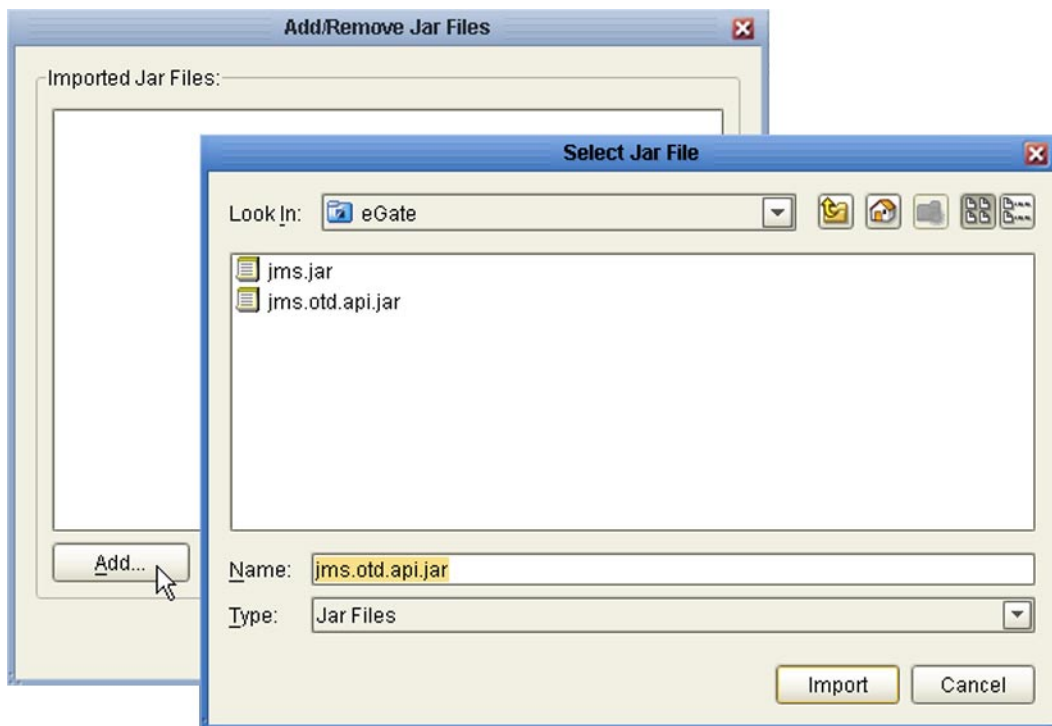
- 1 Open the Collaboration to which you want to add Java classes.
- 2 Click the **JAR File** icon in the main Collaboration Editor toolbar (see Figure 301), which presents the *Add/Remove Jar Files* dialog.

Figure 301 JAR File Icon



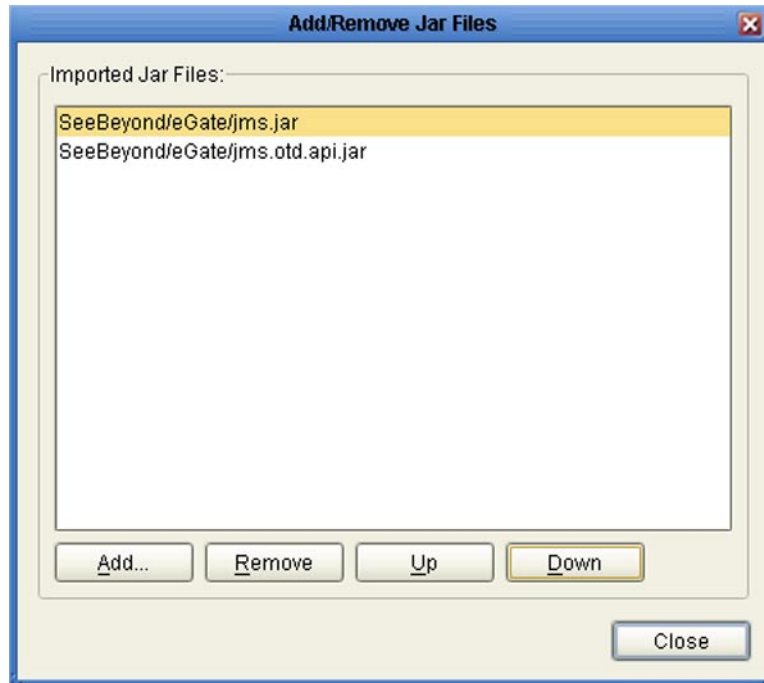
- 3 Click **Add** in the dialog to display a file browser (see Figure 302).
- 4 Locate the desired **.jar** files and click **Import** for each.

Figure 302 Add/Remove Jar File Dialog (1)



- 5 When all the **.jar** files are listed in the *Add/Remove Jar Files* dialog (see Figure 303), click **Close**.

**Figure 303** Add/Remove Jar File Dialog (2)



- ♦ You can also delete **.jar** files by selecting the files and clicking **Remove**, and change the order of the files by using the **Up** and **Down** buttons (files are searched in order, as in the *classpath*).

## 10.14 Copying OTD Nodes

Nodes can be copied from one OTD to another by using the **copyNode()** method contained in class **OtdUtil**. This method performs a node-to-node copy between messagable OTDs when the data types of the fields under the nodes are an exact match.

### copyNode()

#### Syntax

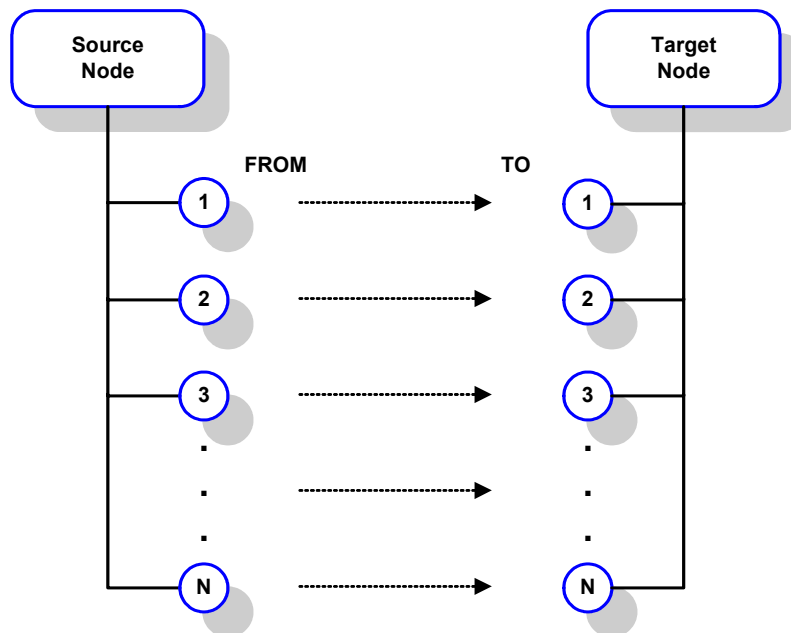
```
public static void copyNode (Object src, Object trg)
```

where src = source node and trg = target node.

#### Operation

The **copyNode** method iteratively cycles through the node structure from top to bottom, checking each descendant node according to specific rules before attempting to copy it. Figure 304 illustrates the process, where the method begins with child 1 in the source node, which it attempts to copy to child 1 in the target node. Once resolved, it proceeds to child 2, then 3, and on to child N. Basic operational rules are outlined following the figure.

**Figure 304** copyNode Operation



#### Operational Rules

**Note:** In the following, the nodes described are descendants of the node on which the **copyNode** method is called.

- If the selected source node cannot be read, the node is skipped and the next source node is selected.



- If the selected source node can be read, but the corresponding target node cannot be written to, the target node is skipped.
- If the source and target are structurally equivalent, but any source node is absent from the instance, the corresponding target node is unchanged.
- If the selected source node is a **leaf** node, and:
  - A The selected target node is *not* a leaf node, the node is skipped and the next source node is selected.
  - B The selected target node *is* a leaf node, but of a different Java Type from the source node, the node is skipped and the next source node is selected.
  - C The selected source node is present, *not* optional, and its parent node is *not* a choice node, it is copied to the target node according to the following matrix:

		Source OTD	
		Single Node	Repeating Node
Target OTD	Single Node	Copy source node to target node.	Copy first source node to target node, if count is greater than zero.
	Repeating Node	Copy source node to first target node.	Copy all source nodes to corresponding target nodes.

- If the selected source node is a **parent** node, and:
  - A The selected target node is *not* a parent node, the node is skipped and the next source node is selected.
  - B The selected target node *is* a parent node, the source node is copied to the target node recursively according to the matrix shown in case C, above.

#### To use the copyNode method

- 1 Import the file **OtdUtil.jar** using the procedure described in [Adding Java classes to Collaboration Definitions](#) on page 354. The file is located in the **Sun SeeBeyond > eGate** Project folder in Project Explorer.
- 2 In the source OTD tree displayed in the Java Collaboration Editor, right-click the node you want to copy and select the Class Browser.
- 3 Search for the **OtdUtil** class and select the **copyNode** method. The **copyNode** method box will appear in the editor's mapping panel.
- 4 Connect the output node of the method box to the appropriate target node in the destination OTD.
- 5 Click **Save**.

## 10.15 Using Try-Catch

Clicking the **try** icon in the toolbar adds a **try** statement to the Business Rules tree, initiating a number of programming statements that are monitored to see whether they succeed or fail. A **finally** statement is added automatically. To perform a **try-catch**, you must add the **catch** statement manually as described in the following procedure.

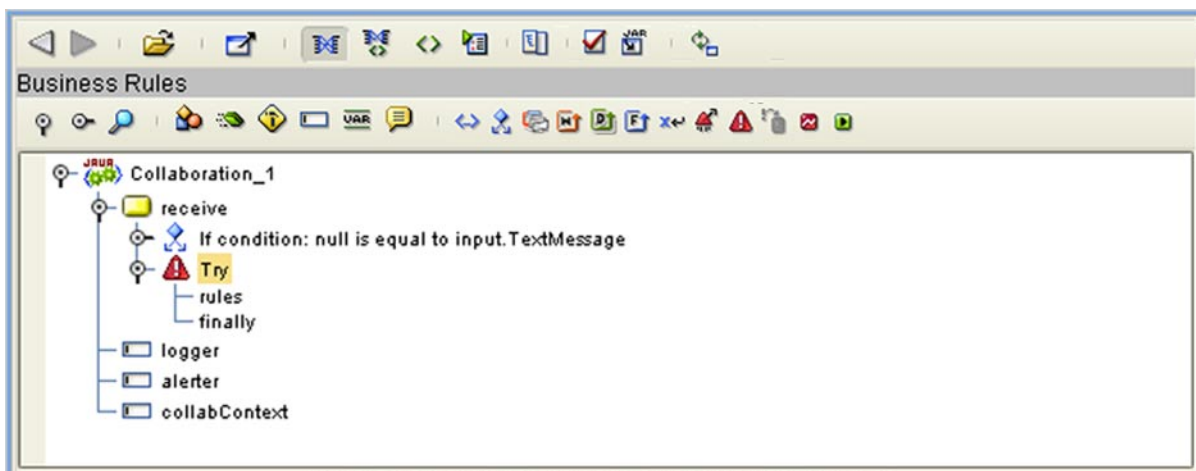
To create a try-catch operation

- 1 Click the **Try** icon in the toolbar (see Figure 305), which adds the **try** statement to the business rules tree (see Figure 306). It also enables the **Catch** icon in the Business Rules editor toolbar.

Figure 305 Try Icon



Figure 306 Try Statement

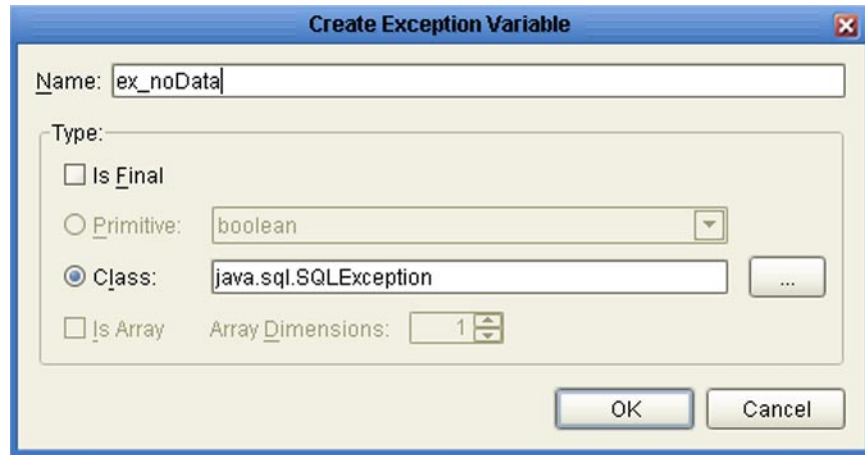


- 2 Select the **Catch** icon (see Figure 307), which presents the **Create Exception Variable** dialog (see Figure 308).

Figure 307 Catch Icon

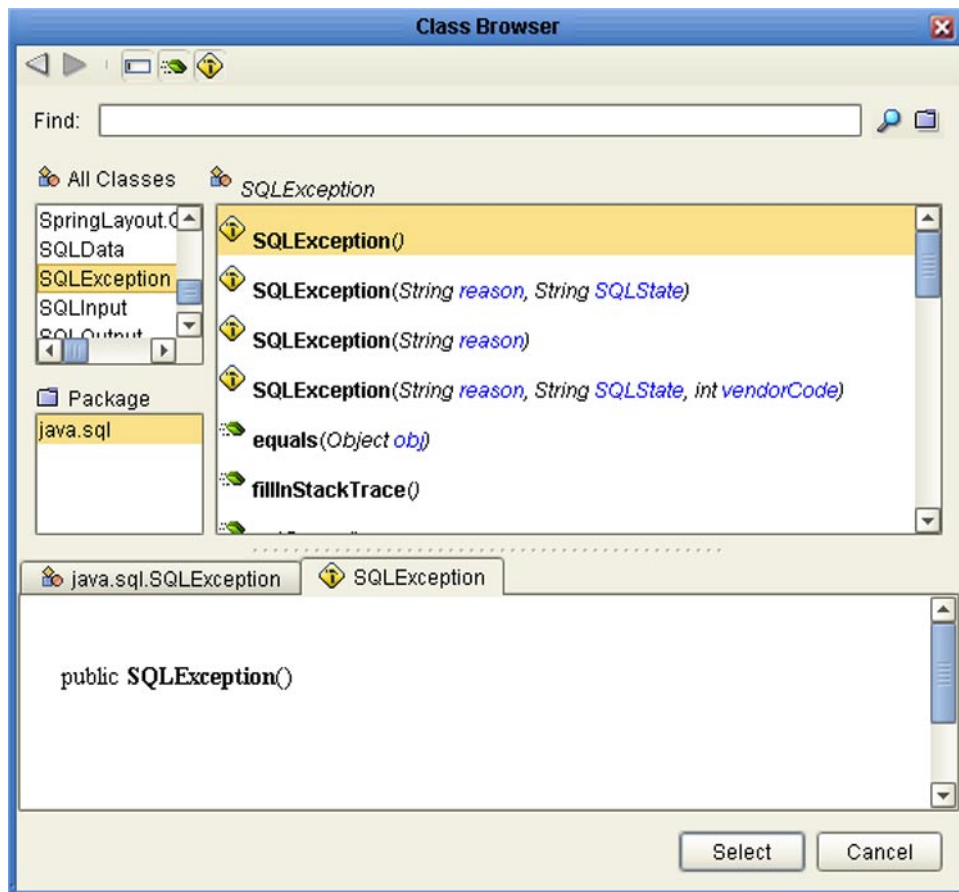


**Figure 308** Create Exception Variable Dialog



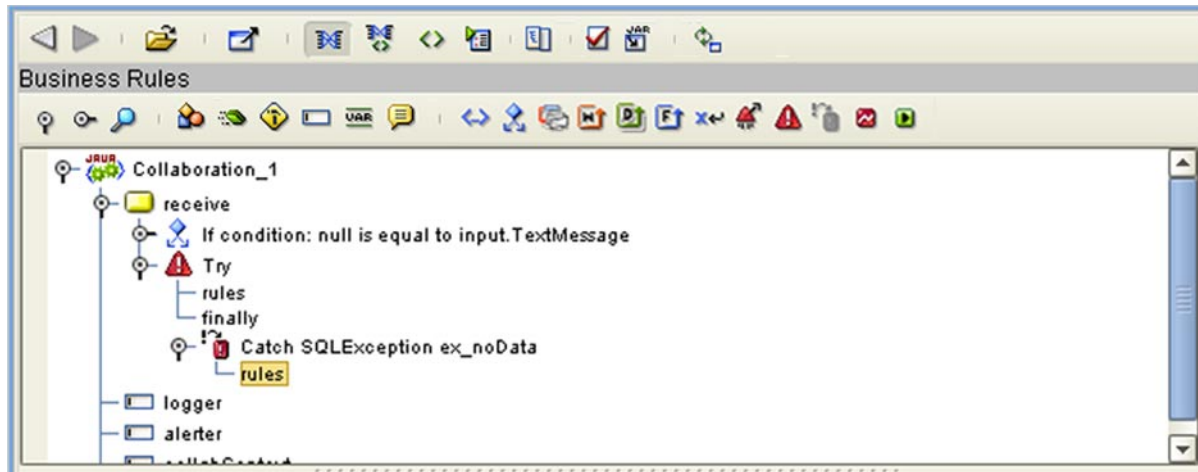
- 3 Enter a name for the exception, for example: **ex\_noData**.
- 4 Click the ellipsis (...) button to invoke the *Class Browser* shown in Figure 309.

**Figure 309** Class Browser Dialog



- 5 To specify that the exception represents a database error, for example, select **SQLException** and click **Select**.
- 6 The **catch** statement is now added to the business rules tree (see Figure 310).

**Figure 310** Catch Statement



- 7 The rule shown in Figure 310 can contain a literal representing the message that will be assigned to an outbound element when the rule is executed.

## 10.16 Using Byte-Array Converters

Several data-encoding converters are available for byte arrays (byte[]s) within specific versions of eGate Integrator. To access these converters, you must enable the extended language options as described in [Setting Up Options](#) on page 53.

**Note:** *Byte-array converters are currently available only when using either Japanese or Korean localized versions of eGate Integrator.*

For Japanese encoding, these converters automatically translate data between Shift-JIS (SJIS) character encoding (the Japanese native code for Java) and the following encoding methods used by mainframe computers, both with and without Gaiji:

- EBCDIC-J (for IBM mainframes)
- JEF (for Fujitsu mainframes)
- JIPSE (for NEC mainframes)
- KEIS (for Hitachi mainframes)

**Table 81** Byte-Array Encoding Converters - Japanese

Converter Designation	Converts	
	From	To
SJIS->EBCDIC-J	SJIS	EBCDIC-J
SJIS->JEF	SJIS	JEF
SJIS->JIPSE	SJIS	JIPSE
SJIS->KEIS	SJIS	KEIS
EBCDIC-J->SJIS	EBCDIC-J	SJIS
JEF->SJIS	JEF	SJIS
JIPSE->SJIS	JIPSE	SJIS
KEIS->SJIS	KEIS	SJIS

For Korean encoding, these converters automatically translate data between EUC-KR and EBCDIC-K.

**Table 82** Byte-Array Encoding Converters - Korean

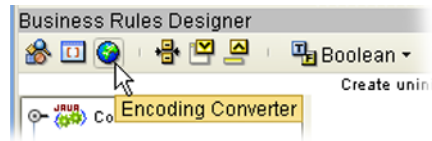
Converter Designation	Converts	
	From	To
EUC-KR->EBCDIC-K	EUC-KR	EBCDIC-K
EBCDIC-K->EUC-KR	EBCDIC-K	EUC-KR

## 10.16.1 Defining Encoding Converter Methods

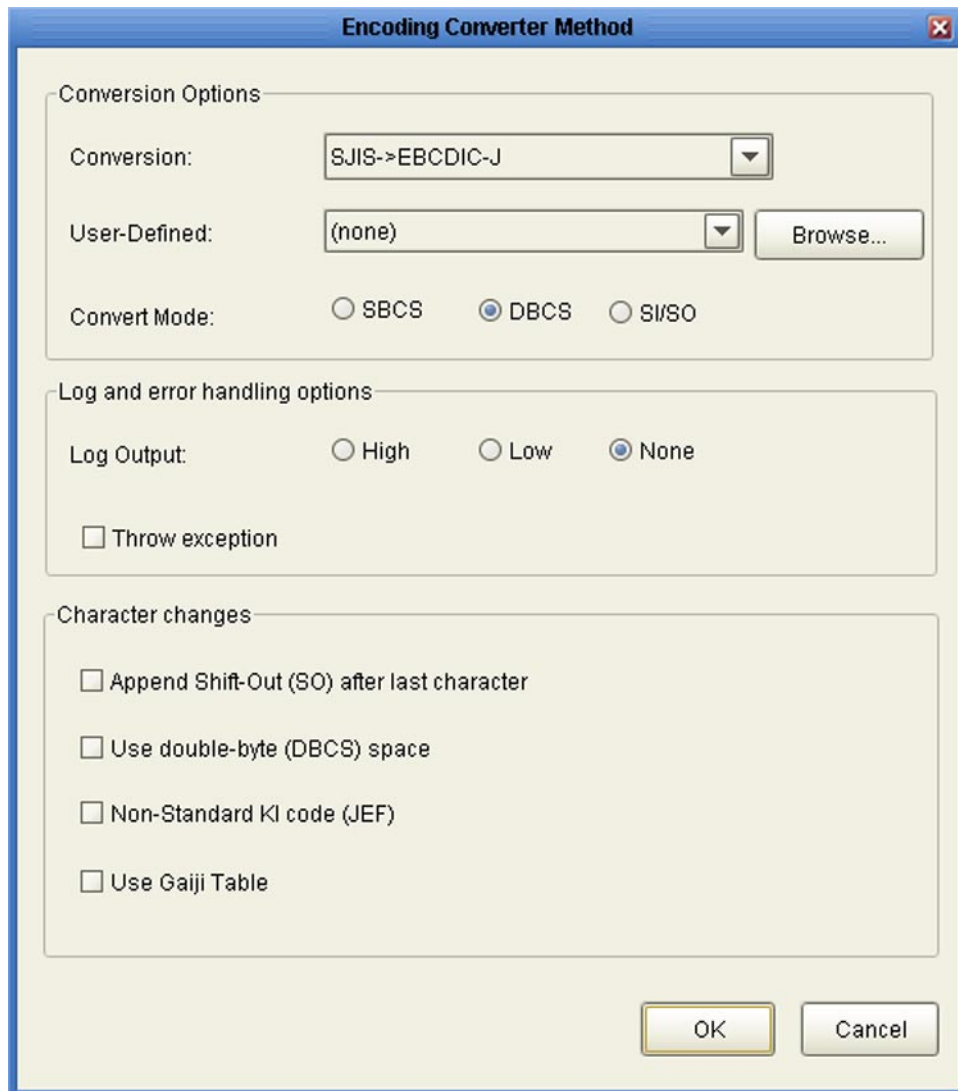
You can specify encoding converter methods by clicking the **Encoding Converter** icon in the Business Rules Designer toolbar (see Figure 311), which displays the dialog shown in Figure 312. The options in the dialog are described in Table 83.

*Note:* This icon is displayed only in the Japanese and Korean versions of eGate Integrator.

**Figure 311** Encoding Converter Icon



**Figure 312** Encoding Converter Methods Dialog



**Table 83** Encoding Converter Methods Dialog Options

Option	Description
Conversion	Specifies the type of byte-array encoding converter to use (see Table 81 and Table 82 for list of converters). Applies to all converters.
User-Defined	Specifies the location of a user-defined mapping table, in which exceptions to the standard mapping can be entered. This mapping table is checked first when performing a conversion, and its contents will override the conversion logic of the standard conversion specified above. If the specific character is not found in this table, then the standard conversion mapping will be used. The format of this table is an ASCII text file with the following characteristics: <ul style="list-style-type: none"> <li>▪ # = comment.</li> <li>▪ one line = one record.</li> <li>▪ one record contains a source code point in HEX format, followed by a blank space, followed by a destination code point in HEX format.</li> <li>▪ HEX format can be with or without <b>0x</b>.</li> </ul> Applies to all converters except those for EBCDIC-J and EBCDIC-KR.
Convert Mode	Specifies whether the field is SBCS only, DBCS only (with no SI/SO characters), or mixed with SI/SO characters. Applies to all converters.
Log Output	Specifies the logging level to use during conversion. Applies to all converters except those for EBCDIC-J and EBCDIC-KR.
Throw Exception	If checked, converter will throw an exception when characters are out of range of SJIS. If unchecked, it will convert to a question mark (?), consistent with the Java specification. Applies to all converters except those for EBCDIC-J and EBCDIC-KR.
Append Shift-Out after last character	If checked, converter appends the last SO character when the data ends with a double byte character. Applies to all SJIS->xxx converters except SJIS->EBCDIC-J when the SI/SO option is selected for Convert Mode.
Use double-byte (DBCS) space	If checked, converts two single byte spaces into one double byte space. Applies to all xxx->SJIS converters except EBCDIC-J->SJIS when either the DBCS or SI/SO option is selected for Convert Mode.

**Table 83** Encoding Converter Methods Dialog Options

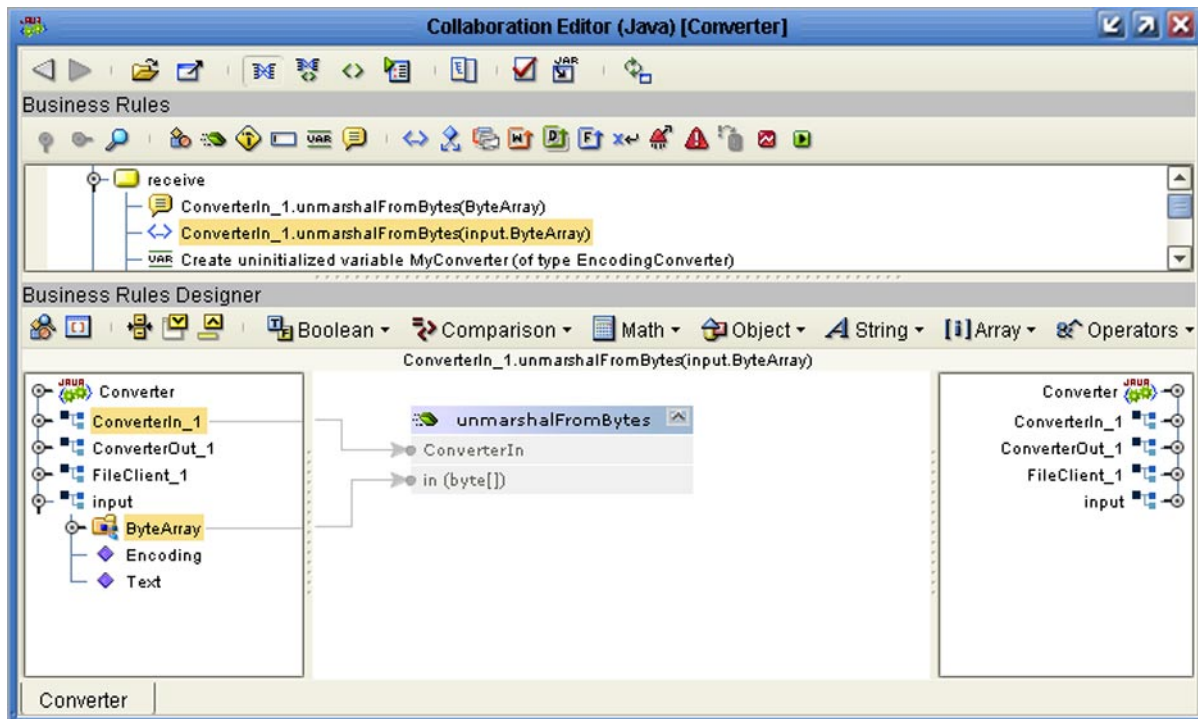
Option	Description
Non-Standard KI code (JEF)	If checked, uses 0x38 instead of standard 0x28 as the KI code. Applies only to JEF converters when the SI/SO option is selected for Convert Mode.
Use Gaiji Table	If checked, converter will use the User-Defined mapping table specified above. Applies to all converters except those for EBCDIC-J and EBCDIC-KR.



### To use a byte-array encoding converter in a Collaboration

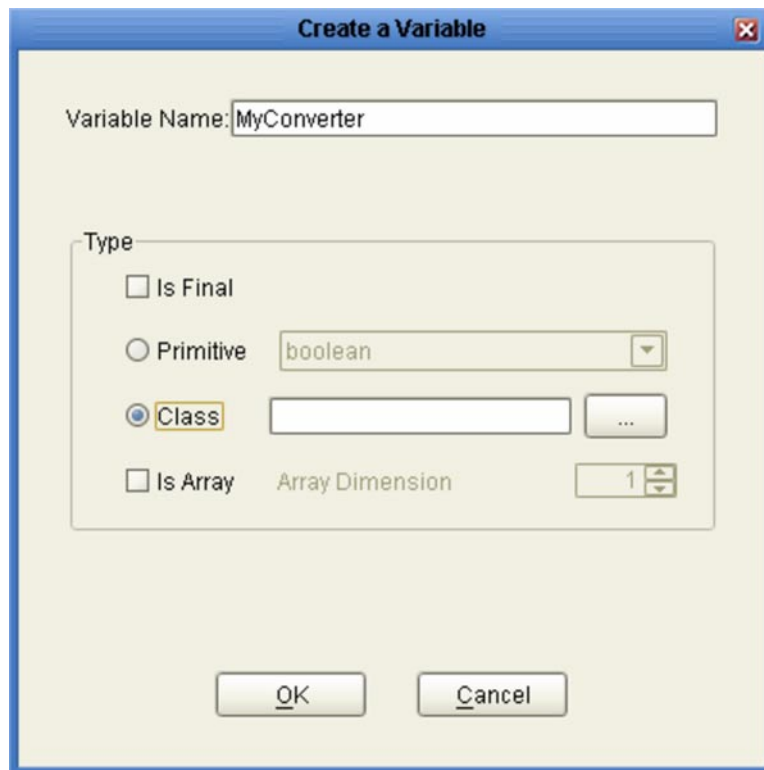
- 1 Enable the *extended language options* as described in [Setting Up Options](#) on page 53.
- 2 Define your business rules to receive the input data as a byte array, using the **unmarshalFromBytes** Java method mapped as shown in Figure 313.

**Figure 313** Using the unmarshalFromBytes Method



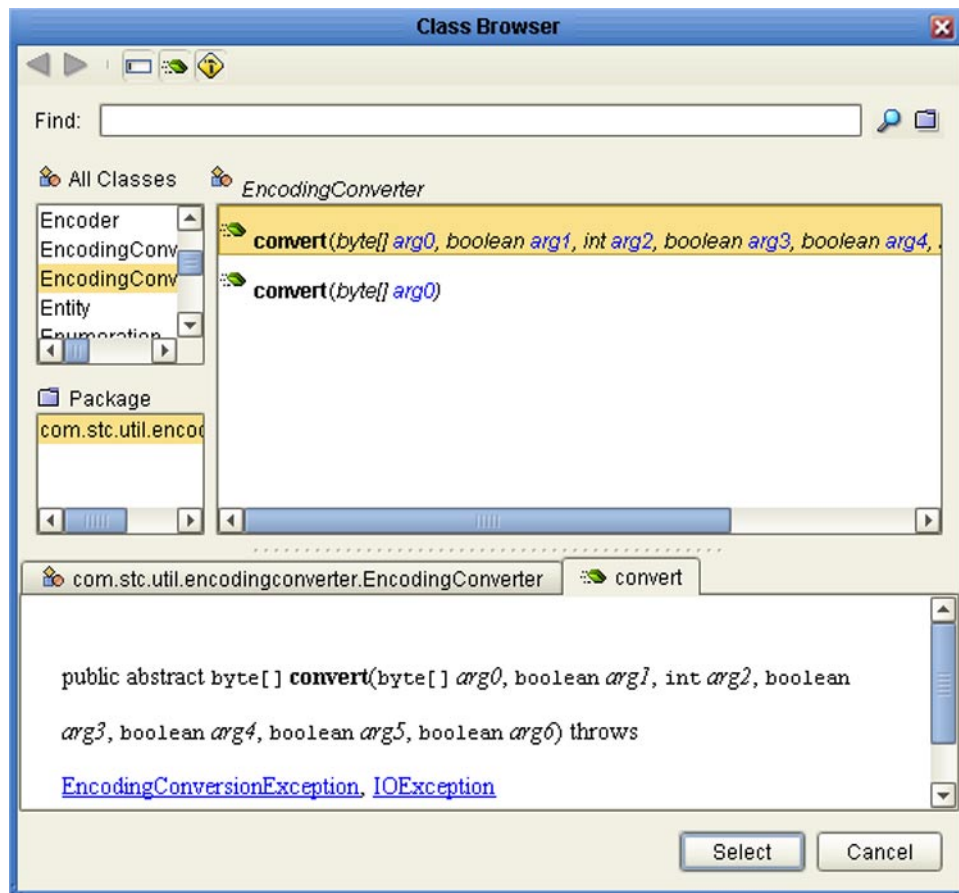
- 3 Create a local variable of the Encoding Converter class.
  - A Click the **Local Variable** icon to display the *Create a Variable* dialog (see Figure 314).

**Figure 314** Create a Variable Dialog



- B** Enter a name for the variable, for example: **MyConverter**.
- C** Click the **Class** option button to enable the entry field.
- D** Click the ellipsis (...) button to display the *Class Browser* (see Figure 315).

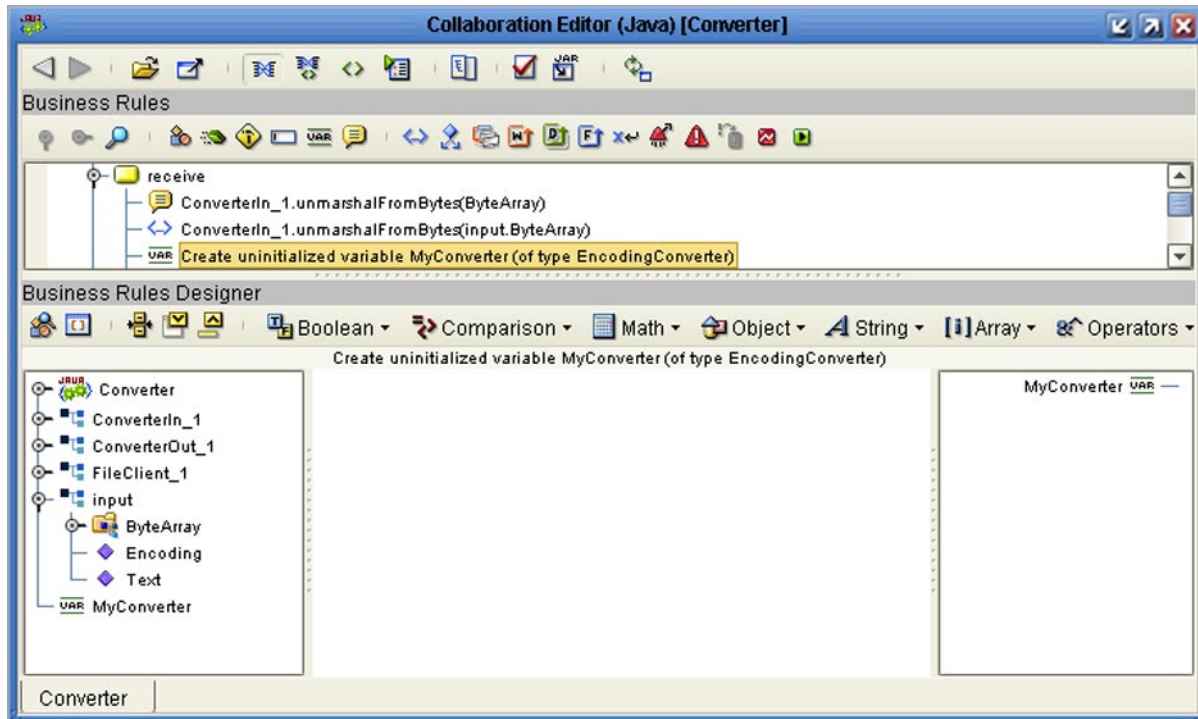
**Figure 315** Class Browser Dialog



- E Scroll to **Encoding Converter** and select it.
- F Click **Select** to return to the **Create a Variable** dialog.

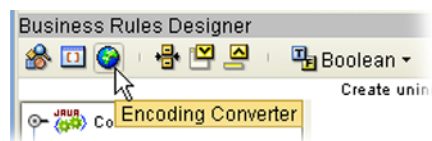
- G Click **OK** again to return to the main Collaboration Editor interface, where the local variable is now displayed (see Figure 316).

**Figure 316** Adding the MyConverter Local Variable



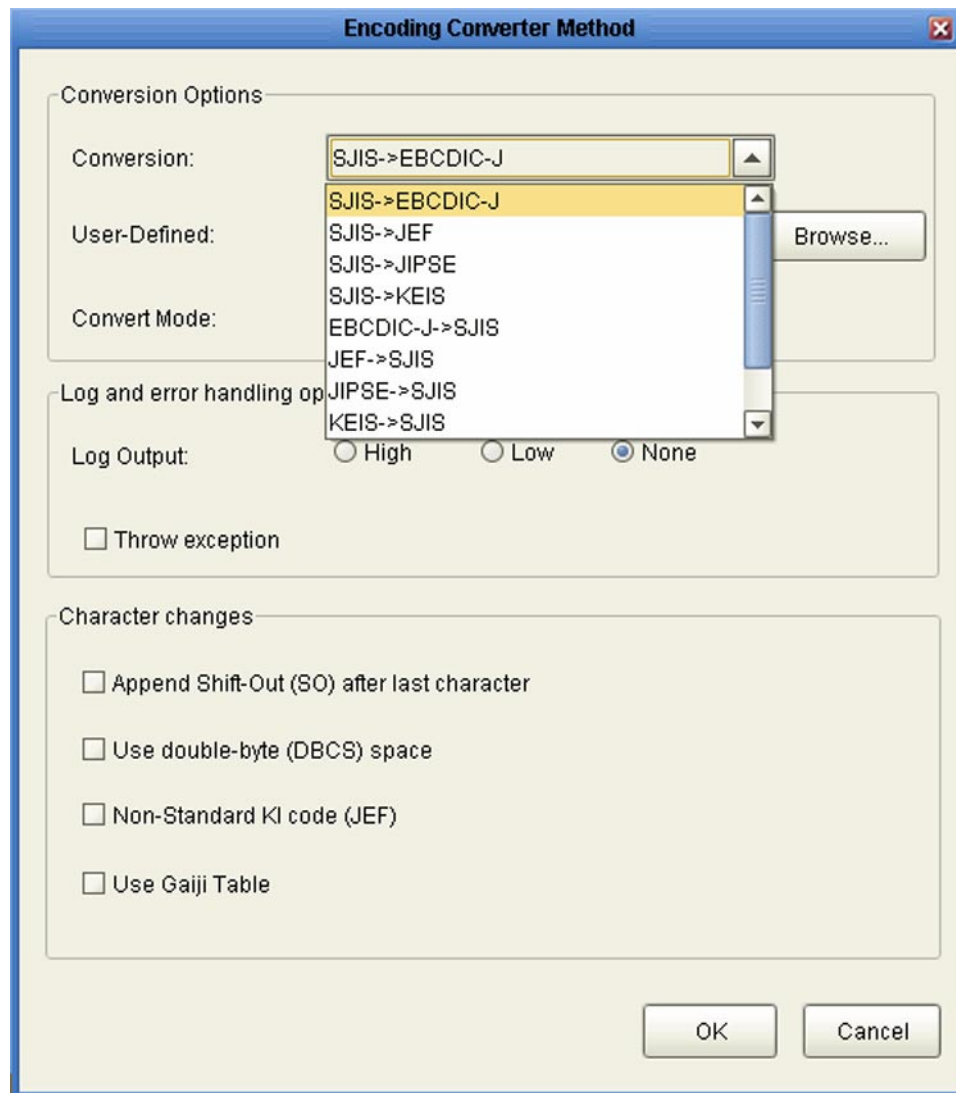
- 4 Click the **Encoding Converter** icon (see Figure 317) to display the *Encoding Converter Method* dialog shown in Figure 318.

**Figure 317** Encoding Converter Icon



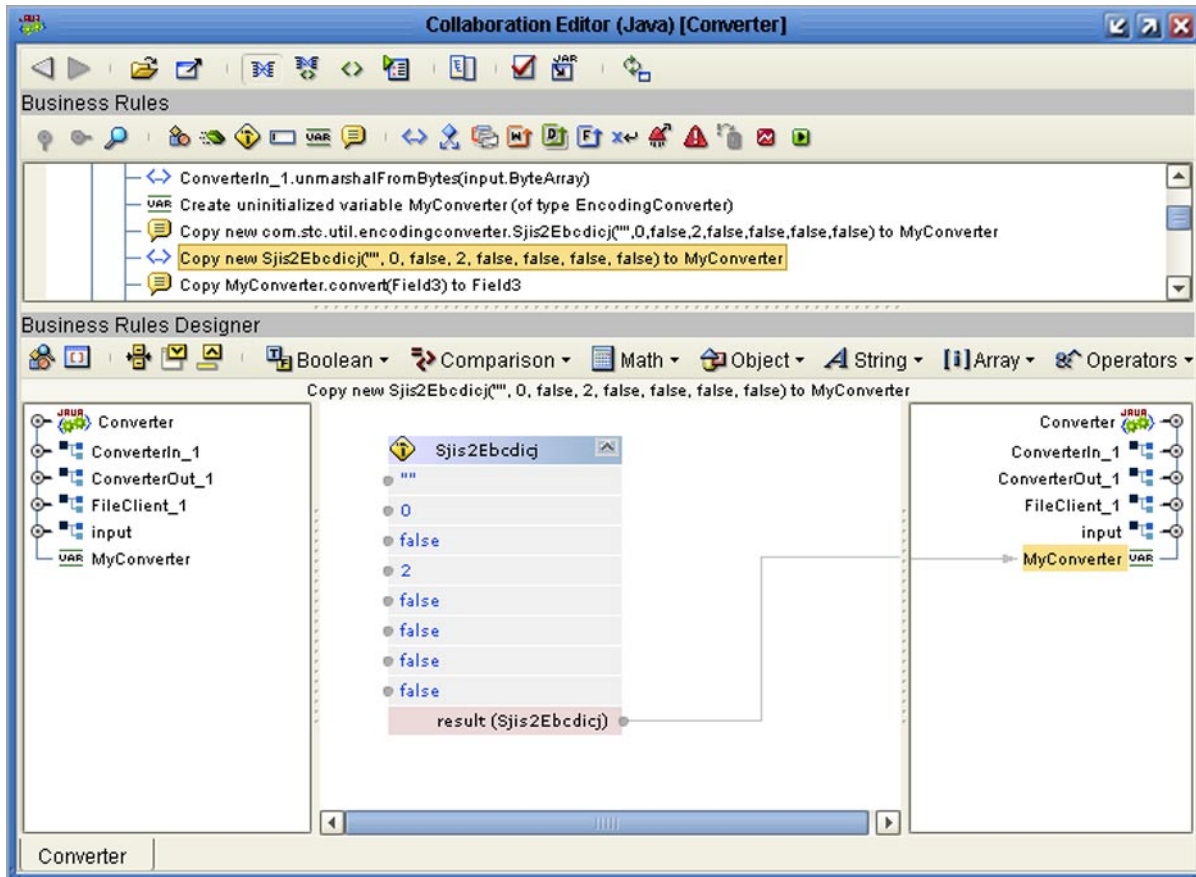
**Note:** This icon is displayed only in the Japanese and Korean versions of eGate Integrator, and is not shown in the other screen captures in this section.

**Figure 318** Encoding Converter Method Dialog



- 5 From the Conversion Options group box, select **SJIS->EBCDIC-J**, as shown in Figure 318, which creates the encoding converter rule shown in Figure 319.
- 6 Select any desired options from the other group boxes and click **OK**.

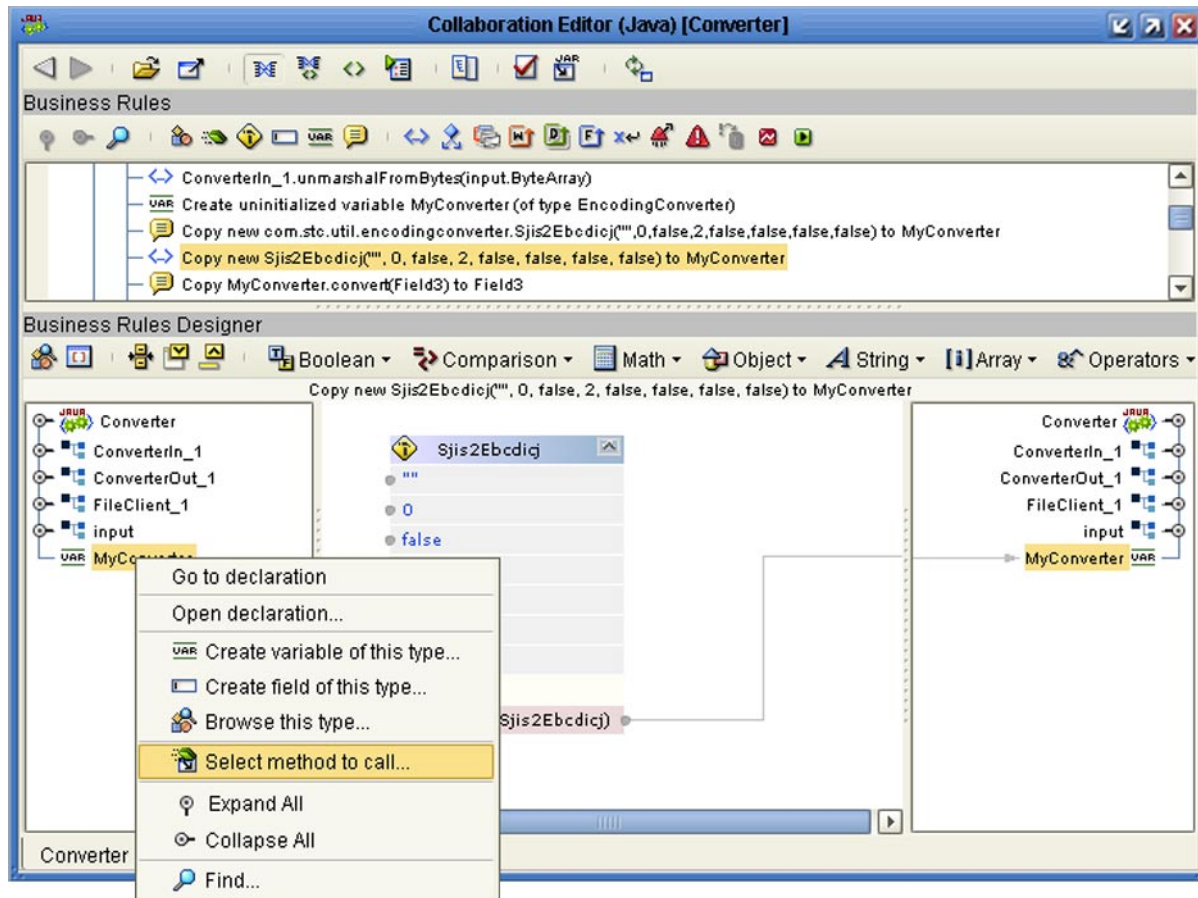
**Figure 319** Creating the Encoding Converter Rule



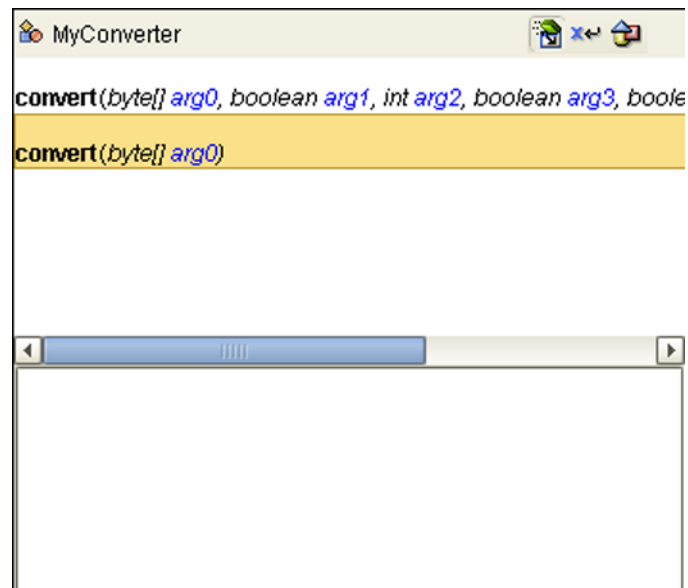
- 7 Map the result to the local variable created in step 3 (**MyConverter**) in the right pane as shown in Figure 319.

- 8 Click on the variable **MyConverter** in the left pane to display its context menu and click **Select a method to call** (see Figure 320). This presents the dialog shown in Figure 321.

Figure 320 Selecting a Method



**Figure 321** Convert Method List

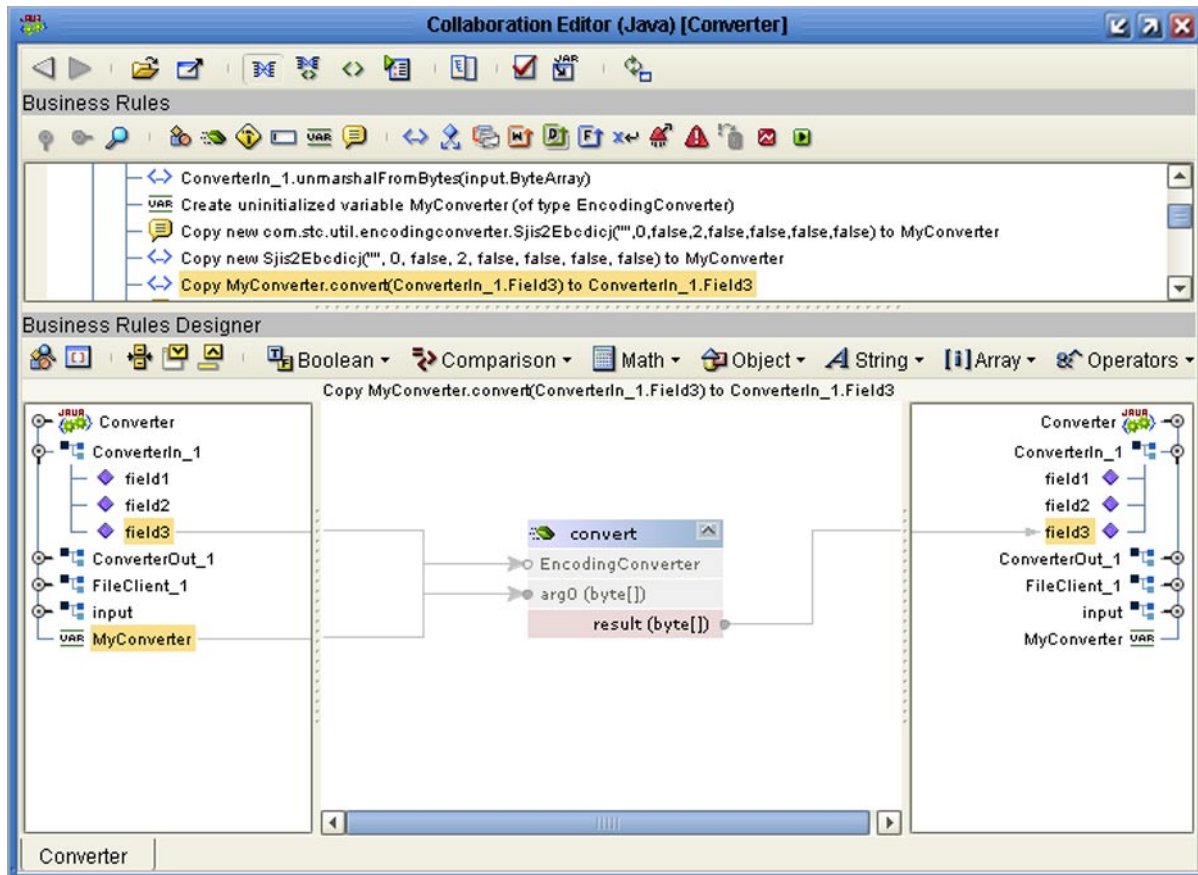


- 9 Select `convert(byte[] arg0)`, which places a *Convert* method box onto the mapping canvas.



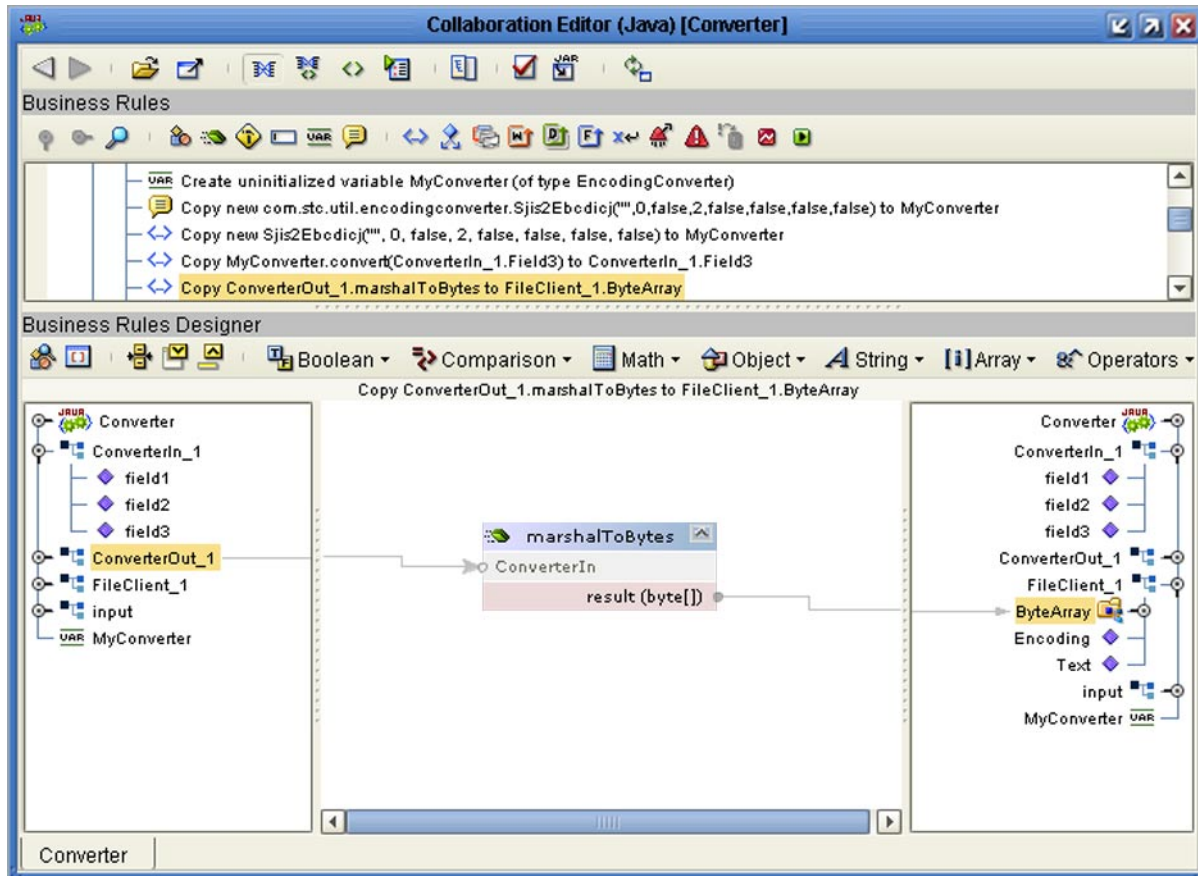
- 10 Map the node that contains data to be converted (**ConverterIn\_1 / field3**) to **arg0** (see Figure 322), and map the **result** to the node that gets the converted data (**ConverterOut\_1 / field3**).

Figure 322 Data Mapping



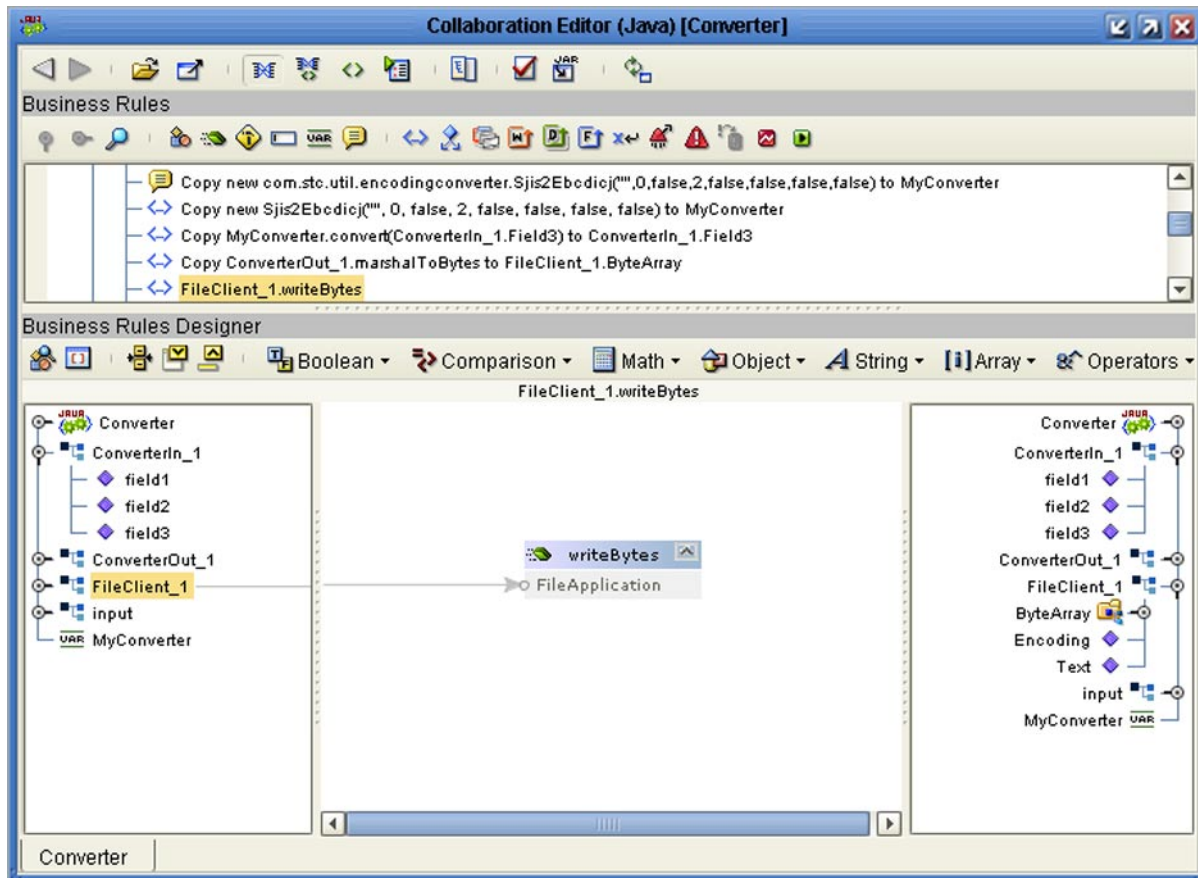
- 11 When finished with the Collaboration rules, marshal the output data using the **marshalToBytes** Java method as shown in Figure 323.

Figure 323 Marshaling Output Data



- 12 Finally, output the data using the **writeBytes** Java method as shown in Figure 324.

Figure 324 Output Data



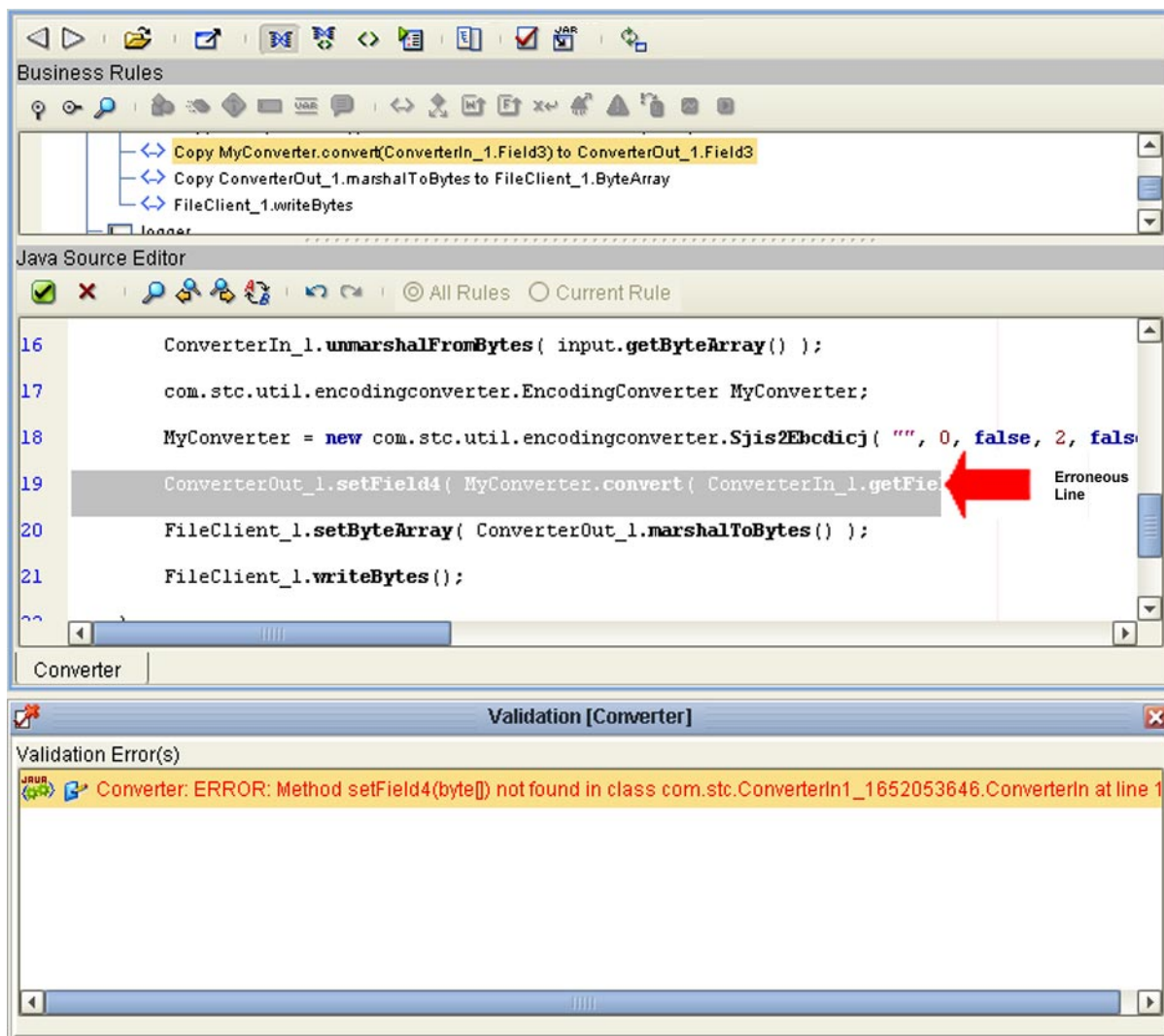
## 10.17 Validating Java-based Collaboration Definitions

Clicking the **Validate** icon in the Collaboration Editor Toolbar allows you to “precompile” the Java-based Collaboration Definition and display the errors in a validation panel below the Java Source Editor. To locate the error, double-click on the error message and the erroneous line of code will be identified as shown in Figure 326.

Figure 325 Validate Icon



Figure 326 Validating Java Code



# Collaboration Definitions (XSLT)

This chapter describes the features of the Collaboration Definition Editor (XSLT) and outlines procedures for building XSLT-based Collaboration Definitions.

## What's in This Chapter

- [Overview](#) on page 377
- [Creating an XSLT-based Collaboration Definition](#) on page 381
- [Using the Collaboration Editor \(XSLT\)](#) on page 387
- [Graphical Collaboration Methods \(XSLT\)](#) on page 393

---

## 11.1 Overview

Collaboration Definitions define how data should be processed and routed between Project components, how databases should be queried in response to requests, and how APIs to one or more applications should be invoked. The external data formats that characterize the input and output data structures in a Collaboration Definition are described by Object Type Definitions (OTDs).

A Collaboration will typically receive a message containing the external representation of a particular OTD. It will use the *unmarshal* method of an instance of that OTD to parse the data and make it accessible through the hierarchical data structure. Then it will perform some operation — for example, copying parts of the data to another OTD instance. Finally, it will invoke the *marshal* method on the other OTD instance to render the contents of its data structure as a single, serialized data stream for further transport.

At run time, an OTD instance is accessed from BPEL using XPath expressions. Each of the nodes comprising the hierarchy of the data structure has a set of properties with *get* and *set* methods.

**Important:** *If you delete an OTD in the Project Explorer, any Collaboration Definitions that have been built using that OTD will be affected. It is recommended that you run the Impact Analyzer before attempting to delete any OTDs (see [Managing Component Versions](#) on page 78).*

As with other eGate Integrator components, it is essential to manage versions of Collaboration Definitions carefully. See [Managing Component Versions](#) on page 78 for descriptions of various version control features applicable to Collaboration Definitions.

The Enterprise Designer includes two primary tools, the Collaboration Definition Wizard (XSLT) and Collaboration Editor (XSLT), that are used to create and customize your XSLT-based Collaboration Definitions. These tools are described in the following sections.

## 11.2 Using the Collaboration Definition Context Menu

**Figure 327** Collaboration Definition Context Menu



**Table 84** Collaboration Definition Context Menu Options

Option	Function
Open	Opens the appropriate Collaboration Editor, showing the selected Collaboration Definition. See <a href="#">Using the Collaboration Editor (XSLT)</a> on page 387.
ACL Management	Presents the ACL Properties dialog, with which an Administrator can assign read/write privileges to users for the selected Collaboration Definition. See the <i>Sun SeeBeyond eGate™ Integrator System Administration Guide</i> .
Version Control	Accesses the version control features for the selected component. See <a href="#">Accessing Component Version Control Features</a> on page 79 for additional information.
Cut	Copies the selected Collaboration Definition (Java only) and removes it from the current Project, after which you can paste it to another Project (once only). All changes must be committed before you can cut the Collaboration. Cut and paste is disabled for other users when you have the Collaboration checked out. See <a href="#">Moving and Replicating Repository Objects</a> on page 97 for additional information.
Copy	Copies the selected Collaboration Definition (Java only), after which you can paste it to other Projects (multiple times). All changes must be committed before you can copy the Collaboration. You can copy and paste a Collaboration even when another user has the Collaboration checked out. See <a href="#">Moving and Replicating Repository Objects</a> on page 97 for additional information.



**Table 84** Collaboration Definition Context Menu Options

Option	Function
Create Diff	Presents a dialog with which you can create a difference file representing two versions of the selected Collaboration Definition. See <a href="#">Merging Changes from Another Version</a> on page 94.
Merge Diff	Presents a dialog with which you can merge modifications into a specific version of the selected Collaboration Definition. See <a href="#">Merging Changes from Another Version</a> on page 94.
Properties	Presents the appropriate Collaboration Definition Properties dialog for the selected Collaboration Definition.
Generate Report	Presents a dialog in which you can generate an HTML-formatted report on the currently open Collaboration. The primary utility for this feature is for business processes.
Delete	<p>Deletes the selected Collaboration Definition, subject to the following conditions:</p> <ul style="list-style-type: none"> <li>▪ You have <i>write</i> privileges for the component (see ACL Management, above).</li> <li>▪ You have the component checked out to your workspace.</li> <li>▪ The component is not checked out by anyone other than yourself.</li> </ul> <p>If these conditions are met, a dialog is presented in which you confirm that you want to delete the selected component. Clicking <b>Yes</b> then deletes the component.</p>
Rename	Activates the field, allowing you to rename the selected Project.



## 11.3 Creating an XSLT-based Collaboration Definition

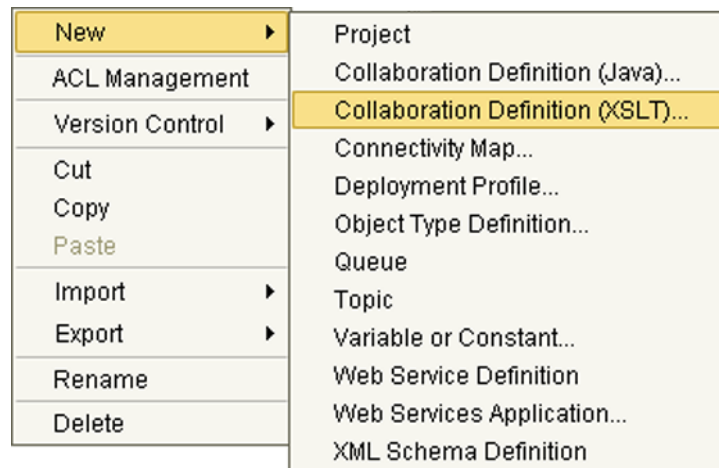
The Collaboration Definition Wizard (XSLT) guides you through the initial phases of creating an XSLT-based Collaboration Definition, and then invokes the Collaboration Editor (XSLT). Creating a Collaboration Definition (XSLT)

**Note:** Any changes made to the names of the Collaborations should be done when the Collaboration is created. If the name is changed later, the Collaboration should be opened, regenerated if applicable, and saved again. This procedure should also be performed before creating the Connectivity Map and Deployment Profile.

### To create an XSLT-based Collaboration Definition

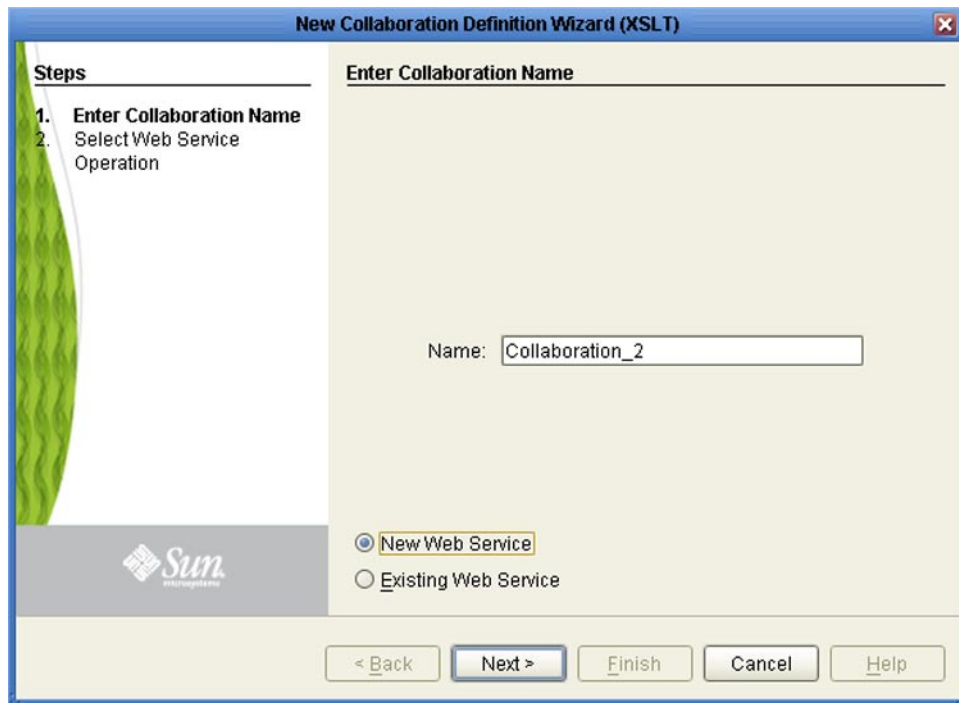
- 1 Right-click on a Project in the Enterprise Explorer to display the Project context menu (see Figure 328).

**Figure 328** Project Context Menu - New Web Service Definition



- 2 Select **New > Collaboration Definition (XSLT)** to invoke the Collaboration Definition Wizard (XSLT).
- 3 Enter a **Name** for your Collaboration, as shown in Figure 329.

**Figure 329** Collaboration Wizard (XSLT) Dialog



- 4 Select a web service, which can be either:
  - ♦ A new web service.
  - ♦ An existing web service (for example, an eInsight process or a Java Web Service Operation).
- 5 Click **Next** to proceed to the next Wizard dialog.

The dialog sequence that appears next depends on which Web Service Type you selected in step 4.

## New Web Service

If you selected a new web service, you will be presented with the Wizard dialog shown in Figure 330.

To create a new web service

- 1 Enter an operation name, as shown in Figure 330. This will become the *method* that can be used to invoke the XSLT-based Collaboration as a web service.

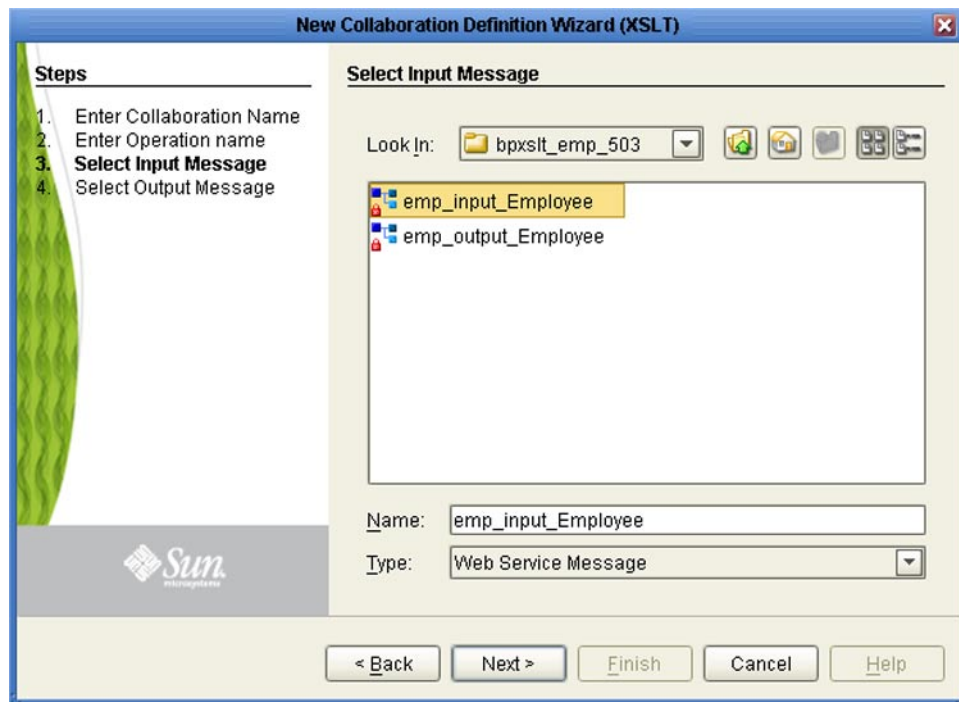
**Figure 330** New Web Service: Operation Name

The screenshot shows a Windows-style dialog box titled "New Collaboration Definition Wizard (XSLT)". On the left, a "Steps" pane lists four steps: 1. Enter Collaboration Name, 2. Enter Operation name (highlighted with a green arrow), 3. Select Input Message, and 4. Select Output Message. The main area is titled "Enter Operation name" and contains a text input field with the text "DemoOperation" and a cursor at the end. Below the input field is the Sun Microsystems logo. At the bottom, there are five buttons: "< Back", "Next >", "Finish", "Cancel", and "Help".

- 2 Click **Next** to proceed to the next Wizard dialog.

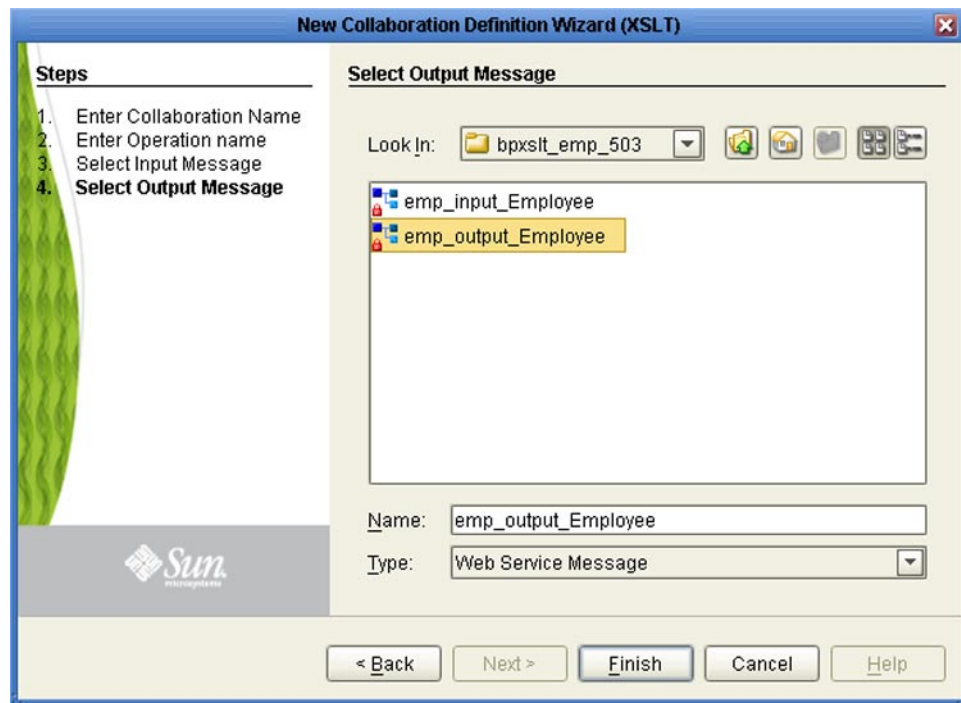
- 3 Select the input web service message, as shown in Figure 331.

**Figure 331** New Web Service: Input Message



- 4 Click **Next** to proceed to the next Wizard dialog.
- 5 Select the output web service message, as shown in Figure 332.

**Figure 332** New Web Service: Output Message



- 6 Click **Finish** to proceed to the Collaboration Editor (XSLT).

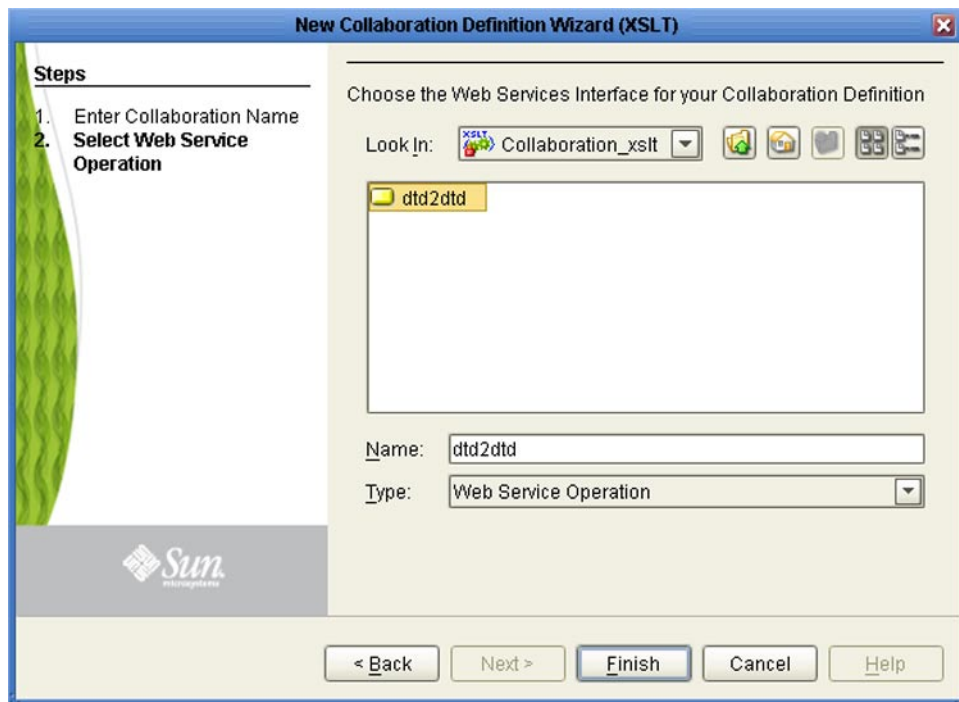
## Existing Web Service

If you selected an existing Web service, you will be presented with the Wizard dialog shown in Figure 333.

To use an existing Web service

- 1 Select a web service operation, which must be an installed web service.

**Figure 333** Existing Web Service: Select Operation



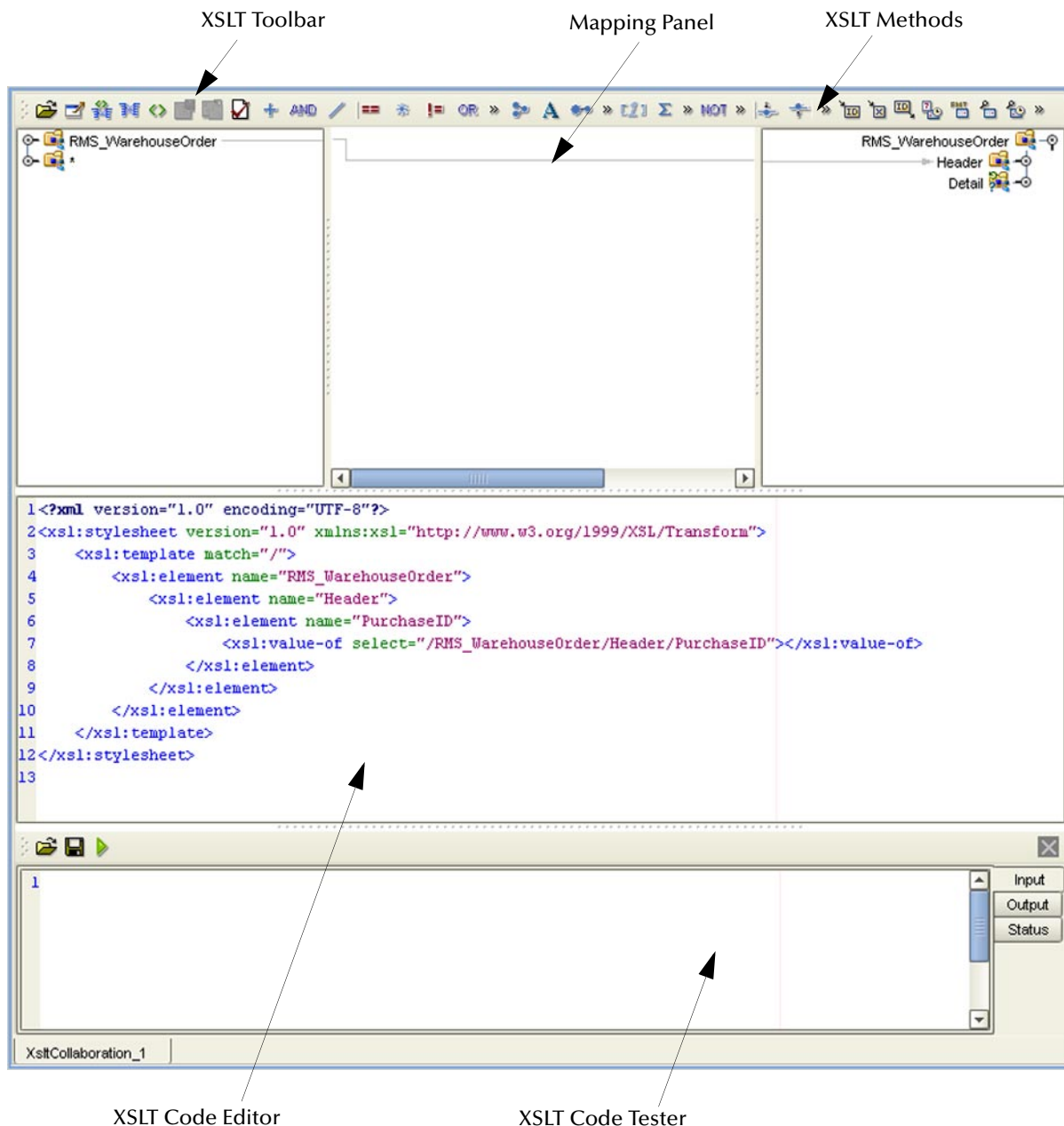
- 2 Click **Finish** to proceed to the Collaboration Editor (XSLT).

## 11.4 Using the Collaboration Editor (XSLT)

The Collaboration Editor (XSLT) displays the structure of the selected XSLT-based Collaboration Definition, and allows you to define the transformations that the Collaboration will perform at run time.

After you have created an XSLT-based Collaboration Definition using the Collaboration Definition Wizard (XSLT), the Collaboration Editor (XSLT) appears in the Editor panel of the Enterprise Designer. Major features of this window are identified in Figure 334.

**Figure 334** Collaboration Editor (XSLT)




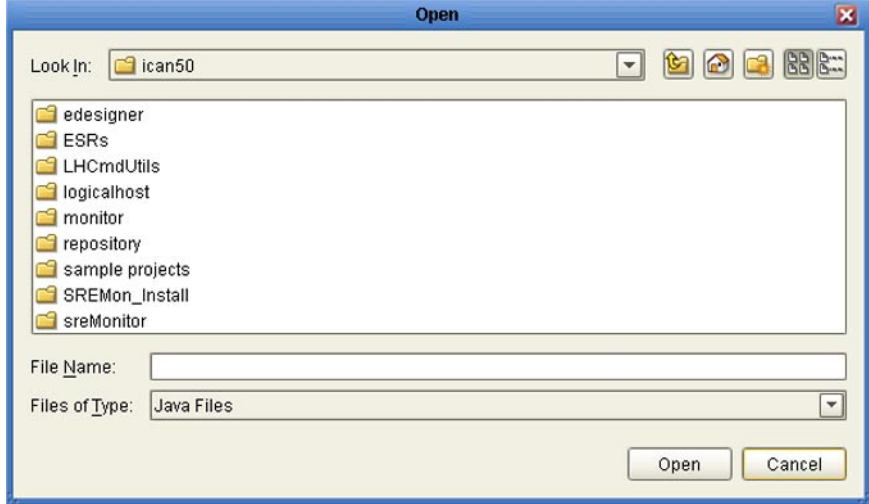







You can also invoke the Collaboration Editor (XSLT) by selecting **Open** in the context menu for an existing XSLT-based Collaboration Definition in the Enterprise Explorer.

The XSLT Mapping panel is used to map fields and add methods to the Collaboration Definition. At the top left of the Mapping panel is the toolbar, containing icons as described in Table 85. At the top right of the Mapping area is the XSLT Method Palette, which contains a collection of XSLT methods. The XSLT Code Editor panel allows you to view, enter and edit the XSLT code for the Collaboration Definition. The Tester panel allows you to run the XSLT code without deploying the Project.




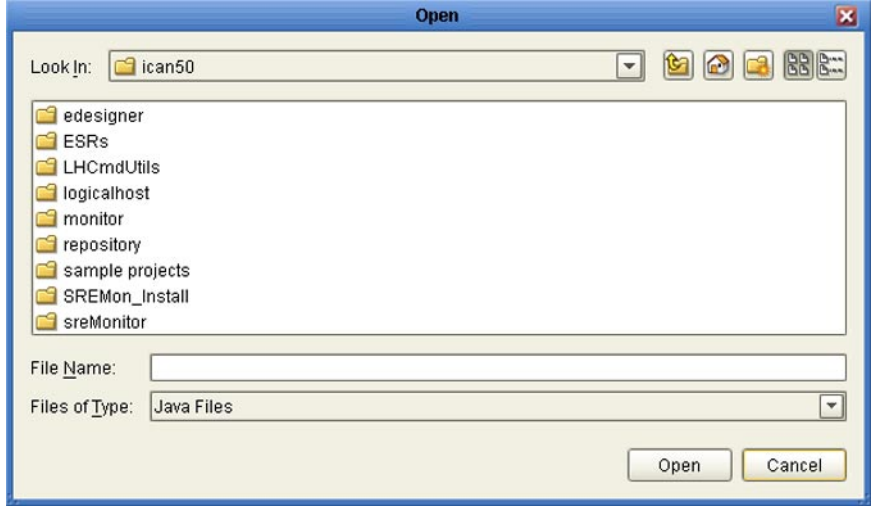


## 11.4.1 XSLT Toolbar Icons

**Table 85** XSLT Toolbar Icons

Icon	Command	Function
	Import File	<p>Presents a dialog with which you can locate and select a Collaboration Definition (XSLT) to import. When you import a file, any previously generated code or rules are deleted. The imported code does not get appended to the existing Collaboration Rules.</p> 
	Save XSLT to a Local File	Presents a dialog with which you can save the selected XSLT-based Collaboration Definition to a file.
	Show Maps and Code	Displays both the Mapping and XSLT Code areas. This is the default view setting.
	Show Mapping Only	Displays the Mapping area and hides the XSLT Code area.
	Show XSLT Code Only	Displays the XSLT Code area and hides the Mapping area.
	Commit Code Changes	Commits changes made to the XSLT code since the last time it was committed. Changes will now be shown in the Mapping panel.
	Roll Back Code Changes	Cancels changes made to the XSLT code since the last time it was committed.
	Test XSLT Code	Displays the XSLT Tester panel.

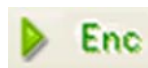
## 11.4.2 XSLT Tester Icons

**Table 86** XSLT Tester Icons

Icon	Command	Function
	Import File	<p>Presents a dialog with which you can locate and select a Collaboration Definition (XSLT) to import. When you import a file, any previously generated code or rules are deleted. The imported code does not get appended to the existing Collaboration Rules.</p> 
	Save	Saves changes made to the Collaboration Definition (XSLT) file.
	Start Transforming	Runs the tester with the data values as shown.

If you are using the extended language options (see [Setting Up Options](#) on page 53), an *Encoding* icon (see Figure 335) appears in the Collaboration Tester. This icon allows you to specify the data encoding for the input file (applies to XML-based data only).

**Figure 335** Encoding Icon



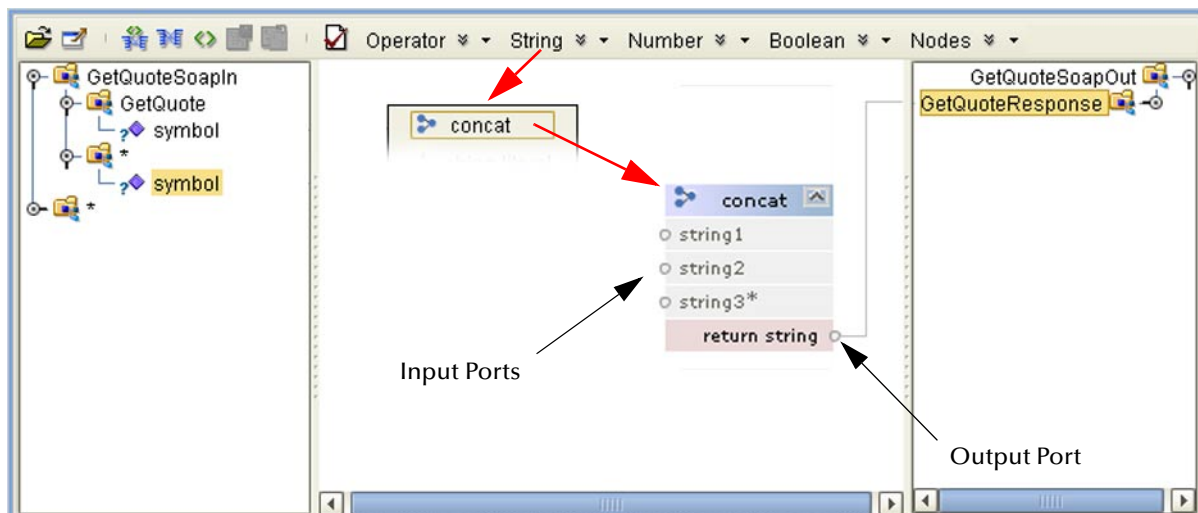
**Note:** *Extended language options currently apply only to Japanese and Korean localized versions of eGate Integrator.*

## Collaboration Method Menus

The Collaboration Method menus list commonly-used methods for use in XSLT-based Collaboration Definitions. Using these methods, you can create your Collaboration Definition graphically, without entering XSLT code (see [Graphical Collaboration Methods \(XSLT\)](#) on page 393). Clicking a category in the toolbar displays a list of methods for that category. Clicking an individual method in the list places the corresponding Method Box onto the Business Rules Designer mapping canvas.

As an example, clicking **String** in the toolbar displays a menu listing string methods. Clicking **concat** in the menu places an **concat** method box on the mapping canvas. The method box contains input and output ports which you connect to the appropriate nodes in the OTD tree structures by dragging the cursor.

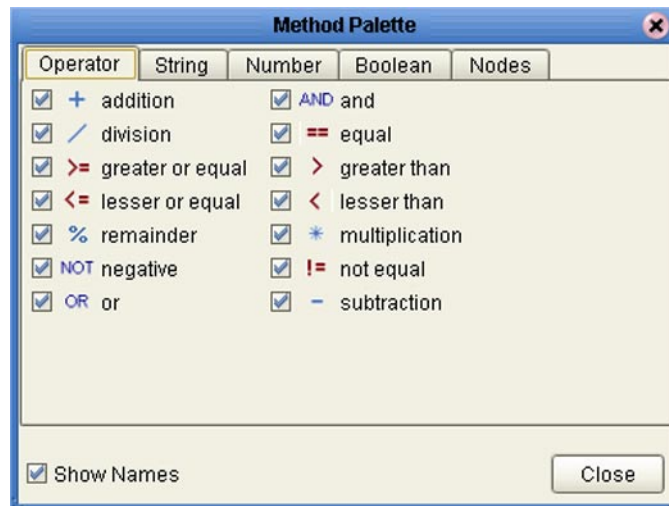
**Figure 336** Collaboration Definition Editor (XSLT): Method Menu and Boxes



Clicking the **Settings** option in any menu presents the dialog shown in Figure 337, which allows you to select the methods that are displayed in the menus. Select a check box to add the method to the menu; clear a check box to remove the method from the menu.

## Collaboration Method Palette (XSLT)

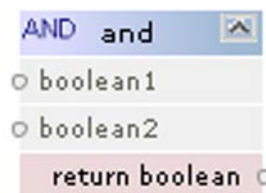
**Figure 337** Collaboration Method Palette (XSLT)



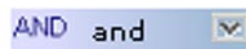
## Collaboration Method Boxes (XSLT)

The method boxes are placed on the mapping canvas of the Business Rules Designer by clicking the method in the drop-down method menu. As shown in [Figure 336 on page 391](#), the method boxes typically have input and output ports that you link to fields in the left and right panels, respectively. The method boxes are expanded by default (see Figure 338); you can collapse them (see Figure 339) by clicking the caret (^) in the upper right corner of the box. Clicking the now-inverted caret expands the box. Some boxes expand further as needed to provide additional argument nodes.

**Figure 338** Expanded Method Box



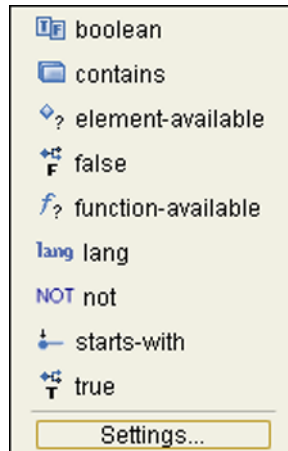
**Figure 339** Collapsed Method Box



## 11.5 Graphical Collaboration Methods (XSLT)

### 11.5.1 Boolean Methods


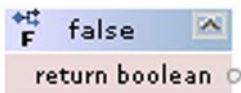

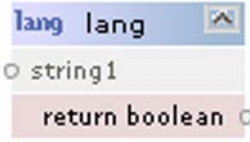
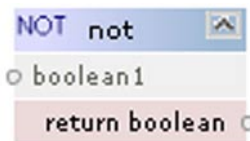
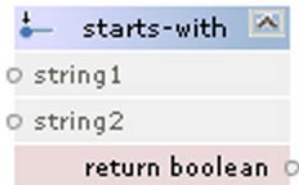
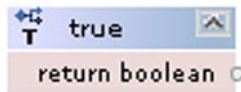
**Figure 340** Boolean Methods Menu



**Table 87** Boolean Collaboration Methods (XSLT)

Method Box	Description/Usage
	<p>At run time, the <b>boolean</b> method converts the argument <i>object1</i> to a Boolean true or false as follows:</p> <ul style="list-style-type: none"> <li>A number is true if and only if it is neither <math>\pm</math>zero nor NaN (not a number).</li> <li>A node-set is true if and only if it is non-empty.</li> <li>A string is true if and only if it is non-zero.</li> <li>An object of a type other than the four basic types is converted to a Boolean in a way that is dependent on that type.</li> </ul>
	<p>At run time, the <b>contains</b> method returns Boolean <b>true</b> if the first argument (<i>string1</i>) contains the second argument (<i>string2</i>); if not, returns Boolean <b>false</b>.</p>

**Table 87** Boolean Collaboration Methods (XSLT)

Method Box	Description/Usage
	<p>The argument <i>string1</i> must be a QName, which is expanded into an expanded-name using the namespace declarations in scope for the expression. At run time, the <b>element-available</b> method returns true if and only if the expanded-name is the name of an instruction. If the expanded-name has a namespace URI equal to the XSLT namespace URI, then it refers to an element defined by XSLT. Otherwise, it refers to an extension element. If the expanded-name has a null namespace URI, the element-available function will return false.</p>
	<p>At run time, the <b>false</b> method returns Boolean <b>false</b>.</p>
	<p>The argument <i>string1</i> must be a QName, which is expanded into an expanded-name using the namespace declarations in scope for the expression. At run time, the <b>function-available</b> method returns true if and only if the expanded-name is the name of a function in the function library. If the expanded-name has a non-null namespace URI, then it refers to an extension function; otherwise, it refers to a function defined by XPath or XSLT.</p>
	<p>At run time, the <b>lang (language)</b> method returns Boolean <b>true</b> or <b>false</b> depending upon whether the language of the context node as specified by <i>xml:lang</i> attributes is the same as, or is a sub-language of, the language specified by the argument <i>string</i> (<i>string1</i>). Returns Boolean <b>false</b> if the attribute <i>xml:lang</i> does not exist.</p>
	<p>At run time, the <b>not</b> method returns the inverse of <i>boolean1</i>.</p>
	<p>At run time, the <b>starts-with</b> method returns Boolean <b>true</b> if the first argument (<i>string1</i>) starts with the second argument (<i>string2</i>); if not, returns Boolean <b>false</b>.</p>
	<p>At run time, the <b>true</b> method returns Boolean <b>true</b>.</p>

## 11.5.2 Nodes Methods

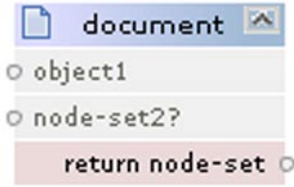
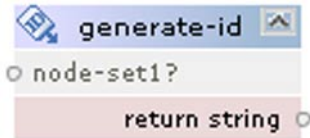
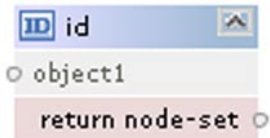
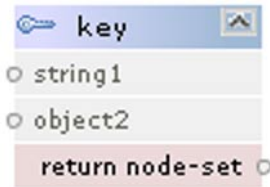
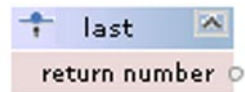
**Figure 341** Nodes Methods Menu



**Table 88** Nodes Collaboration Methods (XSLT)


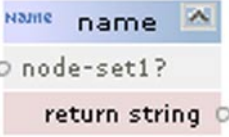
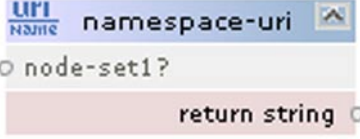
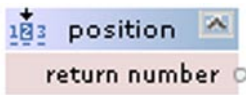
Method Box	Description/Usage
<p>The 'count' method box shows the 'CNT' icon, the text 'count', and a small icon. Below it, 'node-set1' is listed as an input, and 'return number' is shown as the output.</p>	At run time, the <b>count</b> method returns the number of nodes in the argument <i>node-set1</i> .
<p>The 'current' method box shows a blue arrow icon, the text 'current', and a small icon. Below it, 'return node-set' is shown as the output.</p>	At run time, the <b>current</b> method returns a node-set that has the current node as its only member.

**Table 88** Nodes Collaboration Methods (XSLT)

Method Box	Description/Usage
	<p>The <b>document</b> method allows access to XML documents other than the main source document.</p> <ul style="list-style-type: none"> <li>When the method has exactly one argument and the argument is a node-set, then for each node in the argument node-set, the result is the union of the result of calling the document function with the first argument being the string-value of the node, and the second argument being a node-set with the node as its only member.</li> <li>When the method has two arguments and the first argument is a node-set, then for each node in the argument node-set, the result is the union of the result of calling the document function with the first argument being the string-value of the node, and with the second argument being the second argument passed to the document function. When the first argument is not a node-set, the first argument is converted to a string as if by a call to the string function. This string is treated as a URI reference; the resource identified by the URI is retrieved.</li> </ul>
	<p>At run time, the <b>generate-id</b> method returns a string that uniquely identifies the node in the argument <i>node-set1?</i> that is first in document order. The unique identifier must consist of ASCII alphanumeric characters and must start with an alphabetic character. Thus, the string is syntactically an XML name.</p>
	<p>When the argument <i>object1</i> is not of type node-set, the <b>id</b> method converts the argument to a string as if by a call to the string function; the string is split into a whitespace-separated list of tokens; the result is a node-set containing the elements in the same document as the context node that have a unique ID equal to any of the tokens in the list.</p>
	<p>The <b>key</b> method does for keys what the id function does for IDs. The value of the first argument (<i>string1</i>), which specifies the name of the key, must be a QName — which is expanded into an expanded-name using the namespace declarations in scope for the expression.</p> <ul style="list-style-type: none"> <li>When the second argument is of type node-set, then the result is the union of the result of applying the key function to the string value of each of the nodes in the argument node-set.</li> <li>When the second argument is of any other type, the argument is converted to a string as if by a call to the string function; it returns a node-set containing the nodes in the same document as the context node that have a value for the named key equal to this string.</li> </ul>
	<p>At run time, the <b>last</b> method returns a number equal to the context size from the expression evaluation context.</p>



**Table 88** Nodes Collaboration Methods (XSLT)

Method Box	Description/Usage
 <p>The <b>local-name</b> method box has a blue header with a 'NAME' icon and a 'local-name' label. Below the header is a light gray input field containing 'node-set1?'. At the bottom is a pink output field labeled 'return string'.</p>	<p>At run time, the <b>local-name</b> method returns the local part of the expanded-name of the node in the argument <i>node-set1?</i> that is first in document order. If the argument is empty or the first node has no expanded-name, an empty string is returned. If the argument is omitted, it defaults to a node-set with the context node as its only member.</p>
 <p>The <b>name</b> method box has a blue header with a 'NAME' icon and a 'name' label. Below the header is a light gray input field containing 'node-set1?'. At the bottom is a pink output field labeled 'return string'.</p>	<p>At run time, the <b>name</b> method returns a string containing a QName representing the expanded-name of the node in the argument <i>node-set1?</i> that is first in document order. If the argument is empty or the first node has no expanded-name, an empty string is returned. If the argument is omitted, it defaults to a node-set with the context node as its only member.</p>
 <p>The <b>namespace-uri</b> method box has a blue header with a 'URI' icon and a 'namespace-uri' label. Below the header is a light gray input field containing 'node-set1?'. At the bottom is a pink output field labeled 'return string'.</p>	<p>At run time, the <b>namespace-uri</b> method returns the namespace URI of the expanded-name of the node in the argument <i>node-set1?</i> that is first in document order. If the argument is empty, the first node has no expanded-name, or the namespace URI of the expanded-name is null, an empty string is returned. If the argument is omitted, it defaults to a node-set with the context node as its only member.</p>
 <p>The <b>position</b> method box has a blue header with a '1 2 3' icon and a 'position' label. Below the header is a pink output field labeled 'return number'.</p>	<p>At run time, the <b>position</b> method returns a number equal to the context position from the expression evaluation context.</p>

## 11.5.3 Number Methods


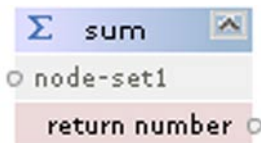
**Figure 342** Number Methods Menu



**Table 89** Number Collaboration Methods (XSLT)

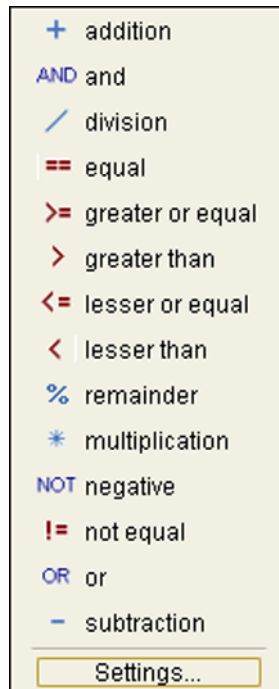
Method Box	Description/Usage
The 'ceiling' method box shows a 'C' icon, '1.2' version, and 'ceiling' text. Below is an input field with 'number1' and a 'return number' output field.	At run time, the <b>ceiling</b> method returns the smallest (closest to negative infinity) number that is not less than the argument and that is an integer.
The 'floor' method box shows an 'F' icon, '1.2' version, and 'floor' text. Below is an input field with 'number1' and a 'return number' output field.	At run time, the <b>floor</b> method returns the largest (closest to positive infinity) number that is not greater than the argument and that is an integer.
The 'number' method box shows a '#' icon and 'number' text. Below is an input field with 'object1?' and a 'return number' output field.	At run time, the <b>number</b> method converts its argument ( <i>object1</i> ) to a number. An object of a type other than the four basic types is converted to a number in a way that is dependent on that type.
The 'number-literal' method box shows a '1' icon and 'number-literal' text. Below is an input field with '12000' and a 'return number' output field.	<p>Dragging the <b>number-literal</b> icon first presents a dialog in which you enter the literal value, such as "12000":</p> A dialog box titled 'Number Literal' with a close button. It contains a 'Value:' label and a text input field with '12000'. At the bottom are 'OK' and 'Cancel' buttons. <p>At run time, the method returns a number having the specified value.</p>

**Table 89** Number Collaboration Methods (XSLT)

Method Box	Description/Usage
 <p>The method box for <b>round</b> (labeled 1.2) has a blue header with the name and a small icon. It has a single input port labeled <code>number1</code> and a single output port labeled <code>return number</code>.</p>	<p>At run time, the <b>round</b> method returns the number that is closest to the argument <i>number1</i> and that is an integer.</p> <ul style="list-style-type: none"> <li>▪ If there are two such numbers, then the one that is closest to positive infinity is returned.</li> <li>▪ If the argument is not a number (NaN), then NaN is returned.</li> <li>▪ If the argument is positive infinity, then positive infinity is returned.</li> <li>▪ If the argument is negative infinity, then negative infinity is returned.</li> <li>▪ If the argument is positive zero, then positive zero is returned.</li> <li>▪ If the argument is negative zero, then negative zero is returned.</li> <li>▪ If the argument is less than zero, but greater than or equal to -0.5, then negative zero is returned.</li> </ul>
 <p>The method box for <b>sum</b> has a blue header with the name and a small icon. It has a single input port labeled <code>node-set1</code> and a single output port labeled <code>return number</code>.</p>	<p>At run time, the <b>sum</b> method returns the sum, for each node in the argument <i>node-set1</i>, of the result of converting the string-values of the node to a number.</p>

## 11.5.4 Operator Methods

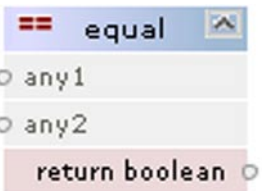
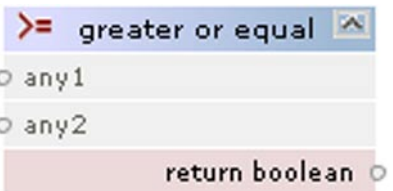
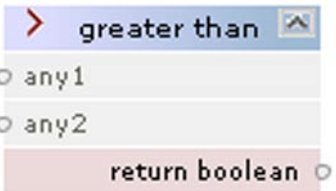
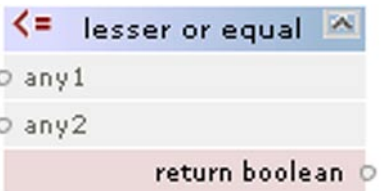
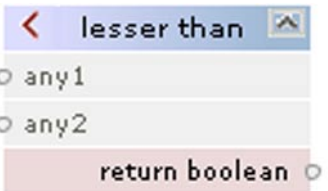
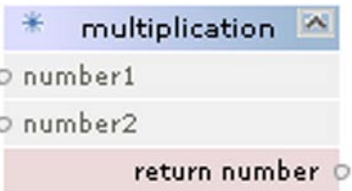
**Figure 343** Operator Methods Menu



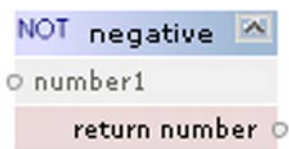
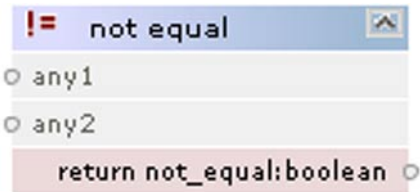
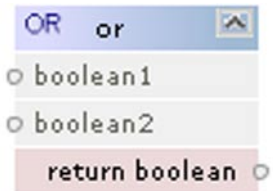
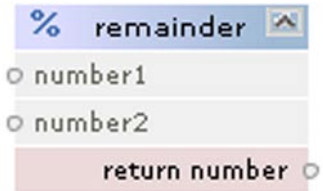
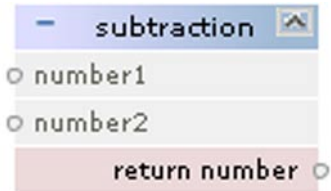
**Table 90** Operator Collaboration Methods (XSLT)

Method Box	Description/Usage
<p>The addition method box contains a blue header with a plus sign and the text "addition". Below the header are two input fields labeled "number1" and "number2". At the bottom is a red output field labeled "return number".</p>	At run time, the <b>addition</b> method adds the value of <i>number1</i> to the value of <i>number2</i> , returns the sum.
<p>The AND method box contains a blue header with the text "AND and". Below the header are two input fields labeled "boolean1" and "boolean2". At the bottom is a red output field labeled "return boolean".</p>	At run time, the <b>and</b> method returns Boolean true if both <i>boolean1</i> and <i>boolean2</i> are true; otherwise, returns Boolean false.
<p>The division method box contains a blue header with a forward slash and the text "division". Below the header are two input fields labeled "number1" and "number2". At the bottom is a red output field labeled "return number".</p>	At run time, the <b>division</b> method divides the value of <i>number1</i> by the value of <i>number2</i> , returns the quotient.

**Table 90** Operator Collaboration Methods (XSLT)

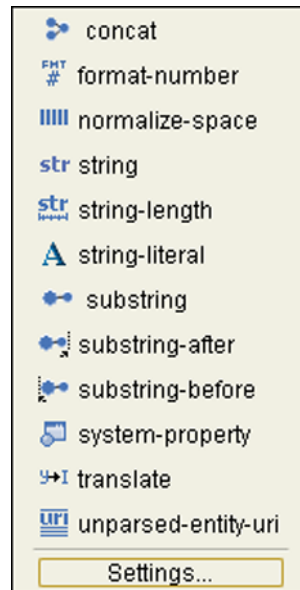
Method Box	Description/Usage
	At run time, the <b>equal</b> method returns Boolean true if <i>any1</i> is equal to <i>any2</i> ; otherwise, returns Boolean false.
	At run time, the <b>greater_or_equal</b> method returns Boolean true if <i>any1</i> is greater than or equal to <i>any2</i> ; otherwise, returns Boolean false.
	At run time, the <b>greater_than</b> method returns Boolean true if <i>any1</i> is greater than <i>any2</i> ; otherwise, returns Boolean false.
	At run time, the <b>lesser_or_equal</b> method returns Boolean true if <i>any1</i> is less than or equal to <i>any2</i> ; otherwise, returns Boolean false.
	At run time, the <b>lesser_than</b> method returns Boolean true if <i>any1</i> is less than <i>any2</i> ; otherwise, returns Boolean false.
	At run time, the <b>multiplication</b> method multiplies the value of <i>number1</i> by the value of <i>number2</i> , returns the product.

**Table 90** Operator Collaboration Methods (XSLT)


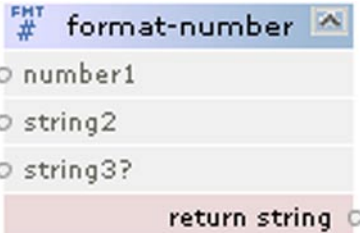
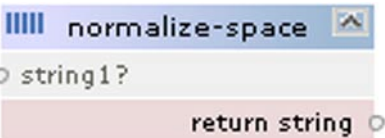
Method Box	Description/Usage
	At run time, the <b>negative</b> method returns the arithmetic negation of <i>number1</i> .
	At run time, the <b>not_equal</b> method returns Boolean true if <i>any1</i> is not equal to <i>any2</i> ; otherwise, returns Boolean false.
	At run time, the <b>OR</b> method returns Boolean false if both <i>boolean1</i> and <i>boolean2</i> are false; otherwise, returns Boolean true.
	At run time, the <b>remainder</b> method divides the numerical value of <i>number1</i> by the numerical value of <i>number2</i> , and returns the remainder.
	At run time, the <b>subtraction</b> method subtracts the numerical value of <i>number2</i> from the numerical value of <i>number1</i> , returns the difference.

## 11.5.5 String Methods

**Figure 344** String Methods Menu



**Table 91** String Collaboration Methods (XSLT)


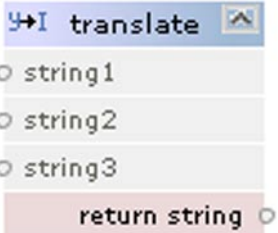

Method Box	Description/Usage
 <p>The 'concat' method box shows a blue icon with a plus sign, the text 'concat', and a small 'A' icon. Below the title are three input fields: 'string1', 'string2', and 'string3*'. At the bottom is a red box labeled 'return string' with a small circle to its right.</p>	<p>At run time, the <b>concat</b> method returns the string created by concatenating <i>string2</i> to the end of <i>string1</i>, and <i>string3*</i> (if present) to the end of <i>string2</i>.</p>
 <p>The 'format-number' method box shows a blue icon with 'FMT #' and the text 'format-number', and a small 'A' icon. Below the title are three input fields: 'number1', 'string2', and 'string3?'. At the bottom is a red box labeled 'return string' with a small circle to its right.</p>	<p>At run time, the <b>format-number</b> method converts its first argument (<i>number1</i>) to a string using the format pattern string specified by the second argument (<i>string2</i>) and the decimal-format named by the third argument (<i>string3</i>) or the default decimal-format, if there is no third argument.</p>
 <p>The 'normalize-space' method box shows a blue icon with '    ' and the text 'normalize-space', and a small 'A' icon. Below the title is one input field: 'string1?'. At the bottom is a red box labeled 'return string' with a small circle to its right.</p>	<p>At run time, the <b>normalize-space</b> method returns the argument <i>string1?</i> with whitespace normalized by stripping leading and trailing whitespace and replacing sequences of whitespace characters by a single space. If the argument is omitted, it defaults to the string-value of the context node.</p>

**Table 91** String Collaboration Methods (XSLT)

Method Box	Description/Usage
	At run time, the <b>string</b> method returns a string representation of the input object.
	At run time, the <b>string-length</b> method returns the number of characters in the string.
	<p>Dragging the <b>string-literal</b> icon first presents a dialog in which you enter the literal value; for example, “Data received”:</p>  <p>At run time, the method returns a string having the specified value.</p>
	At run time, the <b>substring</b> method returns the substring of the first argument ( <i>string1</i> ) starting at the position specified in the second argument ( <i>number2</i> ) with length specified in the third argument ( <i>number3?</i> ). If the third argument is not specified, it returns the substring starting at the position specified in the second argument and continuing to the end of the string.
	At run time, the <b>substring-after</b> method returns the substring of the first argument ( <i>string1</i> ) that follows the first occurrence of the second argument ( <i>string2</i> ) in the first argument string; returns an empty string if the first argument string does not contain the second argument string.
	At run time, the <b>substring-before</b> method returns the substring of the first argument ( <i>string1</i> ) that precedes the first occurrence of the second argument ( <i>string2</i> ) in the first argument string; returns an empty string if the first argument string does not contain the second argument string.



**Table 91** String Collaboration Methods (XSLT)

Method Box	Description/Usage
 <p>The <b>system-property</b> method box has a blue header with a key icon on the left and a help icon on the right. Below the header is a light gray input field labeled "string1". At the bottom is a red output field labeled "return object".</p>	<p>At run time, the <b>system-property</b> method returns an object representing the value of the system property identified by <i>string1</i>. If there is no such system property, the empty string should be returned.</p>
 <p>The <b>translate</b> method box has a blue header with a "y→I" icon on the left and a help icon on the right. Below the header are two light gray input fields labeled "string1" and "string2". Below "string2" is a third light gray input field labeled "string3". At the bottom is a red output field labeled "return string".</p>	<p>At run time, the <b>translate</b> method returns the first argument (<i>string1</i>) with occurrences of characters in the second argument (<i>string2</i>) replaced by the character at the corresponding position in the third argument (<i>string3</i>). If a character occurs more than once in the second argument (<i>string2</i>), then the first occurrence determines the replacement character. If the third argument (<i>string3</i>) is longer than the second argument (<i>string2</i>), then excess characters are ignored. Refer to the W3C <i>XML Path Language</i> documentation for additional conditions.</p>
 <p>The <b>unparsed-entity-uri</b> method box has a blue header with a "uri" icon on the left and a help icon on the right. Below the header is a light gray input field labeled "string1". At the bottom is a red output field labeled "return string".</p>	<p>At run time, the <b>unparsed-entity-uri</b> method returns the URI of the unparsed entity with the specified name (<i>string1</i>) in the same document as the context node. It returns an empty string if there is no such entity.</p>

## 11.6 Design Procedures

### 11.6.1 Creating a For-Each Construct

To create a for-each construct in a Collaboration Definition (XSLT)

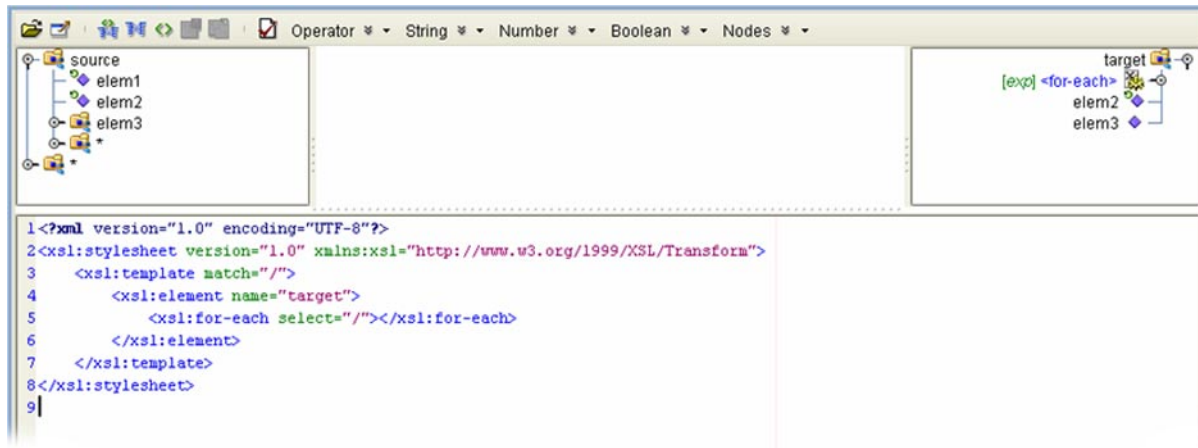
- 1 Open a Collaboration Definition.
- 2 Select a repeating node, for example **target > elem1** (see Figure 345), and right-click to display the context menu.
- 3 Select **Insert XSLT Rule** from the menu, and then **for-each**.

Figure 345 For-Each Construct - Step 1



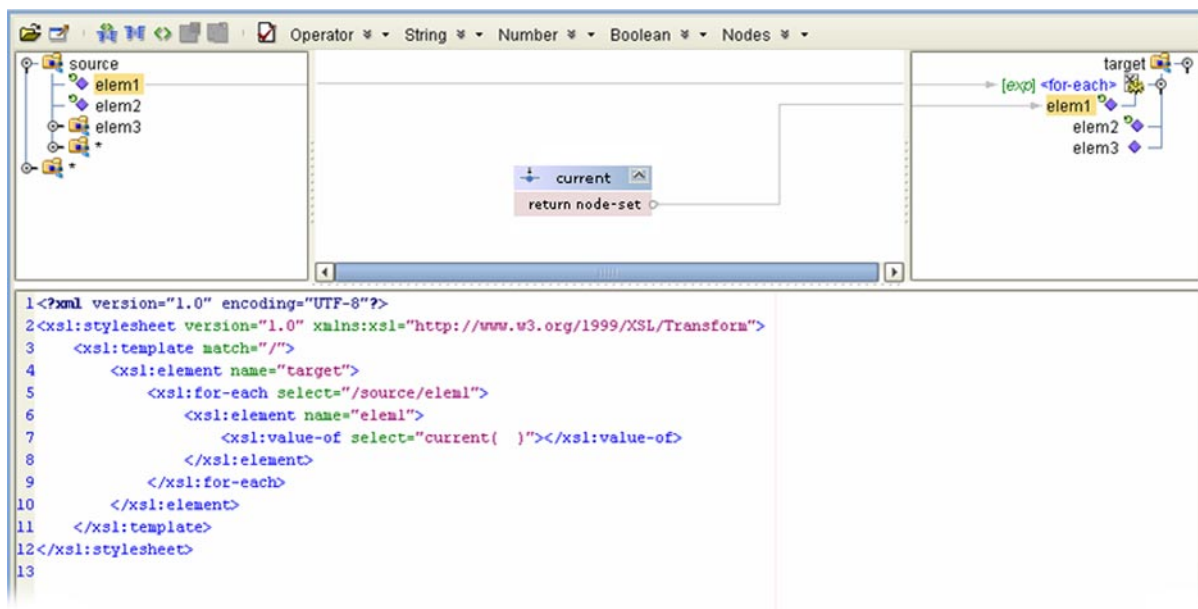
- 4 The **for-each** construct appears as shown in Figure 346.

**Figure 346** For-Each Construct - Step 2



- 5 As shown in Figure 347:
  - A Expand the node to display **elem1** as a sub-node.
  - B Select the node **source > elem1** and map it to the node **target > [exp] <for-each>**.
  - C From the **Nodes** menu, drag the **current** method onto the canvas.
  - D Map the output node of the **current** method box to the **target > [exp] <for-each> > elem1** node.

**Figure 347** For-Each Construct - Step 3



- 6 The **for-each** construct is now created, copying the value of **source > elem1** into **target > elem1**.

- 7 Click the *Test XSLT Code* icon in the toolbar to open the XSLT Tester, and enter the following test data into the input text field:

```
<source>
<elem1>1</elem1>
<elem1>2</elem1>
<elem2>1</elem2>
</source>
```

- 8 Click the *Start Transforming* button in the tester toolbar; the following should appear in the *Output* panel:

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

<target>
  <elem1>1</elem1>
  <elem1>2</elem1>
</target>
```

# Web Services

This chapter describes the use of the web services capability of eGate Integrator, acting with other components of the Java™ Composite Application Platform Suite.

## What's in This Chapter

- [Overview](#) on page 409
- [Developing Web Services](#) on page 413
- [Creating XML Schema Definitions](#) on page 418
- [Creating Web Service Definitions](#) on page 466
- [Configuring WSDL Binding Properties](#) on page 496

---

## 12.1 Overview

Web services developed using the Java™ Composite Application Platform Suite adhere to guidelines and specifications developed by the Web Services-Interoperability Organization (WS-I) to the maximum possible extent. As a result, these web services currently support *document/literal* and *RPC/literal* messaging only.

At the foundation of Sun SeeBeyond Web Services are two standard object types:

- **Web Service Definitions**

Web Service Definitions, embodied as Web Service Definition Language (WSDL) files, can be used to invoke and operate web services on the Internet and/or to access and invoke remote applications and databases. WSDL files are used when you are building a web service.

- **XML Schemas**

XML schemas define the structure, content, data types, and elements allowed in an associated XML document. They also describe the semantic-processing instructions associated with the elements contained in the document. The XML schema is embodied in an XML Schema Definition (XSD) file, itself an XML document. XML schemas are used when you are building a web service based on an eGate Collaboration Definition (Java).

The Java™ Composite Application Platform Suite contains the following tools and features that support web services capability:

- **Web Service Definition Editor**

The Web Service Definition Editor is an editor within Enterprise Designer used to develop new, or edit existing, WSDL documents. See [Creating Web Service Definitions](#) on page 466.

- **XML Schema Editor**

The XML Schema Editor is an editor within Enterprise Designer used to develop new, or edit existing, XML schemas. See [Creating XML Schema Definitions](#) on page 418.

- **WSDL Binding Editor**

The WSDL Binding Editor is an editor within Enterprise Designer used to modify the default WSDL binding configuration for Web Services External Application Connectors (see [Configuring WSDL Binding Properties](#) on page 496).

- **UDDI Registry**

The UDDI Registry is a centralized server for publishing and discovering Web Service Definitions for all Web Services developed in and exposed from Java CAPS. See [Accessing Web Service Definitions](#) on page 417.

### 12.1.1 Terminology

XML-based metadata used in eGate Integrator appears in various forms that are used in specific ways. To help prevent confusion, the following terminology has been adopted to identify these different forms.

**Table 92** Metadata Forms

Name	Origin and Use
XSD OTD	Created using the XSD OTD Wizard. Can be used within eGate Integrator as basis for both Java and XSD-based Collaboration Definitions. Java-based Collaboration Definitions using XSD OTDs can be exposed as web services.
XSD Object	Created by using the XML Schema Editor. Can be imported by the Web Service Definition Editor as the basis for a WSDL Object, and exported from the XML Schema Editor as an XSD Node.
XSD Node	Created by exporting a top-level node (child of the root node) from an XSD Object in the XML Schema Editor. Can be used within eGate Integrator as the basis for a Java-based Collaboration Definition, which can be exposed as a web service, and within eInsight Business Process Manager to create a container.
WSDL Object	Created using the Web Service Definition Editor. Used within eInsight Business Process Manager to represent predefined business process attributes.

## 12.1.2 Web Services Security

eGate Integrator supports three basic standards-based models for securing web services, which are described briefly in this section. These features are enabled and set up when you configure a SOAP/HTTP Web Service External System (see [Publishing a Web Service Project](#) on page 415).

### HTTP Basic Authentication

HTTP basic authentication enables you to control access to the web service. You must configure two SOAP/HTTP Web Service External Systems: one in server mode and another in client mode.

With HTTP basic authentication, a client requests access to a protected resource on the server. The server sends back a request for the client's user name and password. The client sends this information. If the server verifies that the client is allowed to access the protected resource, then the server returns the resource to the client.

By itself, this mechanism is limited. The user name and password are not encrypted. Therefore, malicious users who eavesdrop on the communications between the client and server can learn the user name and password.

### Transport Layer Security (TLS)

The Transport Layer Security (TLS) protocol is based on version 3.0 of the Secure Sockets Layer (SSL) protocol, and was designed as a replacement for SSL. The Internet Engineering Task Force has a working group that maintains this protocol.

For more information about TLS, go to <http://www.ietf.org/>.

### SOAP Message Security

The OASIS Web Services Security (WS-Security) effort includes a specification for SOAP Message Security. This specification defines a set of security-related extensions to the Simple Object Access Protocol (SOAP).

**Note:** To secure web services using WS-Security, you must configure two SOAP/HTTP Web Service External Systems: one in server mode and another in client mode.

#### User Name Token

The sender of a SOAP message can place security-related information in a security header block. For example, the header can include a *token*, which contains one or more declarations about an entity. In the following header, the token contains a user name and password.

```
<wsse:Security>
  <wsse:UsernameToken>
    <wsse:Username>Administrator</wsse:Username>
    <wsse:Password>STC</wsse:Password>
  </wsse:UsernameToken>
</wsse:Security>
```

The **wsse** prefix is defined in the following namespace:

<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd>

A related WS-Security specification defines two types of passwords that can be included in a user name token:

**Table 93** Password Types for User Name Token

Type	Description
Text	The password is clear text or a password equivalent. A <i>clear text</i> password is the original, unmodified password. A <i>password equivalent</i> is created by applying an algorithm to the password.
Digest	The password is combined with the user name and (optionally) a nonce and/or creation timestamp, and then modified by an algorithm. A <i>nonce</i> is a random value that is used only once.

To guard against replay attacks, the security header can also contain a *nonce* and a *creation timestamp*. Using a nonce and creation timestamp in a digest password helps you to prevent *replay attacks*. In a replay attack, a malicious user eavesdrops on the communications between a sender and a receiver. The malicious user learns the sender's password (encrypted or unencrypted), and then impersonates the sender using the password.

```
<wsse:Security>
  <wsse:UsernameToken>
    <wsse:Username>Administrator</wsse:Username>
    <wsse:Password Type="...#PasswordDigest">
      jg3EkQw5Ko924CZl1wEMk0SaFL87MkJJS==
    </wsse:Password>
    <wsse:Nonce>1T7eQbk04dcFo2zFVp74==</wsse:Nonce>
    <wsu:Created>2005-12-01T12:00:00Z</wsu:Created>
  </wsse:UsernameToken>
</wsse:Security>
```

The **wsu** prefix is defined in the following namespace:

<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd>



---

## 12.2 Developing Web Services

You can expose either an eGate Collaboration (Java) or a eInsight Business Process as a web service. Building and publishing the web service is essentially like designing a non-web service Project and deploying it to an Environment, as described elsewhere in this User's Guide

**Note:** *Procedures for exposing eInsight Business Processes as web services are described in the Sun SeeBeyond eInsight™ Business Process Manager User's Guide.*

### 12.2.1 Exposing Collaboration Definitions as Web Services

Exposing a Collaboration Definition (Java) can be separated into two phases, which are described in the following sections:

- [Developing Web Services](#) on page 413
- [Publishing a Web Service Project](#) on page 415

The the resulting web service can then be accessed via a UDDI Registry, as described in:

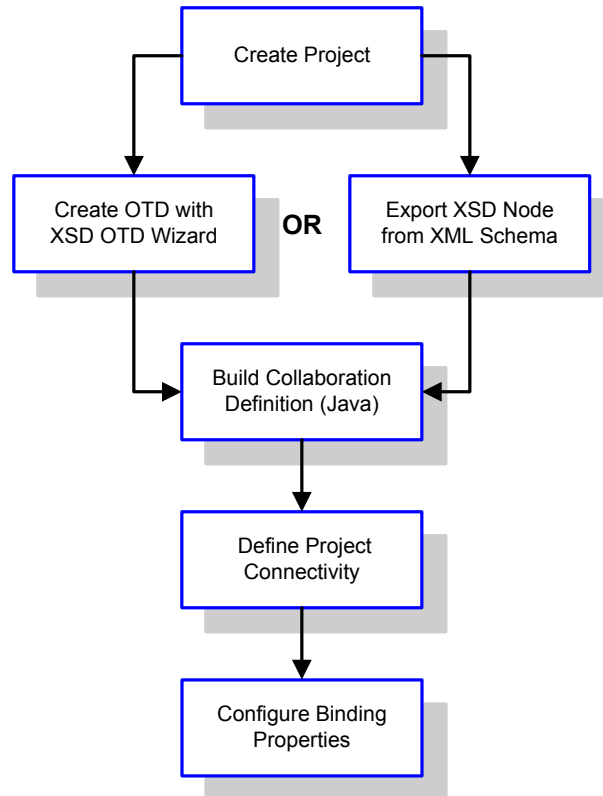
- [Accessing Web Service Definitions](#) on page 417

**Note:** *A simple end-to-end exercise, building and publishing a web service based on an eGate Collaboration (Java), is included in the Sun SeeBeyond eGate™ Integrator Tutorial.*

## 12.2.2 Creating a Web Service Project

A highly-simplified process flow for developing a Collaboration-based web service is shown in Figure 348.

**Figure 348** Development Flow - Creating a Web Service Project



### Procedure

To create a web service client or server

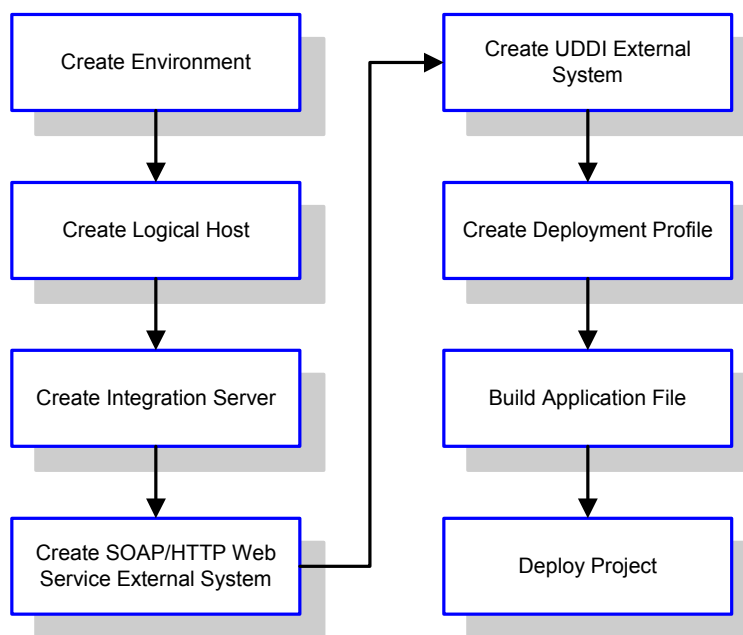
- 1 Create the Project (see [Creating a Project](#) on page 111).
- 2 Build an OTD based on an existing XSD file, using the XSD OTD Wizard (see [Using the XSD Wizard](#) on page 181), or build an XSD using the XML Schema Editor, and export an XSD Node (see [Using the XML Schema Editor](#) on page 419).
- 3 Develop a Java-based Collaboration Definition, using the XSD OTD or Node to provide the input and output messages (see [Creating a Java-based Collaboration Definition](#) on page 316).
  - ♦ Select the **Callable as an external SOAP Web Service** option.
- 4 Create a SOAP/HTTP Web Service External Application and map the Project components (see [Connectivity Maps](#) on page 125).
- 5 Modify the default configuration for the transport binding properties, if necessary (see [Configuring WSDL Binding Properties](#) on page 496).

**Note:** All XSD OTDs that are used in SOAP-callable Java Collaboration Definitions must be derived from XML schemas that have explicitly declared target namespaces.

### 12.2.3 Publishing a Web Service Project

In general, the web service definitions for all Java™ Composite Application Platform Suite objects that expose themselves as web services are listed in a UDDI registry (see [Accessing Web Service Definitions](#) on page 417). The associated WSDL files can be published to the UDDI registry directly from Enterprise Designer as described below.

**Figure 349** Development Flow - Publishing a Web Service Definition



### Procedure

**Note:** The following procedure assumes that you have installed the UDDI Registry server (*stcuddi.sar*) using the Suite Installer, and that it is running prior to creating the UDDI External System.

#### To publish a web service definition

- 1 Create and configure a run-time Environment, including the following components:
  - A A Logical Host with an integration server (see [Logical Hosts and Domains](#) on page 515).
  - B A SOAP/HTTP Web Service External System (see [SOAP/HTTP Web Service External Systems](#) on page 524).
  - C A UDDI External System (see [UDDI External Systems](#) on page 538).

- 2 Create a Deployment Profile for the Project you created in the previous section (see [Creating a Deployment Profile](#) on page 132).
- 3 Generate the Project application file (see [Building an Application File](#) on page 139) and select *Publish WSDL(s) to default UDDI Registry* in the provided dialog.
- 4 You now have the choice of either:
  - A Publishing the WSDL document directly to the UDDI Registry by deploying the Project (see [Deploying the Project](#) on page 145).
  - B Exporting the archive file to a holding directory for publishing from the command line at a later time (see following procedure).

**Note:** *The following procedure assumes that you have installed the UDDI Registry server (stcuddi.sar) using the Suite Installer, and that it is running.*

#### To publish a WSDL file from the command line

- 1 Issue the following command from the command prompt in your *UDDIServer* directory:

```
UDDIPublish -f <wsdlpath> -p UDDIProperties.properties
```

where **<wsdlpath>** is the path to the **.jar** file containing the **.wsdl** file you want to publish.

**Important:** *The .jar file must have been previously **exported** from Enterprise Designer.*

**Note:** *If the path contains spaces, enclose the entire path in quotation marks; for example, "C:\jar files\testwsdls.jar"*

**Note:** *The order of the [-f <>] and [-p <>] commands can be interchanged.*

- 2 The published file will be located at the following URL:  

```
http://<hostname>:<portnumber>/<context name>
```
- 3 To see this procedure online, issue any of the following commands from the command prompt in your *UDDIServer* directory:

```
UDDIPublish  
UDDIPublish -help  
UDDIPublish -HELP
```

### 12.2.4 Accessing Web Service Definitions

The definitions for all Java™ Composite Application Platform Suite objects that expose themselves as web services, such as Java-based Collaborations or eInsight Business Processes, are listed in a UDDI registry. This UDDI registry complies with the OASIS UDDI v 2.0 specification, has configurable properties, and is run as an independent server.

To access a web service from the UDDI registry, the following components must be installed and set up:

- Sun SeeBeyond UDDI Server (see [UDDI External Systems](#) on page 538).
- Web Services Access Manager. This application used in conjunction with the UDDI Server to configure security policies for web services.

Procedures for installing these components are found in the **Managing Access to Web Services** section of the *Sun SeeBeyond eGate™ Integrator System Administration Guide*.

Once you have been granted user access to the specific web service, you can develop web service client applications using the provided security credentials by using the Web Services Access Manager application in the Enterprise Manager.

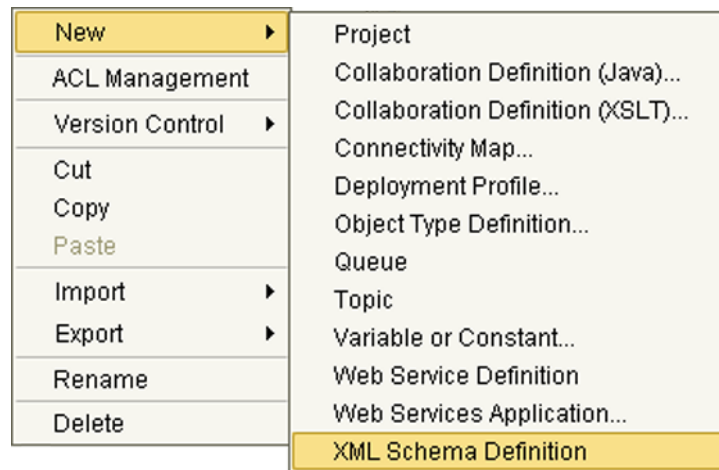
**Figure 350** Web Services Access Manager

WSDL name: <input type="text"/>		<input type="button" value="Search"/>
WSDL	Groups	Users
<a href="#">echoWebservice_BusinessProcess1_8BD52380C8-EBF510A7D9-0154AD8868-1DC9AE5E0F.wsdl</a>	asadmin	Administrator

## 12.3 Creating XML Schema Definitions

Selecting the **New > XML Schema Definition** option from the *Project* context menu (see Figure 351) adds an XSD object to your Project and opens the XML Schema Editor (see [Using the XML Schema Editor](#) on page 419).

**Figure 351** Project Context Menu - New XML Schema Definition



### 12.3.1 Using the Context Menu

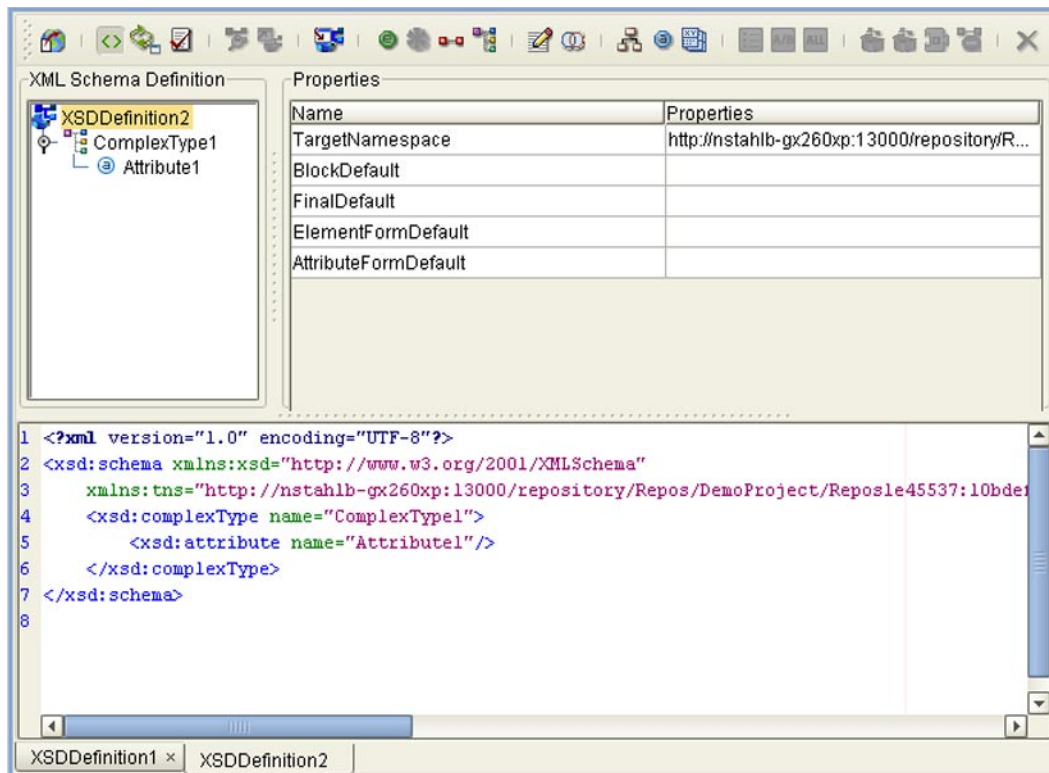
Right-clicking the XSD Object in Project Explorer displays its context menu (see [Using Project Component Context Menus](#) on page 109).

## 12.4 Using the XML Schema Editor

### 12.4.1 Overview

The XML Schema Editor (see Figure 352) is a graphical editor within Enterprise Designer that you can use to develop new, or edit existing, XSD Objects with a minimum of coding.

**Figure 352** XML Schema Editor



The various icons and context menu options are enabled only when the resulting action is allowed or appropriate. If the resulting action is illegal or inappropriate, the icon or option is disabled.

**Note:** Although the user interface for XML Schema Editor contains features that help prevent the creation of invalid code, you need a good understanding of the XML Schema specification to use the editor effectively.

## Toolbar

The first three icons in the toolbar allow you to display or hide the XML source code, build the XML code based on the graphical constructs, and validate the XML code. The remaining icons allow you to build the XML object graphically. See [XML Schema Editor Toolbar Icons](#) on page 421 for detailed descriptions.

## XML Schema Definition Panel

Selecting a node in the *XML Schema Definition* panel displays the properties for that node in the *Properties* panel. These are detailed in [Building an XML Schema Definition](#) on page 428.

## Properties Panel

The *Properties* panel displays the various properties defined for the selected node. Most *Value* fields are automatically populated with default values, and most fields are editable. These features are detailed individually in [Building an XML Schema Definition](#) on page 428.

## XML Source Code Panel

The source code panel displays the XML code for the XML schema definition, and provides a text editor to supplement the XML Schema Editor's graphical capabilities. When editing the XML code in the source code panel, you need to synchronize the definition in the Repository with your edits by clicking the *Refresh Code* icon in the toolbar (see [XML Schema Editor Toolbar Icons](#) on page 421).









The source code panel is hidden by default; you must enable it by clicking the *Show Code* icon shown in [XML Schema Editor Toolbar Icons](#) on page 421.













## 12.4.2 XML Schema Editor Toolbar Icons

Icons in the XML Schema Editor's toolbar provide basic control over the editor and provide shortcuts for adding the most often used XML components to the XSD. These shortcut icons duplicate the corresponding options on the component's context (right-click) menu, but can be quicker to use.







**Table 94** XML Schema Editor Toolbar Icons

Icon	Command	Function
	Export XSD to File	Presents a dialog in which you can save the serialized XML Schema Definition to your local file system as an <b>.xsd</b> file.
	Show Code	Displays (or hides) the XML source code in a panel located in the lower part of the XML Schema Editor.
	Refresh Code	Refreshes the XSD from the Repository.
	Validate XML Schema Definition	Validates the XSD and displays the results in a dialog.
	Preview XSD Node	Presents a dialog containing allowed options; clicking <b>OK</b> in the dialog opens the OTD Editor, displaying the selected top-level Element, ComplexType, or SimpleType and its substructure. (To return to XSD Schema Editor, use Window menu.) See <a href="#">Previewing and Exporting XSD Nodes</a> on page 424.
	Export XSD Node	Presents a dialog containing allowed options; clicking <b>OK</b> in the dialog exports the selected top-level Element, ComplexType, or SimpleType as an <i>XSD Node</i> , which can be used in a Collaboration Definition (Java). See <a href="#">Previewing and Exporting XSD Nodes</a> on page 424.
	Import XSD	Opens a dialog (see <a href="#">Figure 358 on page 431</a> ) in which you can import an XSD file to the currently open XSD (duplicates the action of the <b>Add Import</b> option on the context menu for the root node).
	Add Element	Adds an <b>xsd:element</b> sub-node to the selected XSD root, XML sequence, XML choice, or XML all node (duplicates the action of the <b>Add Element</b> option on the context menus for those nodes). See <a href="#">Element Node</a> on page 437.

**Table 94** XML Schema Editor Toolbar Icons

Icon	Command	Function
	Add Any	Adds an XML <b>any</b> sub-node to the selected XSD root node, and to XML sequence or XML choice nodes when they appear as sub-nodes of XML complexType or XML restriction nodes (duplicates the action of the <b>Add Any</b> option on the context menus for those nodes). See <a href="#">Any Node</a> on page 455.
	Add SimpleType	Adds an XML <b>simple type</b> sub-node to the selected XSD root node (duplicates the action of the <b>Add Simple Type</b> option on the XSD root node context menu). See <a href="#">SimpleType Node</a> on page 446.
	Add ComplexType	Adds an XML <b>complex type</b> sub-node to the selected XSD root node (duplicates the action of the <b>Add ComplexType</b> option on the XSD root node context menu). See <a href="#">ComplexType Node</a> on page 444.
	Add ListType	Adds an XML <b>listType</b> sub-node to the selected XSD root node (duplicates the action of the <b>Add ListType</b> option on the context menus for those nodes). See <a href="#">ListType Node</a> on page 451.
	Add Union	Adds an XML <b>union</b> sub-node to the selected XSD root node (duplicates the action of the <b>Add Union</b> option on the context menus for those nodes). See <a href="#">Union Node</a> on page 441.
	Add Group	Adds an XML <b>group</b> sub-node to the selected XSD root node (duplicates the action of the <b>Add Group</b> option on the XSD root node context menu). See <a href="#">Group Node</a> on page 443.
	Add Attribute	Adds an XML <b>attribute</b> sub-node to the selected XSD root, XML complex type, or XML attribute group node (duplicates the action of the <b>Add Attribute</b> option on the context menus for those nodes). See <a href="#">Attribute Node</a> on page 433.
	Add AttributeGroup	Adds an XML <b>attributeGroup</b> sub-node to the selected XSD root node (duplicates the action of the <b>Add AttributeGroup</b> option on the context menus for those nodes). See <a href="#">AttributeGroup Node</a> on page 435.
	Add Sequence	Adds an XML <b>sequence</b> sub-node to the selected XML complex type, XML group, or XML restriction node (duplicates the action of the <b>Add Sequence</b> option on the context menus for those nodes). See <a href="#">Sequence Node</a> on page 452.
	Add Choice	Adds an XML <b>choice</b> sub-node to the selected XML complex type, XML group, or XML restriction node (duplicates the action of the <b>Add Choice</b> option on the context menus for those nodes). See <a href="#">Choice Node</a> on page 453.

**Table 94** XML Schema Editor Toolbar Icons

Icon	Command	Function
	Add All	Adds an XML <b>all</b> sub-node to the selected XML complex type, XML group, or XML restriction node (duplicates the action of the <b>Add All</b> option on the context menus for those nodes). See <a href="#">All Node</a> on page 454.
	Add SimpleContent	Adds an XML <b>simpleContent</b> sub-node to the selected XML complexType node (duplicates the action of the <b>Add SimpleContent</b> option on the context menus for those nodes). See <a href="#">SimpleContent Node</a> on page 458.
	Add ComplexContent	Adds an XML <b>complexContent</b> sub-node to the selected XSD complexType node (duplicates the action of the <b>Add ComplexContent</b> option on the context menus for those nodes). See <a href="#">ComplexContent Node</a> on page 457.
	Add Extension	Adds an XML <b>extension</b> sub-node to the selected XSD simpleContent or XML complexContent node (duplicates the action of the <b>Add Extension</b> option on the context menus for those nodes). See <a href="#">Extension Node</a> on page 459.
	Add Restriction	Adds an XML <b>restriction</b> sub-node to the selected XSD complexContent node (duplicates the action of the <b>Add Restriction</b> option on the context menus for those nodes). See <a href="#">Restriction Node</a> on page 460.
	Delete	Deletes the selected node from the XSD tree, including all sub-nodes.

### 12.4.3 Previewing and Exporting XSD Nodes

In both the XML Schema Editor and the Web Services Editor, you can define global schema components (Elements, SimpleTypes, and ComplexTypes) as *XSD Nodes*. These XSD Nodes serve as pointers to the global components in XML Schema Definitions, or inline schemas in Web Service Definitions, and behave in the same manner as OTDs. Hence, they can be previewed in the OTD Editor and their runtime behavior can be examined in the OTD Tester. They also can be exported for use in a Collaboration Definition (Java) or to form a WSDL message in the Business Process Editor (see the *Sun SeeBeyond eInsight™ Business Process Manager User's Guide*).

**Note:** *Exporting actually creates an XSD Node, while previewing does not.*

XSD Nodes are subject to the following rules and restrictions:

- XSD Nodes cannot be exported unless a target namespace is declared in the XSD or the inline schema of the WSDL. Previewing is generally allowed, however.
- After the XSD Node is exported, changing either the target namespace of the schema, or the name of the global component to which the XSD Node refers, breaks the link between the XSD Node and the XSD or WSDL. When this occurs, the XSD Node can no longer be opened, and it must be re-exported from its source.
- When you export an XSD Node, it is automatically named according to the format:

`<XSD/WSDL_Name>_<Global_Component_Name>`

If a previously exported XSD Node with the same name already exists, the content of the existing XSD Node is overwritten. If there is an existing Project component of a different type having the same name, you are presented with a dialog asking for an alternative name for the XSD Node.

### Preview/Export Options

**Note:** *The content of the XSD Node Export Options and XSD Node Preview Options dialogs is identical.*

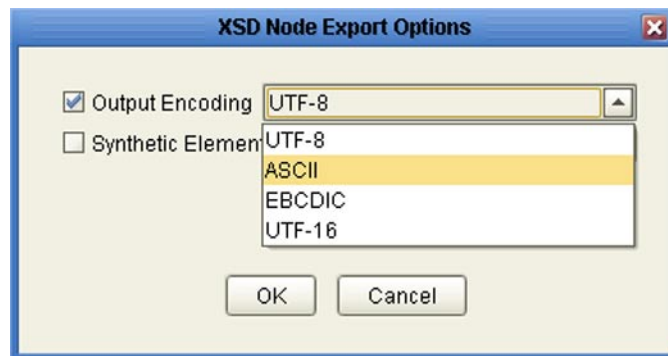
**Table 95** XSD Node Preview/Export Options

Name	Description
Output Encoding	Specifies the output encoding. See <a href="#">Specifying Data Conversion and Encoding</a> on page 151. Option is enabled only when the extended language option is enabled (see <a href="#">Setting Up Options</a> on page 53).
Synthetic Element Name	Allows you to specify a synthetic element name for the selected ComplexType or SimpleType. Option is disabled for Elements, which always use their local name.

## Output Encoding

If you have enabled the extended language option (see [Setting Up Options](#) on page 53), you can specify the output data encoding for the XSD Node you are previewing or exporting. When you select a global Element, SimpleType, or ComplexType to preview or export as an XSD Node, you are presented with the *XSD Node Preview (or Export) Options* dialog. Selecting **Output Encoding** enables the drop-down list of encoding formats (see Figure 353), where you can select an alternate data encoding.

**Figure 353** XSD Node Preview/Export Options - Output Encoding

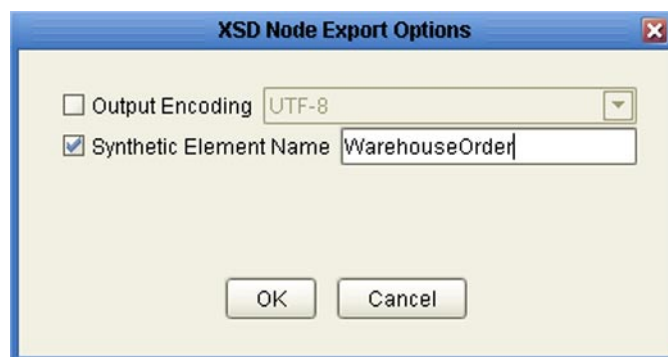


## Synthetic Element Name

When you form an XSD Node from a global XML type definition (SimpleType or ComplexType), it must be given a qualified name (a "synthetic element name"), since the type definition only specifies the content. By default, it is assigned a name based on its type, such as *SimpleType1*, for example. In most cases, you will want to assign a more meaningful name to the XSD Node.

When you select a global XML type definition to preview or export as an XSD Node, you are presented with the *XSD Node Preview (or Export) Options* dialog. Selecting **Synthetic Element Name** allows you to enter an appropriate name for the XSD Node (see Figure 354). (In the case of a global Element, the Element's local name is always used, since there is no need to define a synthetic element name.)

**Figure 354** XSD Node Preview/Export Options - Synthetic Element Name



## 12.4.4 XML Schema Compilation

There are several conditions that cause compilation errors to occur when the validation rules are strictly enforced. When these occur, the XSD Nodes are usually still valid for their intended purpose, but the strictness of the parser must be relaxed for the application file to build properly (*codegen* process).

The **Tools > Options** menu in the Enterprise Designer menu bar displays a dialog (see [Preferences](#) on page 55) containing options that let you specify whether specific XML Schema validation rules are set to be *lenient* or *strict*, as described below.

### Allow duplicated global schema components

In many situations, such as when a WSDL has an inline schema that is exactly the same as an imported schema, global components are duplicated. As long as the duplication is *exact*, no problem should be encountered when using the WSDL. When you validate a WSDL or an XSD in the respective editor, issues of duplicated global schema components will either show up as errors or as warnings, depending on how the option is set. By default, the option is checked, and duplicated global schema components are allowed.

**Table 96** Duplicated Global Schema Component Options

Option	Description
Check (True)	This is the lenient (default) mode, in which any duplicates of previously-encountered global components are ignored. When these duplicates are found, warnings are logged to the <b>ide.log</b> and the codegen process continues.
Clear (False)	This is the strict mode, which will cause codegen to fail if any duplicated global schema components are encountered. Errors are logged to the <b>ide.log</b> and the codegen process stops.

### Enforce Unique Particle Attribution rule

The unique particle attribution (UPA) rule requires that the content model be formed such that during validation of an XML element, the particle component related to this element is uniquely determined without examining the content or attributes of that element, and without any information about the XML elements in the remainder of the sequence. If the particle component is not uniquely determined as required, the schema violates the rule. You can select whether or not to enforce the UPA rule when compiling the schema; by default, the option is not checked, and the rule is not enforced.

**Table 97** Enforce UPA Rule Options

Option	Description
Check (True)	This is the strict mode, in which the UPA rule will be enforced, and schemas violating the rule will stop the codegen process.
Clear (False)	This is the lenient (default) mode, in which the UPA rule will <i>not</i> be enforced, and code will still be generated even when schemas violating the UPA rule are present.

This option sets the `xsdcodegen.checkupa` switch in the `runed.bat` file, which you can also set using a text editor.

### Enforce Particle Restriction rules

The particle restriction rule states, basically, that all instances that are valid with respect to the restricted type must also be valid with respect to the base type. If a restricted type is defined in a way that conflicts with the definition of the base type, the schema will violate the particle restriction rule. You can select whether or not to enforce the particle restriction rule when compiling the schema; by default, the option is checked, and the rule is enforced.

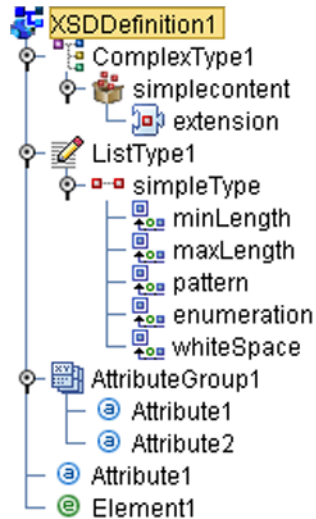
**Table 98** Enforce Particle Restriction Rule Options

Option	Description
Check (True)	This is the strict (default) mode, in which the particle restriction rule will be enforced, and schemas violating the rule will stop the codegen process.
Clear (False)	This is the lenient mode, in which the particle restriction rule will <i>not</i> be enforced, and code will still be generated even when schemas violating the particle restriction rule are present.

### 12.4.5 Building an XML Schema Definition

An XSD is represented in the XML Schema Editor by a tree structure with a hierarchy of nodes, consistent with the representations in other Enterprise Designer editors. You can build XSD structures graphically by adding nodes to the tree, beginning with the root node, using the **Add** option in the individual node context menus or the toolbar icons. Figure 355 shows an XSD tree structure illustrating some basic node combinations.

**Figure 355** XSD Tree Example





## XML Schema Definition Root Node

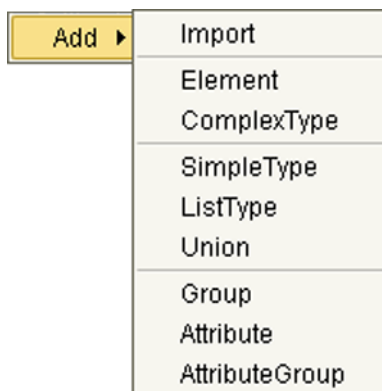
The XSD root node represents the `<xsd:schema>` element, to which a variety of other elements can be added as sub-nodes. These elements are listed as options in the root node's context menu, as shown in Figure 357. No properties are shown for the root node.

**Figure 356** XSD Root Node Icon



### Context Menu

**Figure 357** XSD Root Node Menu



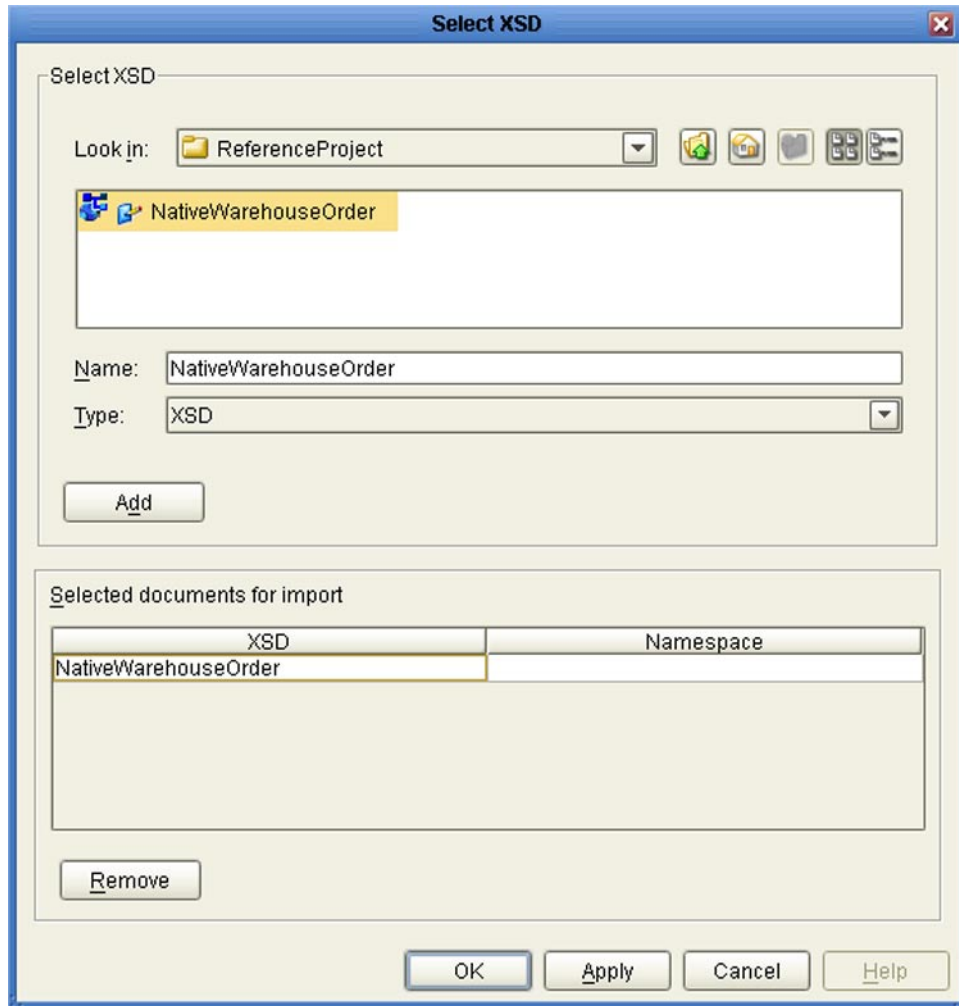
**Table 99** XSD Root Node Menu Options

Option		Action
Add	Import	Opens a dialog (see Figure 358) in which you can import an XSD file as a sub-node to selected XSD root node (duplicates the action of the <b>Import XSD</b> toolbar icon).
	Element	Adds an <b>Element</b> sub-node to the selected XSD root node (duplicates the action of the <b>Add Element</b> toolbar icon). An element at this level becomes a <i>global</i> element. See <a href="#">Element Node</a> on page 437.
	ComplexType	Adds an <b>ComplexType</b> sub-node to the selected XSD root node (duplicates the action of the <b>Add ComplexType</b> toolbar icon). See <a href="#">ComplexType Node</a> on page 444.
	SimpleType	Adds an <b>SimpleType</b> sub-node to the selected XSD root node (duplicates the action of the <b>Add SimpleType</b> toolbar icon). See <a href="#">SimpleType Node</a> on page 446.

**Table 99** XSD Root Node Menu Options

Option		Action
Add	ListType	Adds an <b>ListType</b> sub-node to the selected XSD root node. See <a href="#">ListType Node</a> on page 451.
	Union	Adds an <b>Union</b> sub-node to the selected XSD root node. See <a href="#">Union Node</a> on page 441.
	Group	Adds an <b>Group</b> sub-node to the selected XSD root node (duplicates the action of the <b>Add Group</b> toolbar icon). See <a href="#">Group Node</a> on page 443.
	Attribute	Adds an <b>Attribute</b> sub-node to the selected XSD root node (duplicates the action of the <b>Add Attribute</b> toolbar icon). An attribute at this level becomes a <i>global</i> attribute. See <a href="#">Attribute Node</a> on page 433.
	Attribute Group	Adds an <b>AttributeGroup</b> sub-node to the selected XSD root node. See <a href="#">AttributeGroup Node</a> on page 435.

**Figure 358** Select XSD Dialog



### Properties

The set of properties associated with the XSD root node is shown in Figure 361.

**Figure 359** XSD Root Node Properties



**Note:** Clicking the Value field for a property enables it, after which you can edit the text. For exceptions to this rule, see the following table.

**Table 100** XSD Root Node Property Options

Name	Description
targetNamespace	The namespace within which the names in this node belong.
blockDefault	Selecting the field and clicking the ellipsis (...) button displays a dialog containing a list of existing block types within the target namespace.
finalDefault	Selecting the field and clicking the ellipsis (...) button displays a dialog containing a list of existing block types within the target namespace.
elementFormDefault	Specifies whether the elementFormDefault is <b>qualified</b> or <b>unqualified</b> .
attributeFormDefault	Specifies whether the attributeFormDefault is <b>qualified</b> or <b>unqualified</b> .

## Attribute Node

The **Attribute** node corresponds to the `<xsd:attribute>` element, which represents an attribute declaration.

**Figure 360** Attribute Node Icon



### Context Menu

This node cannot have sub-nodes assigned to it; its context menu only offers the **Delete** option.

### Properties

The set of properties associated with a non-global *Attribute* node is shown in Figure 361; global *Attribute* nodes have a subset of these properties associated with them, as described in Table 101.

**Figure 361** Attribute Node Properties

Properties	
Name	Properties
name	Attribute1
type	<input type="text"/> ...
referenceUsed	false
form	
use	optional
default	
fixed	

**Note:** Clicking the Value field for a property enables it, after which you can edit the text. For exceptions to this rule, see the following table.

**Table 101** Attribute Node Property Options

Name	Description
name	Name of the <b>Attribute</b> node.
type	Instead of assigning a new name to this node, you can reference an existing attribute name. If the <b>referenceUsed</b> property is set to <b>true</b> , selecting the field and clicking the ellipsis (...) button displays a dialog containing a list of existing attributes within the target namespace. This property and the <b>name</b> property are mutually exclusive.

Name	Description
referenceUsed	Specifies whether or not this element is named by reference. Setting this property to <b>true</b> enables the <b>type</b> property value field (see below). <b>Note:</b> Does not apply to global Attribute nodes.
form	Specifies whether or not this (local) attribute must be <b>qualified</b> or <b>unqualified</b> (or neither). <b>Note:</b> Does not apply to global Attribute nodes.
use	Specifies whether the node is optional, required, or prohibited. <ul style="list-style-type: none"> <li>▪ <b>optional</b> specifies that the <code>&lt;xsd:attribute&gt;</code> element is optional and may have any value.</li> <li>▪ <b>required</b> specifies that the <code>&lt;xsd:attribute&gt;</code> element is required and may have any value.</li> <li>▪ <b>prohibited</b> specifies that the <code>&lt;xsd:attribute&gt;</code> element must not appear.</li> </ul> <b>Note:</b> Does not apply to global Attribute nodes.
default	If the <b>use</b> property is specified to be <b>optional</b> , the default value applies when <code>&lt;xsd:attribute&gt;</code> is missing from the instance document; otherwise, the value in the instance document applies. The <b>default</b> and <b>fixed</b> properties are mutually exclusive — you can declare values for one or the other, not both.
fixed	If the <b>use</b> property is specified to be <b>optional</b> , the fixed value is always used for <code>&lt;xsd:attribute&gt;</code> whether or not it is missing from the instance document. The <b>default</b> and <b>fixed</b> properties are mutually exclusive — you can declare values for one or the other, not both.

## AttributeGroup Node

The **AttributeGroup** node corresponds to the `<xsd:attributeGroup>` element, which is used to contain multiple attributes which can then be referenced as a group elsewhere in the XSD.

**Figure 362** Attribute Group Node Icon



### Context Menu

**Figure 363** AttributeGroup Node Menu



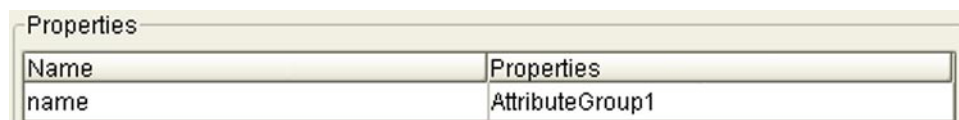
**Table 102** AttributeGroup Node Menu Options

Option		Action
Add	Attribute	Adds an <b>Attribute</b> sub-node to the selected <b>AttributeGroup</b> node (duplicates the action of the <b>Add Attribute</b> toolbar icon). See <a href="#">Attribute Node</a> on page 433.
	Attribute Group	Adds an <b>AttributeGroup</b> sub-node to the selected <b>ComplexType</b> node. See <a href="#">AttributeGroup Node</a> on page 435.
Delete		Deletes the selected node from the XSD tree, including all sub-nodes.

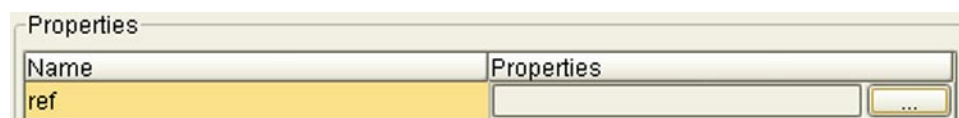
### Properties

The properties associated with *AttributeGroup* are shown in Figure 364 and Figure 365.

**Figure 364** AttributeGroup Node Properties (Global)



**Figure 365** AttributeGroup Node Properties (Non-global)

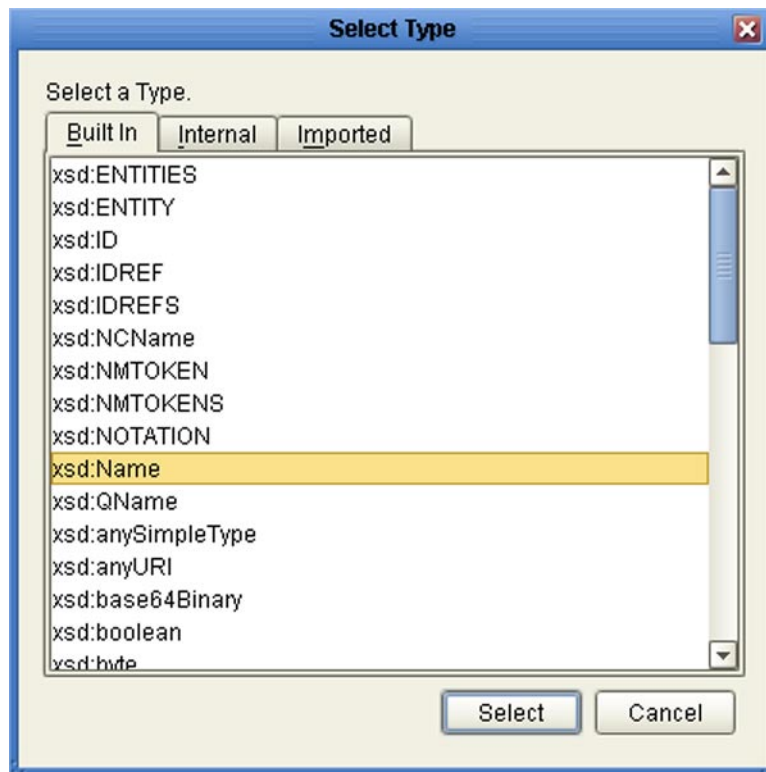


**Note:** Clicking the Value field for a property enables it, after which you can edit the text. For exceptions to this rule, see the following table.

**Table 103** AttributeGroup Node Property Options

Name	Description
name	Name of the <b>AttributeGroup</b> node. Parameter is shown only for global AttributeGroups.
ref	If the <b>AttributeGroup</b> node is non-global, it can only reference an existing, global attributeGroup. Selecting the field and clicking the ellipsis (...) button displays a dialog containing a list of existing internally-defined types and imported types. (see Figure 366). Parameter is shown only for non-global AttributeGroups.

**Figure 366** Select Type Dialog





## Element Node

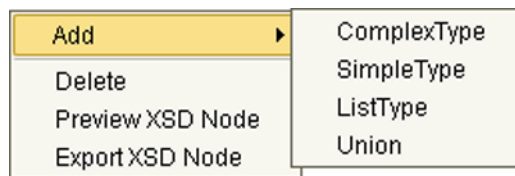
The **Element** node corresponds to the `<xsd:element>` element, which represents an element declaration.

**Figure 367** Element Node Icon



## Context Menu

**Figure 368** Element Node Menu



**Table 104** Element Node Menu Options

Option		Action
Add	ComplexType	Adds an <b>ComplexType</b> sub-node to the selected element node (duplicates the action of the <b>Add ComplexType toolbar</b> icon). See <a href="#">ComplexType Node</a> on page 444.
	SimpleType	Adds an <b>SimpleType</b> sub-node to the selected element node (duplicates the action of the <b>Add SimpleType toolbar</b> icon). See <a href="#">SimpleType Node</a> on page 446.
	ListType	Adds an <b>ListType</b> sub-node to the selected element node. See <a href="#">ListType Node</a> on page 451.
	Union	Adds an <b>Union</b> sub-node to the selected element node. See <a href="#">Union Node</a> on page 441.
Delete		Deletes the selected node from the XSD tree, including all sub-nodes.
Preview XSD Node		For top-level nodes only, opens OTD Editor, displaying the in-process XSD Node structure. (To return to the XSD Schema Designer, use the <i>Window</i> menu.)
Export XSD Node		For top-level nodes only, exports the node and its sub-nodes as an XSD Node. The XSD root (parent) node must have a declared targetNamespace.

## Properties

The sets of properties associated with the *Element* node are shown in Figure 369 (global) and Figure 370 (non-global).

**Figure 369** Element Node Properties (Global)

Name	Value
Name	Element1
Type	<input type="text"/> ...
abstract	false
default	
fixed	
nillable	false
substitutionGroupUsed	false
substitutionGroup	

**Note:** Clicking the Value field for a property enables it, after which you can edit the text. For exceptions to this rule, see the following table.

**Table 105** Element Node Property Options (Global)

Name	Description
Name	Name of the <b>Element</b> node.
Type	Specifies the existing type upon which the node is based. Selecting the field and clicking the ellipsis (...) button displays a dialog containing lists of built-in types, internally-defined types, and imported types.
abstract	
default	If a default value is given, the value is used when the <b>Element</b> node is empty in the instance document. <b>Note:</b> The <b>default</b> and <b>fixed</b> properties are mutually exclusive — you can declare values for one or the other, not both.
fixed	If a fixed value is given, the <b>Element</b> node is defined as optional, and the value is always used for the node whether or not it appears in the instance document. <b>Note:</b> The <b>default</b> and <b>fixed</b> properties are mutually exclusive — you can declare values for one or the other, not both.
nillable	Setting this property to <b>true</b> specifies that the element is allowed to contain no value.
substitutionGroupUsed	Specifies whether or not this element can be substituted for another element. Setting this property to <b>true</b> enables the <b>substitutionGroup</b> property value field (see below).

Name	Description
substitutionGroup	If the <b>substitutionGroupUsed</b> property is set to <b>true</b> , selecting the field and clicking the ellipsis (...) button displays a dialog containing a lists of elements for which this element can be substituted.

**Figure 370** Element Node Properties (Non-global)

Properties	
Name	Element1
referenceUsed	false
targetNamespace	http://gx260xp:12000/repository/R...
minOccurs	1
maxOccurs	1
type	<input type="text"/> ...
default	
fixed	
nillable	false

**Note:** Clicking the Value field for a property enables it, after which you can edit the text. For exceptions to this rule, see the following table.

**Table 106** Element Node Property Options

Name	Description
name	Name of the <b>Element</b> node.
referenceUsed	Specifies whether or not this element is named by reference. Setting this property to <b>true</b> enables the <b>type</b> property value field (see below).
targetNamespace	The namespace within which the names in this (local) element belong.
minOccurs	Specifies the minimum number of times this element may appear; the default value is <b>1</b> .
maxOccurs	Specifies the maximum number of times this element may appear; the default value is <b>1</b> .

Name	Description
type	Instead of assigning a new name to this node, you can reference an existing element name. If the <b>referenceUsed</b> property is set to <b>true</b> , selecting the field and clicking the button displays a dialog containing a list of existing elements within the target namespace. <b>Note:</b> This property and the <b>name</b> property are mutually exclusive.
default	If a default value is given, the value is used when the <b>Element</b> node is empty in the instance document. <b>Note:</b> The <b>default</b> and <b>fixed</b> properties are mutually exclusive — you can declare values for one or the other, not both.
fixed	If a fixed value is given, the <b>Element</b> node is defined as optional, and the value is always used for the node whether or not it appears in the instance document. <b>Note:</b> The <b>default</b> and <b>fixed</b> properties are mutually exclusive — you can declare values for one or the other, not both.
nillable	Setting this property to <b>true</b> specifies that the element is allowed to contain no value.

## Union Node

The **Union** node corresponds to the `<xsd:union>` element, which enables the value of an element or attribute to be an instance of a type drawn from the union of multiple simple or list types.

**Figure 371** Union Node Icon



### Context Menu

This node cannot have sub-nodes assigned to it; its context menu only offers the **Delete** option.

### Properties

The set of properties associated with *Union* is shown in Figure 372.

**Figure 372** Union Node Properties

Name	Properties
name	Union1
memberType	<input type="text"/> ...

**Note:** Clicking the Value field for the *name* property enables it, after which you can edit the text. Double-clicking the Value field for the *memberType* property enables it; clicking the ellipsis (...) button displays the dialog shown in Figure 373.

**Table 107** Union Node Property Options

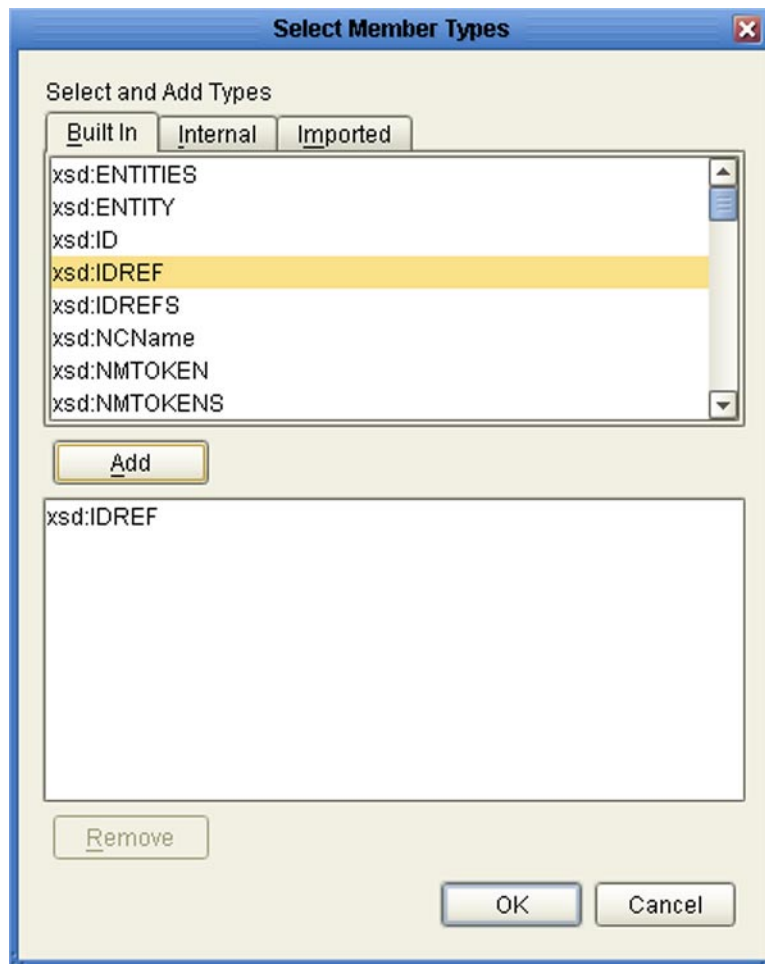
Name	Description
name	For global nodes only, name of the <b>Union</b> node.
memberType	A list of all types in the union. Double-clicking the value field and clicking the ellipsis (...) button displays a dialog containing a list of existing simple types, from which you can build a list for the union.

The dialog shown in Figure 373 shows all existing simple types defined for the XSD, displayed in the upper panel.

### To list types to include in the union

- 1 Select one type at a time and click **Add**, which adds the type to the list in the lower panel.
- 2 When you have selected the types you want for the union, click **OK**. You can return to the dialog to add other types at a later time.

**Figure 373** Select Member Types Dialog



## Group Node

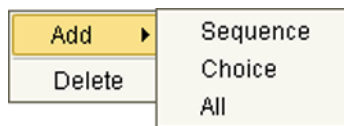
The **Group** node corresponds to the `<xsd:group>` element, which is used to contain multiple groups which can then be referenced as a single group elsewhere in the XSD. The groups can be of the `<xsd:sequence>`, `<xsd:choice>`, or `<xsd:all>` types.

**Figure 374** Group Node Icon



## Context Menu

**Figure 375** Group Node Menu



**Table 108** Group Node Menu Options

Option		Action
Add	Sequence	Adds a <b>Sequence</b> sub-node to the selected Group node (duplicates the action of the <b>Add Sequence</b> toolbar icon). See <a href="#">Sequence Node</a> on page 452.
	Choice	Adds a <b>Choice</b> sub-node to the selected Group node (duplicates the action of the <b>Add Choice</b> toolbar icon). See <a href="#">Choice Node</a> on page 453.
	All	Adds an <b>All</b> sub-node to the selected Group node (duplicates the action of the <b>Add All</b> toolbar icon). See <a href="#">All Node</a> on page 454.
Delete		Deletes the selected node from the XSD tree, including all sub-nodes.

## Properties

The only property associated with the *Group* node is the name of the node.

## ComplexType Node

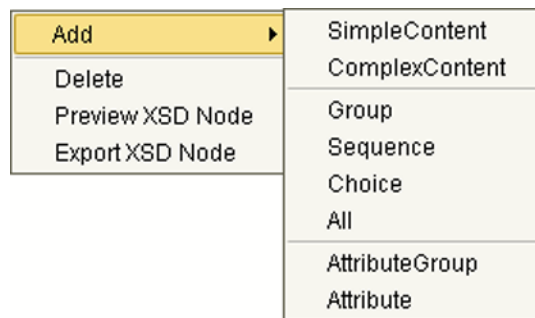
The **ComplexType** node corresponds to the `<xsd:complexType>` element, used for defining complex types.

**Figure 376** ComplexType Node Icon



### Context Menu

**Figure 377** ComplexType Node Menu



**Table 109** ComplexType Node Menu Options

Option		Action
Add	Simple Content	Adds a <b>SimpleContent</b> sub-node to the selected ComplexType node. See <a href="#">SimpleContent Node</a> on page 458.
	Complex Content	Adds a <b>ComplexContent</b> sub-node to the selected ComplexType node. See <a href="#">ComplexContent Node</a> on page 457.
	Group	Adds a <b>Group</b> sub-node to the selected ComplexType node (duplicates the action of the <b>Add Group</b> toolbar icon). See <a href="#">Group Node</a> on page 443.
	Sequence	Adds a <b>Sequence</b> sub-node to the selected ComplexType node (duplicates the action of the <b>Add Sequence</b> toolbar icon). See <a href="#">Sequence Node</a> on page 452.
	Choice	Adds a <b>Choice</b> sub-node to the selected ComplexType node (duplicates the action of the <b>Add Choice</b> toolbar icon). See <a href="#">Choice Node</a> on page 453.
	All	Adds an <b>All</b> sub-node to the selected ComplexType node (duplicates the action of the <b>Add All</b> toolbar icon). See <a href="#">All Node</a> on page 454.



**Table 109** ComplexType Node Menu Options

Option		Action
Add	Attribute Group	Adds an <b>AttributeGroup</b> sub-node to the selected ComplexType node. See <a href="#">AttributeGroup Node</a> on page 435.
	Attribute	Adds an <b>Attribute</b> sub-node to the selected ComplexType node (duplicates the action of the <b>Add Attribute</b> toolbar icon). See <a href="#">Attribute Node</a> on page 433.
Delete		Deletes the selected node from the XSD tree, including all sub-nodes.
Preview XSD Node		For top-level nodes only, opens OTD Editor, displaying the in-process XSD Node structure. (To return to the XSD Schema Designer, use the <i>Window</i> menu.)
Export XSD Node		For top-level nodes only, exports the node and its sub-nodes as an XSD Node. The XSD root (parent) node must have a declared targetNamespace.

### Properties

The only property associated with the *ComplexType* node is the name of the node.

## SimpleType Node

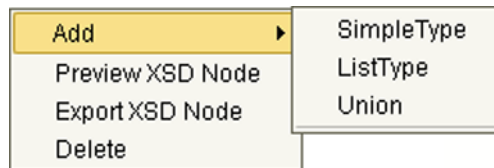
The **SimpleType** node corresponds to the `<xsd:simpleType>` element, and is used to define a new simple type. The new type is derived from an existing simple type by restricting the existing type; therefore, the range of values for new simple type is a subset of the range of values for the existing (base) type. The restrictions are specified by assigning values to *facets*, which constrain the range of values for the new type.

**Figure 378** SimpleType Node Icon



### Context Menu

**Figure 379** SimpleType Node Menu



**Table 110** SimpleType Node Menu Options

Option		Action
Add	SimpleType	Adds an <b>SimpleType</b> sub-node to the selected node (duplicates the action of the <b>Add SimpleType</b> toolbar icon). The sub-node then restricts the parent SimpleType node.
	ListType	Adds an <b>ListType</b> sub-node to the selected element node. See <a href="#">ListType Node</a> on page 451.
	Union	Adds an <b>Union</b> sub-node to the selected element node. See <a href="#">Union Node</a> on page 441.
Preview XSD Node		For top-level nodes only, opens OTD Editor, displaying the in-process XSD Node structure. (To return to the XSD Schema Designer, use the <i>Window</i> menu.)
Export XSD Node		For top-level nodes only, exports the node and its sub-nodes as an XSD Node. The XSD root (parent) node must have a declared targetNamespace.
Delete		Deletes the selected node from the XSD tree, including all sub-nodes.

### Properties

The set of properties associated with *SimpleType* is shown in Figure 380.

**Figure 380** SimpleType Node Properties

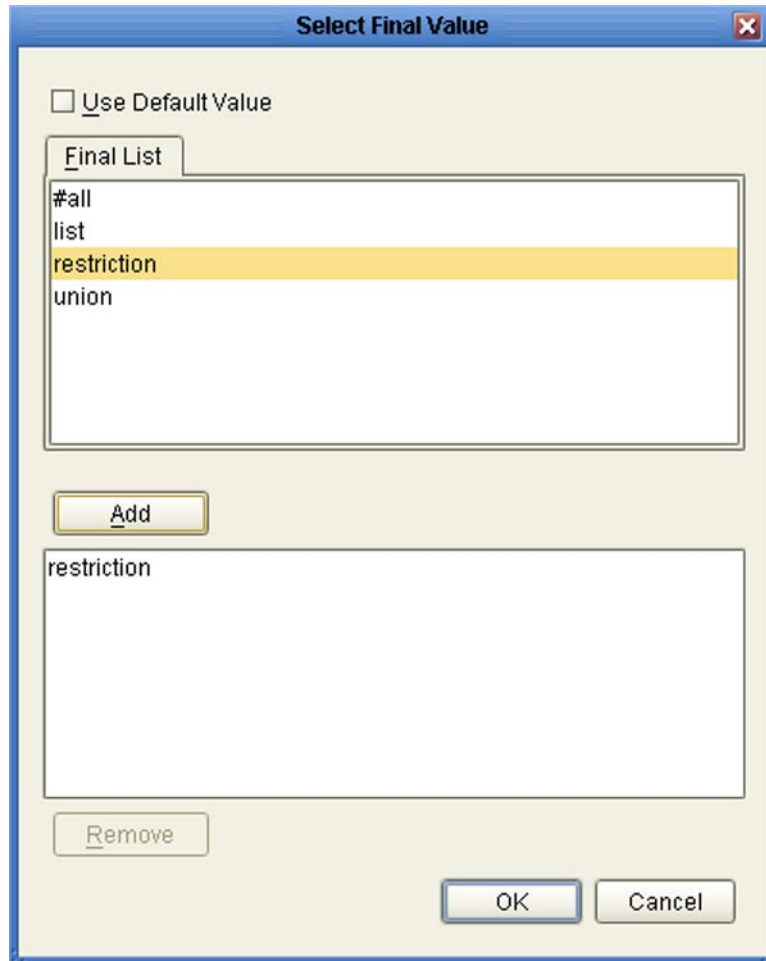
Name	Properties
name	SimpleType1
restriction base	<input type="text"/> ...
final	

**Note:** Clicking the Value field for a property enables it, after which you can edit the text. For exceptions to this rule, see the following table.

**Table 111** SimpleType Node Property Options

Name	Description
name	For global nodes only, name of the <b>SimpleType</b> node.
restriction base	Specifies the existing type upon which the declared restriction is based. Selecting the field and clicking the ellipsis (...) button displays a dialog containing lists of built-in simple types, internally-defined types, and imported types.
final	For global nodes only, double-clicking the value field and clicking the ellipsis (...) button displays a dialog in which you can select a final value (see Figure 381).

**Figure 381** Select Final Value Dialog



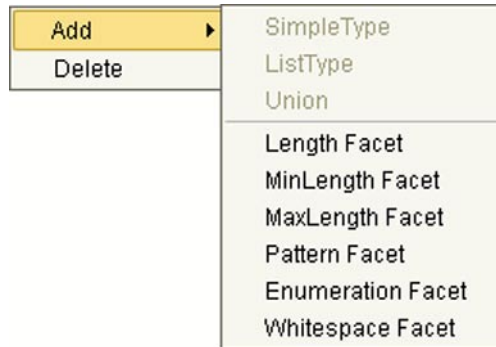
Deselecting the **Use Default Value** check box allows you to select multiple items from the *Final List*. Click **OK** to use the selected values.

## SimpleType Subnode

When added as a sub-node to another SimpleType, a ListType, an Attribute, or a Union, a SimpleType presents a different context menu and property set.

### Context Menu

**Figure 382** SimpleType Subnode Menu

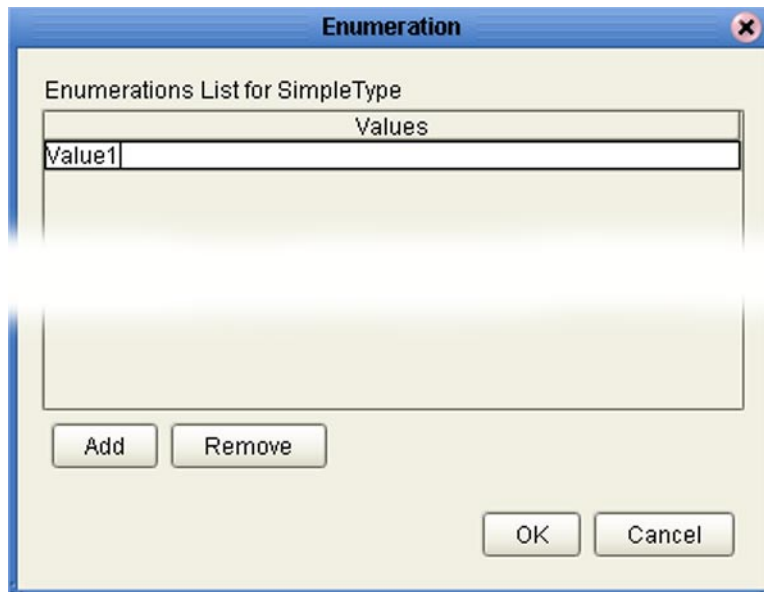


**Table 112** SimpleType Subnode Menu Options

Option		Action
Add	Length Facet	Specifying a value for this facet constrains the allowed value of the new type to a specific length (number of bytes).
	MinLength Facet	Specifying a value for this facet sets a minimum length (number of bytes) for the allowed value of the new type.
	MaxLength Facet	Specifying a value for this facet sets a maximum length (number of bytes) for the allowed value of the new type.
	Pattern Facet	This facet constrains the allowed value of the new type to some pattern appropriate to the type, as defined by a regular expression.
	Enumeration Facet	This facet constrains the allowed value of the new type to a set of specific values. Double-clicking the value field and clicking the ellipsis (...) button displays a dialog in which you can build a list of values (see Figure 383).

**Table 112** SimpleType Subnode Menu Options

Option		Action
Add	Whitespace Facet	<p>This facet specifies how the blank spaces (whitespace) in the initial value of the new simple type are treated (normalized) to produce a normalized value.</p> <ul style="list-style-type: none"><li>▪ <b>preserve</b> specifies that no normalization is to be performed, thereby defining the normalized value to be the initial value.</li><li>▪ <b>replace</b> specifies that all occurrences of <b>tab</b> (#x9), <b>line feed</b> (#xA), and <b>carriage return</b> (#xD) be replaced by a <b>space</b> (#x20).</li><li>▪ <b>collapse</b> specifies that a <b>replace</b> be performed, following which contiguous sequences of spaces be collapsed to a single space, and initial and/or final spaces be deleted.</li></ul>
Delete		Deletes the selected node from the XSD tree, including all sub-nodes.

**Figure 383** Enumeration Dialog

In the dialog shown in Figure 383, Click **Add** to open a field, then double-click the field to enable the field for editing. Then enter an appropriate value for the simple type. Clicking **Remove** deletes the selected entry.

### Properties

The only property associated with the *SimpleType* sub-node is the **restriction base** (see [Table 111 on page 447](#)).

## ListType Node

The **ListType** node corresponds to a special case of the `<xsd:simpleType>` element, which contains a sequence of simple (atomic) types.

**Figure 384** List Type Node Icon



### Context Menu

This node cannot have sub-nodes assigned to it; its context menu only offers the **Delete** option.

### Properties

The set of properties associated with *ListType* is shown in Figure 385.

**Figure 385** ListType Node Properties

Properties	
Name	Properties
name	ListType1
itemType	<input type="text"/> ...

**Note:** Clicking the Value field for the *name* property enables it, after which you can edit the text. For the *itemType* property, clicking the ellipsis (...) button displays a dialog in which you can select an allowed existing type.

**Table 113** ListType Node Property Options

Name	Description
name	For global nodes only, name of the <b>ListType</b> node.
itemType	Specifies the existing type upon which the list is based. Selecting the field and clicking the ellipsis (...) button displays a dialog containing lists of built-in simple types, internally-defined types, and imported types. The default is <b>string</b> .

### Properties

The only property associated with the *ListType* node is the name of the node.

## Sequence Node

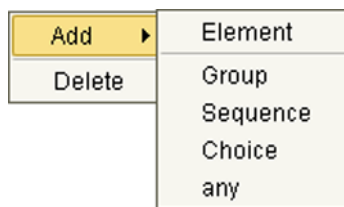
The **Sequence** node corresponds to the `<xsd:sequence>` element, which designates a sequence of elements of the allowed types. The elements in this group must appear in the order specified.

**Figure 386** Sequence Node Icon



## Context Menu

**Figure 387** Sequence Node Menu



**Table 114** Sequence Node Menu Options

Option		Action
Add	Element	Adds an <b>Element</b> sub-node to the selected Sequence node (duplicates the action of the <b>Add Element</b> toolbar icon). See <a href="#">Element Node</a> on page 437.
	Group	Adds a <b>Group</b> sub-node to the selected Sequence node (duplicates the action of the <b>Add Group</b> toolbar icon). See <a href="#">Group Node</a> on page 443.
	Sequence	Adds a <b>Sequence</b> sub-node to the selected Sequence node (duplicates the action of the <b>Add Sequence</b> toolbar icon). See <a href="#">Sequence Node</a> on page 452.
	Choice	Adds a <b>Choice</b> sub-node to the selected Sequence node (duplicates the action of the <b>Add Choice</b> toolbar icon). See <a href="#">Choice Node</a> on page 453.
	Any	Adds an <b>Any</b> sub-node to the selected Sequence node. See <a href="#">Any Node</a> on page 455.
Delete		Deletes the selected node from the XSD tree, including all sub-nodes.

## Properties

None.



## Choice Node

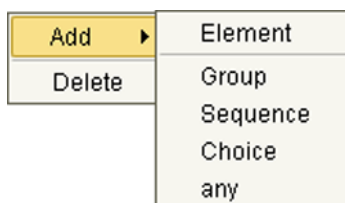
The **Choice** node corresponds to the `<xsd:choice>` element, which groups multiple elements together, only one of which can be chosen.

**Figure 388** Choice Node Icon



## Context Menu

**Figure 389** Choice Node Menu



**Table 115** Choice Node Menu Options

Option		Action
Add	Element	Adds an <b>Element</b> sub-node to the selected Choice node (duplicates the action of the <b>Add Element</b> toolbar icon). See <a href="#">Element Node</a> on page 437.
	Group	Adds a <b>Group</b> sub-node to the selected Choice node (duplicates the action of the <b>Add Group</b> toolbar icon). See <a href="#">Group Node</a> on page 443.
	Sequence	Adds a <b>Sequence</b> sub-node to the selected Choice node (duplicates the action of the <b>Add Sequence</b> toolbar icon). See <a href="#">Sequence Node</a> on page 452.
	Choice	Adds a <b>Choice</b> sub-node to the selected Choice node (duplicates the action of the <b>Add Choice</b> toolbar icon). See <a href="#">Choice Node</a> on page 453.
	Any	Adds an <b>Any</b> sub-node to the selected Choice node. See <a href="#">Any Node</a> on page 455.
Delete		Deletes the selected node from the XSD tree, including all sub-nodes.

## Properties

None.

## All Node

The **All** node corresponds to the `<xsd:all>` element, which is used to form a group of Element nodes. The elements in this group may appear once or not at all, and may appear in any order.

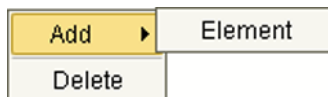
**Figure 390** All Node Icon



**Note:** Since individual elements in an *all* group can appear no more than once, they all must have `minOccurs = 0` and `maxOccurs = 1`.

## Context Menu

**Figure 391** All Node Menu



**Table 116** All Node Menu Options

Option		Action
Add	Element	Adds an <b>Element</b> sub-node to the selected All node (duplicates the action of the <b>Add Element</b> toolbar icon). See <a href="#">Element Node</a> on page 437.
Delete		Deletes the selected node from the XSD tree, including all sub-nodes.

## Properties

None.

## Any Node

The **Any** node corresponds to the `<xsd:any>` element, which defines the content of the parent `<xsd:choice>` element. The content is required to be a well-formed XML.

**Figure 392** Any Node Icon



### Context Menu

This node cannot have sub-nodes assigned to it; its context menu only offers the **Delete** option.

### Properties

The set of properties associated with *Any* is shown in Figure 385.

**Figure 393** Any Node Properties

Name	Properties
minoccurs	1
maxoccurs	1
processContent	strict
targetNamespace	strict
	lax
	skip

**Note:** Clicking the Value field for a property enables it, after which you can edit the text. For the **processContent** property, clicking the arrow button displays a menu from which you can select an attribute.

**Table 117** Any Node Property Options

Name	Description
minOccurs	Specifies the minimum number of times this element may appear; the default value is <b>1</b> .
maxOccurs	Specifies the maximum number of times this element may appear; the default value is <b>1</b> .

Name	Description
processContent	<p>Specifies how the XML processor treats the element content.</p> <ul style="list-style-type: none"><li>▪ <b>strict</b> requires that the content be guaranteed valid (default).</li><li>▪ <b>lax</b> instructs the XML processor to validate the content on a can-do basis; that is, it will validate content for which it can obtain schema information but it will not signal errors for those it cannot obtain such information.</li><li>▪ <b>skip</b> instructs the XML processor not to attempt validation of the content.</li></ul>
targetNamespace	The namespace to which the content must belong.

## ComplexContent Node

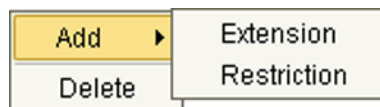
The **ComplexContent** node corresponds to the `<xsd:complexContent>` element, which is used to extend or restrict the content model of the parent complex type node.

**Figure 394** Complex Content Node Icon



### Context Menu

**Figure 395** ComplexContent Node Menu



**Table 118** ComplexContent Node Menu Options

Option		Action
Add	Extension	Adds an <b>Extension</b> sub-node to the selected ComplexContent node. See <a href="#">Extension Node</a> on page 459.
	Restriction	Adds a <b>Restriction</b> sub-node to the selected ComplexContent node. See <a href="#">Restriction Node</a> on page 460.

### Properties

The only property associated with the *ComplexContent* node is the name of the node.

## SimpleContent Node

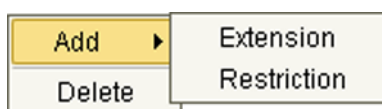
The **SimpleContent** node corresponds to the `<xsd:simpleContent>` element, which is used to indicate that the content model of the parent complex type node contains only character data (no elements).

**Figure 396** Simple Content Node Icon



### Context Menu

**Figure 397** SimpleContent Node Menu



**Table 119** SimpleContent Node Menu Options

Option		Action
Add	Extension	Adds an <b>Extension</b> sub-node to the selected SimpleContent node. See <a href="#">Extension Node</a> on page 459.
	Restriction	Adds a <b>Restriction</b> sub-node to the selected SimpleContent node. See <a href="#">Restriction Node</a> on page 460.

### Properties

The only property associated with the *SimpleContent* node is the name of the node.

## Extension Node

The **Extension** node corresponds to the `<xsd:extension>` element, which is used to declare the extension of the content model of the grandparent complex type node.

**Figure 398** Extension Node Icon



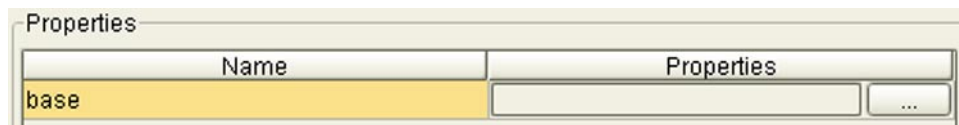
### Context Menu

This node cannot have sub-nodes assigned to it; its context menu only offers the **Delete** option.

### Properties

The set of properties associated with *Extension* is shown in Figure 399.

**Figure 399** Extension Node Properties



**Note:** Clicking the Value field for the **base** property enables it; clicking the ellipsis (...) button displays a dialog in which you can select an allowed existing type.

**Table 120** Extension Node Property Options

Name	Description
base	Specifies the existing type upon which the declared extension is based. Selecting the field and clicking the ellipsis (...) button displays a dialog containing lists of built-in simple types, internally-defined types, and imported types.

## Restriction Node

The **Restriction** node corresponds to the `<xsd:restriction>` element, which is used to declare the restriction of the content model of the parent complex type node.

**Figure 400** Restriction Node Icon



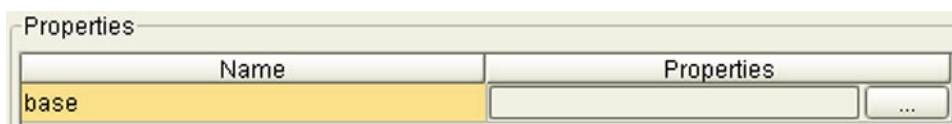
### Context Menu

This node cannot have sub-nodes assigned to it; its context menu only offers the **Delete** option.

### Properties

The set of properties associated with *Restriction* is shown in Figure 401.

**Figure 401** Restriction Node Properties



**Note:** Clicking the Value field for the **base** property enables it; clicking the ellipsis (...) button displays a dialog in which you can select an allowed existing type.

**Table 121** Restriction Node Property Options

Name	Description
base	Specifies the existing type upon which the declared restriction is based.



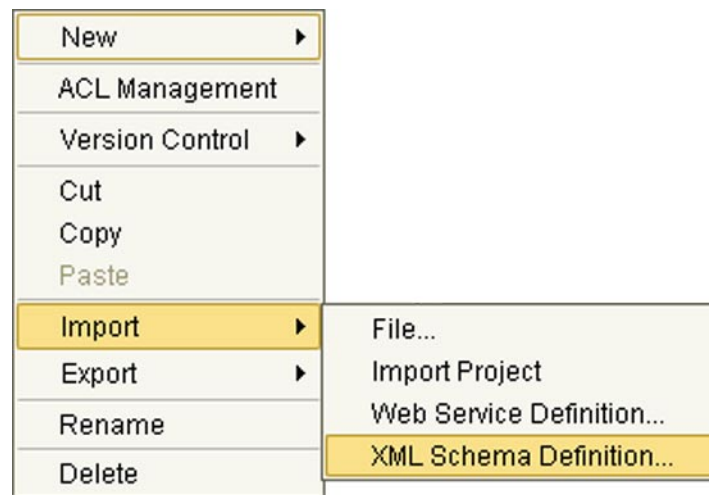
## 12.4.6 Importing Existing XML Schema Definitions

You can import (or re-import) an existing XML Schema Definition (XSD) by selecting the **Import > XML Schema Definition** option from the *Project* context menu, which adds the XSD to your Project and automatically opens the XML Schema Editor. You can use the XML Schema Editor to edit the XSD as described in the previous section.

To import an XSD file

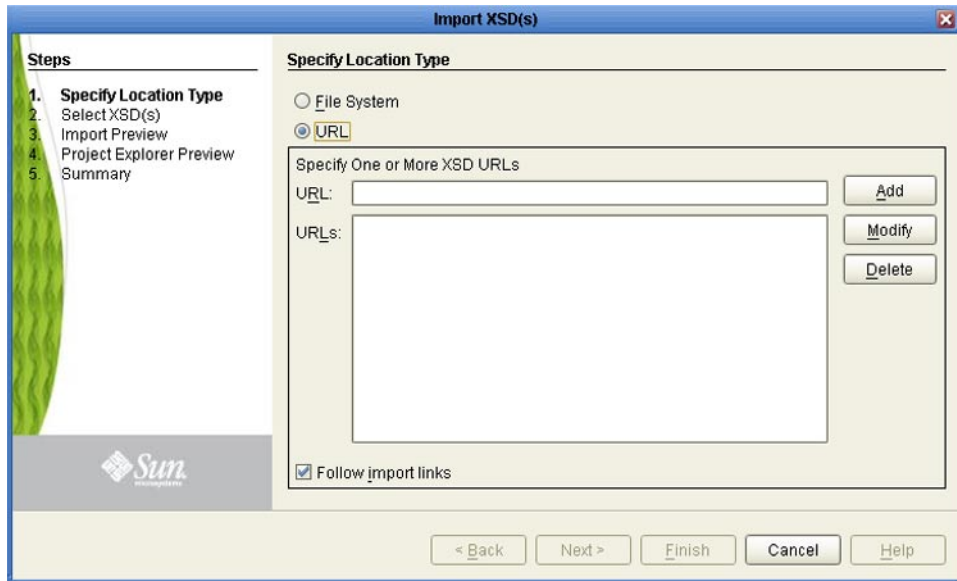
- 1 In Project Explorer, right-click the selected Project to display its context menu (see Figure 402).

**Figure 402** Project Context Menu - Import XML Schema Definition Option



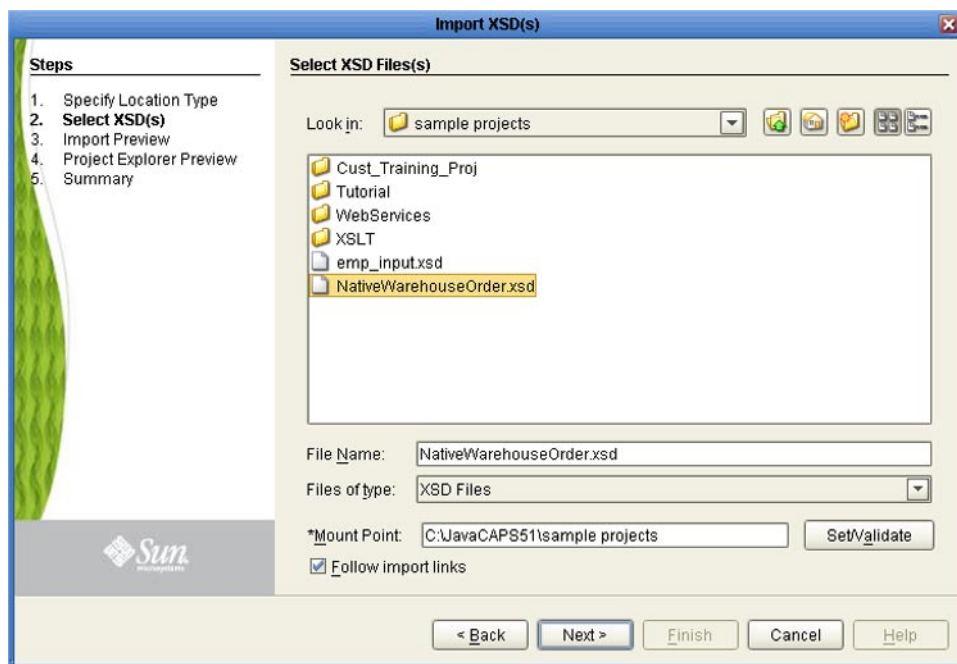
- 2 Select **Import > XML Schema Definition** to display the *XSD Import Wizard*, which begins with the dialog shown in Figure 403.

**Figure 403** Import XSD File Dialog - Step 1



- 3 In Step 1, you indicate whether the XSD file is located in your local (or networked) file system or available from an external URL. In the case of a URL, you may specify multiple URLs, if necessary.
- 4 Click **Next** to proceed to the next dialog (see Figure 404, which shows a local file system).

**Figure 404** Import XSD File Dialog - Step 2

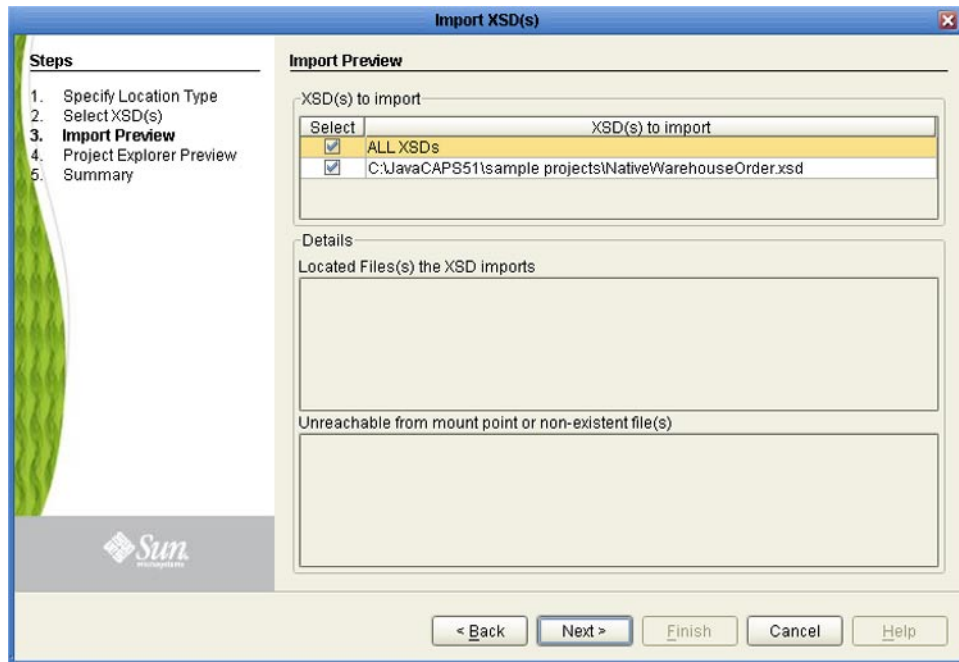


- 5 In Step 2, browse to locate the desired XSD files and select the file.

**Note:** *If the XSD has been successfully imported previously, you can clear the **Follow import links** checkbox and select only the required XSD file. This avoids unnecessarily re-importing any linked files.*

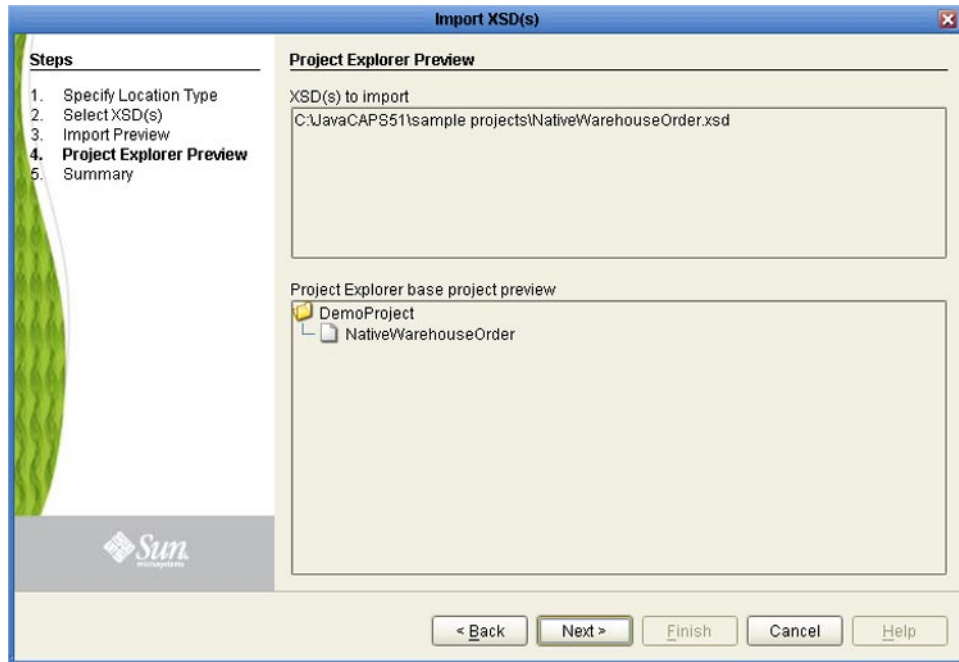
- 6 Click **Next** to proceed to the next dialog (see Figure 405).

**Figure 405** Import XSD File Dialog - Step 3



- 7 In Step 3 you can specify whether to import the specific XSD file only, or include all referenced XSD files. The *Details* panel shows whether the referenced files were or were not accessible.
- 8 Click **Next** to proceed to the next dialog (see Figure 406).

**Figure 406** Import XSD File Dialog - Step 4



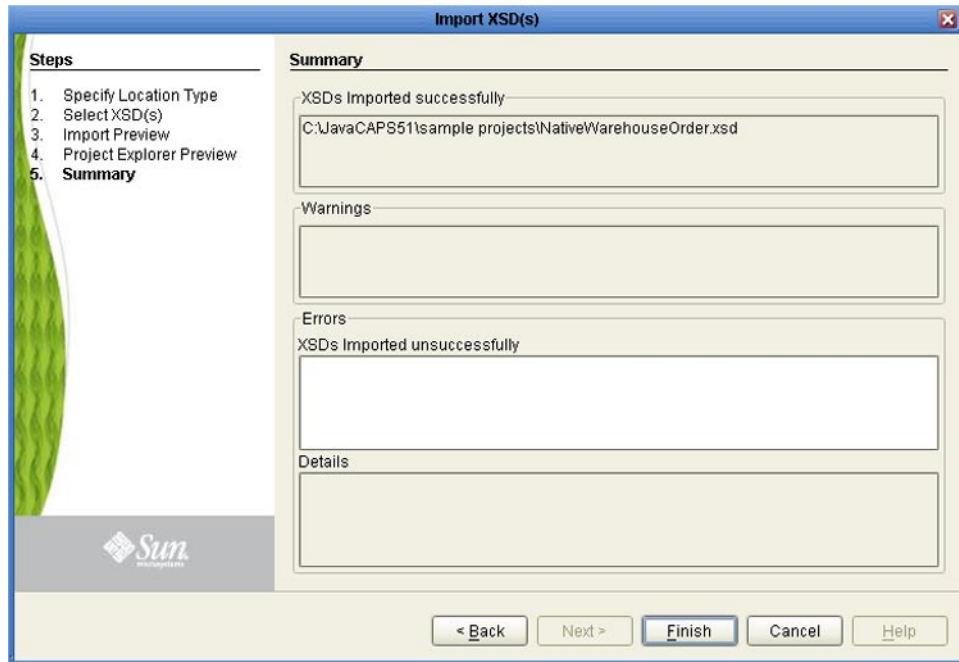
- 9 In Step 4, you are provided with a preview of the files to import and their destinations in the Project tree.
- 10 If the preview is correct, click **Next** to display the confirmation dialog shown in Figure 407.

**Figure 407** Import XSD File Dialog - Step 4 Confirmation



- 11 Assuming everything is correct, click **Yes** to initiate the import process.
- 12 Once the import process has completed, the *Summary* dialog will appear (see Figure 408).

**Figure 408** Import XSD File Dialog - Step 5



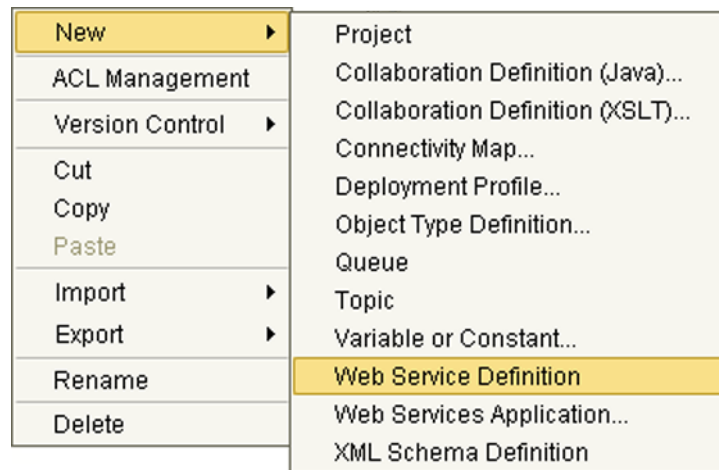
- 13 Assuming your files were imported successfully, click **Finish**.

## 12.5 Creating Web Service Definitions

### 12.5.1 Overview

Selecting the **New > Web Service Definition** option from the *Project* context menu (see Figure 409) adds a WSDL Object to your Project and automatically opens the Web Service Definition Editor (see [Using the Web Service Definition Editor](#) on page 467).

**Figure 409** Project Context Menu - New Web Service Definition



### 12.5.2 Using the Context Menu

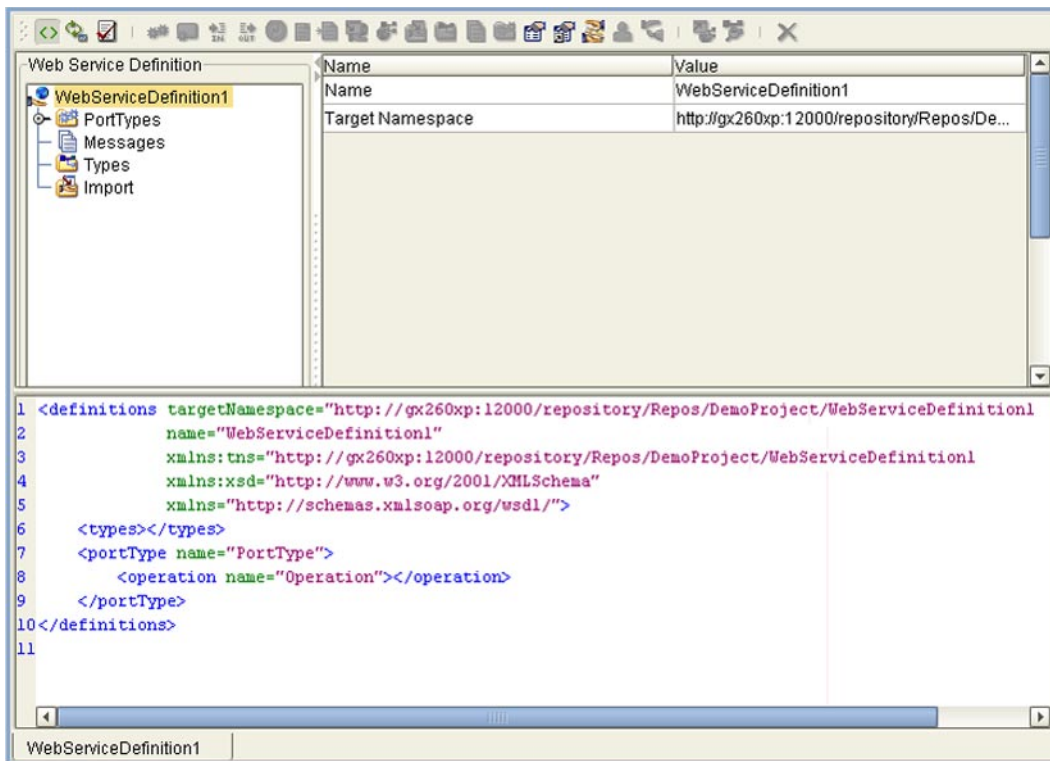
Right-clicking the WSDL Object in Project Explorer displays its context menu (see [Using Project Component Context Menus](#) on page 109).

## 12.6 Using the Web Service Definition Editor

### 12.6.1 Overview

The Web Service Definition Editor (see Figure 410) is an editor within Enterprise Designer, with which you can develop new, or edit existing, WSDL Objects with a minimum of coding.

**Figure 410** Web Service Definition Editor User Interface



The various icons and context menu options are enabled only when the resulting action is allowed or appropriate. If the resulting action is illegal or inappropriate, the icon or option is disabled.

**Note:** Although the user interface for Web Service Definition Editor contains features that help prevent the creation of invalid code, you need to be knowledgeable of WSDL and XSD specifications.

## Toolbar

The first three icons in the toolbar allow you to display or hide the WSDL source code, build the WSDL code based on the graphical constructs, and validate the WSDL code. The remaining icons allow you to build the WSDL Object graphically. See [Web Service Definition Editor Toolbar Icons](#) on page 469 for detailed descriptions.

## Web Service Definition Panel

The *Web Service Definition* panel displays the web service definition as a hierarchical tree structure, as shown in Figure 411. Selecting a node in the tree displays the properties for that node in the *Properties* panel. These features are detailed in [Building a Web Service Definition](#) on page 472.

## Properties Panel

The *Properties* panel displays the various properties defined for the selected node. Most *Value* fields are automatically populated with default values, and most fields are editable. These features are detailed individually in [Building a Web Service Definition](#) on page 472.

## Source Code Panel










The source code panel displays the WSDL code for the web services definition, and provides a text editor to supplement the Web Service Definition Editor's graphical capabilities. When editing the WSDL code in the source code panel, you need to synchronize the definition in the Repository with your edits by clicking the *Synchronize* icon in the toolbar (see [Web Service Definition Editor Toolbar Icons](#) on page 469).

The source code panel is hidden by default; you must enable it by clicking the *Show Web Service Definition Code* icon shown in [Web Service Definition Editor Toolbar Icons](#) on page 469.











## 12.6.2 Web Service Definition Editor Toolbar Icons







Table 122 Web Service Definition Editor Toolbar Icons

Icon	Command	Function
	Export WSDL to File	Presents a dialog in which you can save the serialized Web Service Definition to your local file system as a <b>.wsdl</b> file.
	Show WSDL Code	Displays (or hides) the WSDL source code in a panel located in the lower part of the Web Service Definition Editor.
	Synchronize	Synchronizes the contents of the inactive mode with the active mode. For example, if you are editing in the text mode, activating this function synchronizes the graphical mode with the text mode.
	Validate Web Service Definition	Validates the WSDL code and displays a report displaying descriptions of any errors detected and suggested means of correcting them.
	Port Type	Adds a new PortType under the <i>PortTypes</i> folder (duplicates the action of the <b>Add Port Type</b> option on the <i>PortTypes</i> folder context menu). See <a href="#">PortTypes</a> on page 480.
	Operation	Adds a new Operation to the selected PortType (duplicates the action of the <b>Add Operation</b> option on the PortType context menu). See <a href="#">Operations</a> on page 481.
	Input Message	Adds a new input message to the selected Operation (duplicates the action of the <b>Add Input Message</b> option on the Operation context menu). See <a href="#">Operations</a> on page 481.
	Output Message	Adds a new output message to the selected Operation (duplicates the action of the <b>Add Output Message</b> option on the Operation context menu). See <a href="#">Operations</a> on page 481.
	Exception	Adds an exception (or fault message) to an output message (duplicates the action of the <b>Add Fault</b> option on the Operation context menu). Option is disabled unless an output message exists for the Operation. See <a href="#">Operations</a> on page 481.
	Message Part	Adds a new message under the <i>Messages</i> folder (duplicates the action of the <b>Add Message</b> option on the <i>Messages</i> folder context menu). See <a href="#">Messages</a> on page 476.
	Message Value	Adds a new value to the selected message (duplicates the action of the <b>Add Part</b> option on the Message context menu). See <a href="#">Message Parts</a> on page 478.

**Table 122** Web Service Definition Editor Toolbar Icons

Icon	Command	Function
	Import Web Service	Displays a dialog for importing a WSDL or XSD file to the <i>Import</i> folder (duplicates the action of the <b>Import WSDL, XSD or OTD</b> option on the <i>Import</i> folder context menu). <b>Important:</b> If you are building a Web Service Definition that will be exposed externally as a web service through a SOAP/HTTP interaction, you must base the WSD on an XSD object created in the XML Schema Editor. A WSD having imported OTDs other than XSD Objects will not support SOAP/HTTP.
	Add Schema Type	Adds a new Schema Type under the <i>Types</i> folder. <b>Note:</b> This functionality is not used in Java CAPS 5.1.
	Add Import Folder	Adds a new <i>Import</i> folder under the Web Service Definition root node (duplicates the action of the <b>Add Import Folder</b> option on the Web Service Definition root node context menu). Icon is disabled if an <i>Import</i> folder already exists for the WSDL Object. See <a href="#">Import Folder</a> on page 474.
	Add Schema Folder	Adds a new <i>Types</i> folder under the Web Service Definition root node (duplicates the action of the <b>Add Schema Types</b> option on the Web Service Definition root node context menu). Icon is disabled if a <i>Types</i> folder already exists for the WSDL Object. See <a href="#">Types Folder</a> on page 490.
	Add Message Folder	Adds a new <i>Messages</i> folder under the Web Service Definition root node (duplicates the action of the <b>Add Message Folder</b> option on the Web Service Definition root node context menu). Icon is disabled if a <i>Messages</i> folder already exists for the WSDL Object. See <a href="#">Messages Folder</a> on page 476.
	Add PortTypes Folder	Adds a new <i>PortTypes</i> folder under the Web Service Definition root node (duplicates the action of the <b>Add PortType Folder</b> option on the Web Service Definition root node context menu). Icon is disabled if a <i>PortTypes</i> folder already exists for the WSDL Object. See <a href="#">PortTypes Folder</a> on page 480.
	Add Property	Adds a new <i>Properties</i> folder under the Web Service Definition root node and a new property in the folder (duplicates the action of the <b>Add Property</b> option on the Web Service Definition root node context menu). Icon is disabled if a <i>Properties</i> folder already exists for the WSDL Object. See <a href="#">Property</a> on page 484. <b>Note:</b> This functionality is not used in Java CAPS 5.1.
	Add Property Alias	Adds a new <i>Property Aliases</i> folder under the root node (duplicates the action of the <b>Add Property Alias</b> option on the Web Service Definition root node context menu). Icon is disabled if a <i>Property Aliases</i> folder already exists for the WSDL Object. See <a href="#">Property Aliases Folder</a> on page 486. <b>Note:</b> This functionality is not used in Java CAPS 5.1.

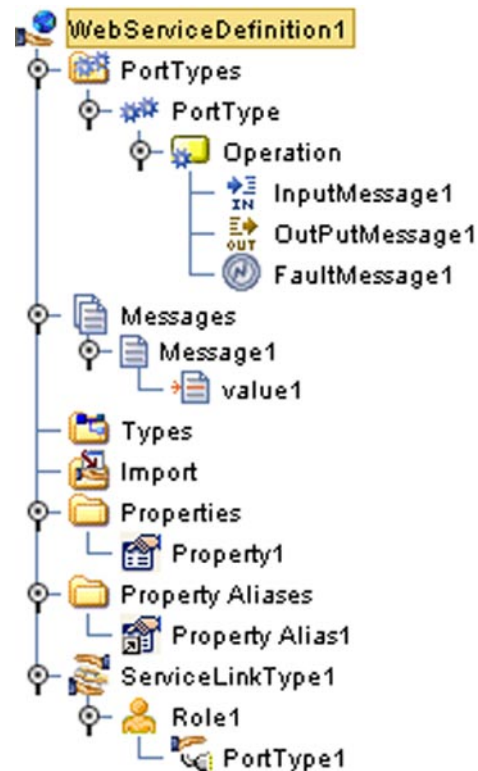
**Table 122** Web Service Definition Editor Toolbar Icons

Icon	Command	Function
	Add Service Link Type	Adds a new <i>Service Link Type</i> folder under the root node (duplicates the action of the <b>Add Service Link Type</b> option on the Web Service Definition root node context menu). Icon is disabled if a <i>Service Link Type</i> folder already exists for the WSDL Object. See <a href="#">Service Link Type Folder</a> on page 488. <b>Note:</b> This functionality is not used in Java CAPS 5.1.
	Add Role	Adds a new Role under the Service Link Type folder (duplicates the action of the <b>Add Service Link Role</b> option on the <i>Service Link Type</i> folder context menu). See <a href="#">Service Link Role</a> on page 488. <b>Note:</b> This functionality is not used in Java CAPS 5.1.
	Add PortType	Adds a new PortType to the Service Link Role (duplicates the action of the <b>Add Service Link PortType</b> option on the <i>Role</i> context menu). See <a href="#">PortTypes</a> on page 498.
	Preview XSD Node	Presents a dialog containing allowed options; clicking <b>OK</b> in the dialog opens the OTD Editor, displaying the selected top-level Element, ComplexType, or SimpleType and its substructure. (To return to XSD Schema Editor, use Window menu.) See <a href="#">Previewing and Exporting XSD Nodes</a> on page 424.
	Export XSD Node	Presents a dialog containing allowed options; clicking <b>OK</b> in the dialog exports the selected top-level Element, ComplexType, or SimpleType as an <i>XSD Node</i> , which can be used in a Collaboration Definition (Java). See <a href="#">Previewing and Exporting XSD Nodes</a> on page 424.
	Delete	Deletes the selected node from the XSD tree, including all sub-nodes.

### 12.6.3 Building a Web Service Definition

A WSDL Object is represented in the Web Service Definition Editor by a tree structure with a hierarchy of nodes, consistent with the representations in other Enterprise Designer editors. You can build WSDL structures graphically by adding nodes to the tree, beginning with the root node, using the **Add** option in the individual node context menus or the toolbar icons. Figure 411 shows an example WSDL tree structure demonstrating all possible node combinations (the *Types* and *Import* folders are unpopulated, however).

**Figure 411** WSDL Tree - All Options



You can also build the WSDL Object by entering text in the source code panel, which you must enable by clicking the *Show Web Service Definition Code* icon shown in [Web Service Definition Editor Toolbar Icons](#) on page 469. When editing the WSDL code in the source code panel, you need to synchronize the definition in the Repository with your edits by clicking the *Synchronize* icon in the toolbar (again, see [Web Service Definition Editor Toolbar Icons](#) on page 469).

## Web Service Definition Root Node

The root node represents the Web Service Definition itself, with all components represented as sub-nodes in a tree structure. A new Web Service Definition is automatically provided with a basic skeleton, containing PortTypes, Messages, Types, and Import folders. You can add other folders as appropriate by using the context menu shown in Figure 412 or the corresponding icons in the toolbar.

### Context Menu

**Figure 412** Web Service Definition Root Node Menu



**Table 123** Web Service Definition Root Node Menu Options

Command	Function
Add Service Link Type	Adds a new <i>Service Link Type</i> folder under the Web Service Definition root node (duplicates the action of the <b>Add Service Link Type</b> icon). The folder is automatically populated with a default Role and Port Type. Option is not shown if a <i>Service Link Type</i> folder already exists for the WSDL Object. <b>Note:</b> This functionality is not used in Java CAPS 5.1.
Add Property Alias Folder	Adds a new Property Alias Folder under the Web Service Definition root node (duplicates the action of the <b>Add Property Alias</b> icon). The folder is automatically populated with a default Property Alias. Option is not shown if a <i>Property Alias</i> folder already exists for the WSDL Object. <b>Note:</b> This functionality is not used in Java CAPS 5.1.
Add Property	Adds a new <i>Properties</i> folder under the Web Service Definition root node and a new property in the folder (duplicates the action of the <b>Add Property</b> icon). The folder is automatically populated with a default Property. Option is not shown if a <i>Properties</i> folder already exists for the WSDL Object. <b>Note:</b> This functionality is not used in Java CAPS 5.1.
Collapse/Expand	Toggles to hide or display the sub-node structure, when sub-nodes are present (option not shown when no sub-nodes are present).

## Import Folder

You can import elements that exist in other WSDL or XSD documents by importing those documents into the *Import* folder. This can be accomplished by using the **Import WSDL, XSD or OTD** option in the Import folder context menu (shown in Figure 413) or by using the *Import Web Service* icon in the toolbar.

### Context Menu

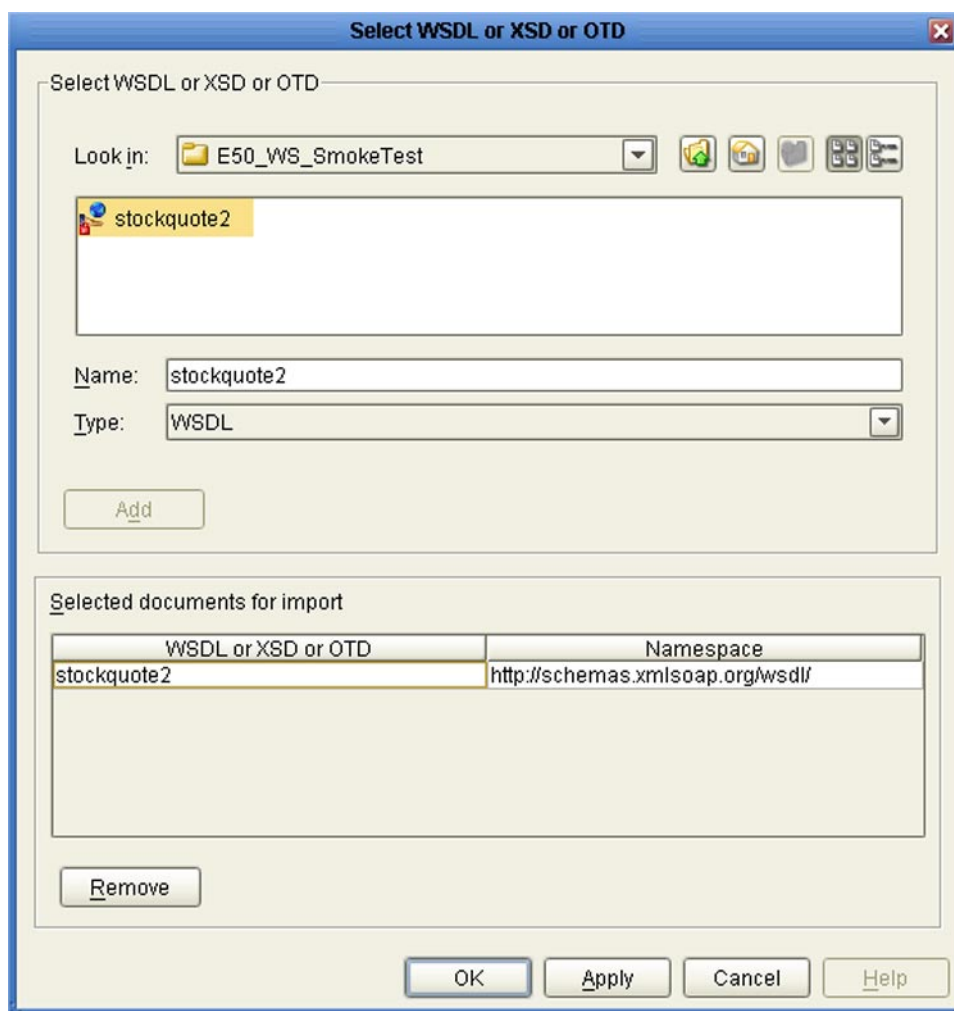
**Figure 413** Import Folder Menu



**Table 124** Import Folder Menu Options

Command	Function
Import WSDL, XSD or OTD	Displays a dialog (see Figure 414) for importing a WSDL or XSD file, or an OTD, to the Import folder (duplicates the action of the <b>Import Web Service</b> icon).
Collapse/Expand	Toggles to hide or display the sub-node structure, when sub-nodes are present (option not shown when no sub-nodes are present).
Delete	Removes the node and all sub-nodes.

**Figure 414** Select WSDL, XSD or OTD Dialog

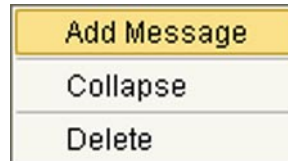


## Messages Folder

Messages represent abstract definitions of the data. They may be entire documents, or arguments that can be mapped to method invocations. You can add messages to the *Messages* folder by using its context menu (shown in Figure 415) or by using the *Message Part* icon in the toolbar.

### Context Menu

**Figure 415** Messages Folder Menu



**Table 125** Messages Folder Menu Options

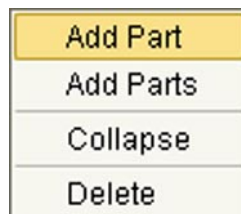
Command	Function
Add Message	Adds a new message under the Messages folder (duplicates the action of the <b>Message Part</b> icon).
Collapse/Expand	Toggles to hide or display the sub-node structure, when sub-nodes are present (option not shown when no sub-nodes are present).
Delete	Removes the node and all sub-nodes.

## Messages

Once you have added a message to the *Messages* folder, you can add parts or values to it by means of its context menu (shown in Figure 416) or by using the *Message Value* icon in the toolbar.

### Context Menu

**Figure 416** Message Menu



**Table 126** Message Menu Options

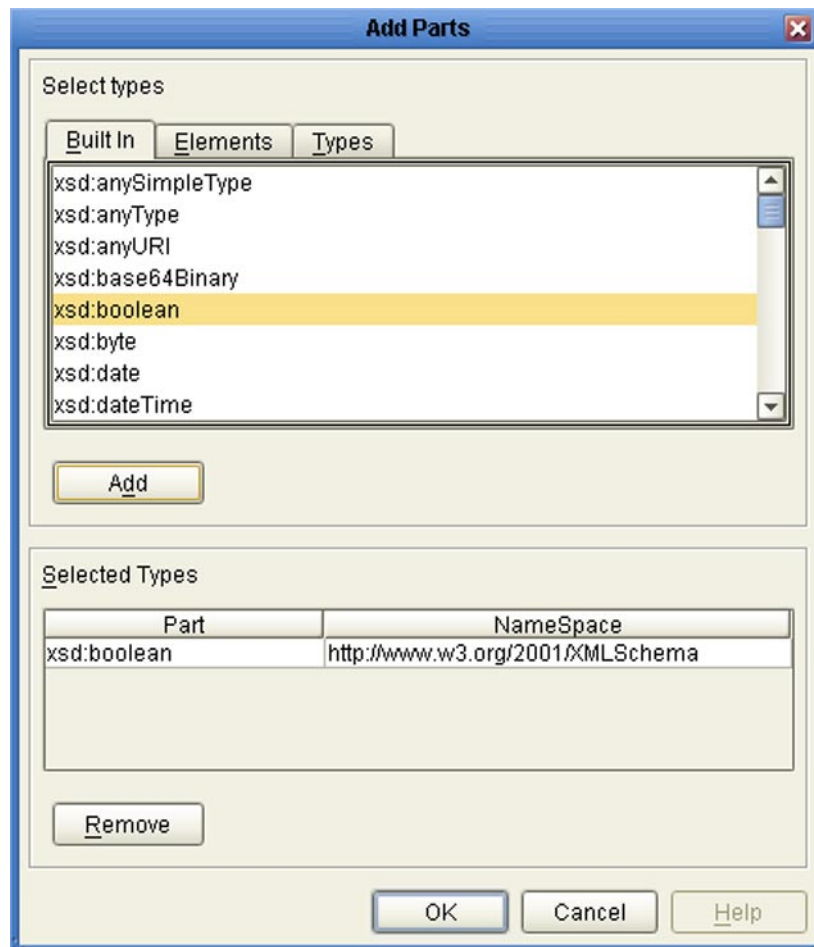
Command	Function
Add Part	Adds a new value to the selected message (duplicates the action of the <b>Message Value</b> icon).



**Table 126** Message Menu Options

Command	Function
Add Parts	Displays a dialog in which you can select multiple values for the message from a scrollable list (see Figure 417).
Collapse/Expand	Toggles to hide or display the sub-node structure, when sub-nodes are present (option not shown when no sub-nodes are present).
Delete	Removes the node and all sub-nodes.

**Figure 417** Add Parts Dialog



The *Select Parts* dialog allows you to assemble a list of values based on standard simple types, or other types existing in internal or imported documents. Clicking **Add** adds the highlighted type to the list; clicking **Select** adds the list to the message.

## Message Parts

A message part, or value, refers to an individual constituent part of the message that can be identified with a defined data type.

### Context Menu

This node cannot have sub-nodes assigned to it; its context menu only offers the **Delete** option.

### Properties

Properties for message values (or parts) are shown in Figure 418. Click in the **Part Type** value field to activate it for editing, then select a message type from the drop-down menu, if an appropriate type has been defined.

**Figure 418** Message Part Properties (Initial)

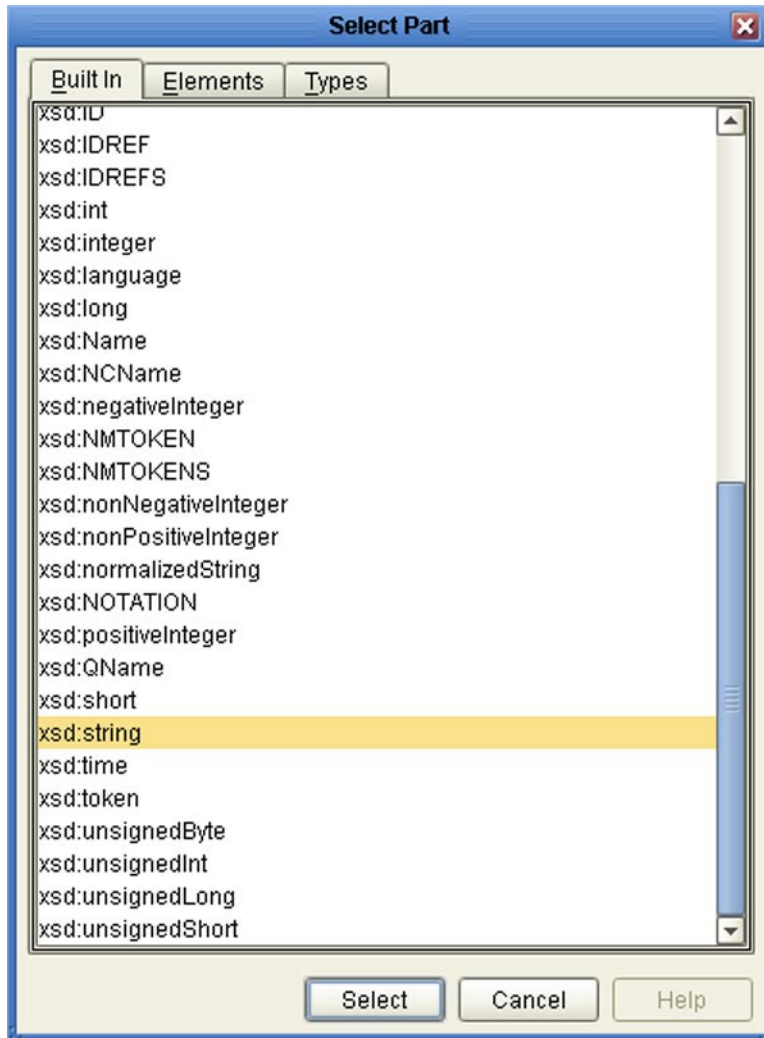
Name	Value
Name	value1
Part Type	<input type="text"/> <input type="button" value="..."/>
Part Type Namespace	

**Note:** Clicking the Value field for a property enables it, after which you can edit the text. For exceptions to this rule, see the following table.

**Table 127** Message Value Node Property Options

Name	Description
name	Name of the value.
Part Type	Selecting the field and clicking the ellipsis (...) button displays a dialog containing a list of existing types (see Figure 419). You can select a type based on standard simple types, or other types existing in internal or imported documents.
Part Type Namespace	The namespace within which this part type belongs (see Figure 420 for an example). The value for this property is entered automatically when you select the Part Type.

**Figure 419** Select Part Dialog



The *Select Part* dialog allows you to select a value based on a standard simple type, or another type existing in internal or imported documents. Clicking **Select** adds the list to the message. For example, selecting **String** populates the properties fields as shown in Figure 420.

**Figure 420** Message Value Properties (String)

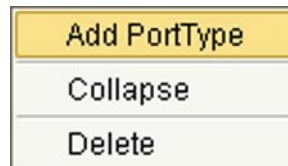
Name	Value
Name	value1
Part Type	xsd:string
Part Type Namespace	http://www.w3.org/2001/XMLSchema

## PortTypes Folder

PortTypes represent logical groupings, or sets, of operations. You can add PortTypes to the *PortTypes* folder by using the folder's context menu (shown in Figure 421) or by using the *PortType* icon in the toolbar.

### Context Menu

**Figure 421** PortTypes Folder Menu



**Table 128** PortTypes Folder Menu Options

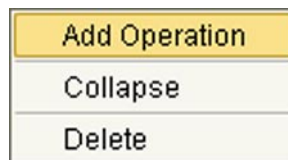
Command	Function
Add PortType	Adds a new PortType under the PortType folder (duplicates the action of the <b>PortType</b> icon).
Collapse/Expand	Toggles to hide or display the sub-node structure, when sub-nodes are present (option not shown when no sub-nodes are present).
Delete	Removes the node and all sub-nodes.

## PortTypes

Once you have added a PortType, you can add operations to it by using its context menu (shown in Figure 422) or by using the *Operation* icon in the toolbar.

### Context Menu

**Figure 422** PortType Menu



**Table 129** PortType Menu Options

Command	Function
Add Operation	Adds a new Operation to the selected PortType (duplicates the action of the <b>Operation</b> icon).
Collapse/Expand	Toggles to hide or display the sub-node structure, when sub-nodes are present (option not shown when no sub-nodes are present).

**Table 129** PortType Menu Options

Command	Function
Delete	Removes the node and all sub-nodes.

## Operations

Operations refer to the type of operation that the web service is to perform on behalf of a message or set of messages. Adding only an input message defines the operation to be *one-way*. Adding only an output message defines a *solicit response* operation. Adding both an input and an output message defines a *request/response* operation. Adding an output message also enables the addition of an optional fault message.

**Note:** *If you are building a Web Service Definition that will be exposed externally as a web service through a SOAP/HTTP interaction, you must base all Operations containing input/output messages on XSD Objects created in the XML Schema Editor. Operations based on OTDs will not support SOAP/HTTP.*

## Context Menu

**Figure 423** Operation Menu**Table 130** Operation Menu Options

Command	Function
Add Input	Adds an input message to the selected Operation (duplicates the action of the <b>Input Message</b> icon). Each Operation can have only one input message.
Add Output	Adds an output message to the selected Operation (duplicates the action of the <b>Output Message</b> icon). Each Operation can have only one output message.
Add Fault	Adds a fault message (or exception) to the output message (duplicates the action of the <b>Exception</b> icon). Multiple fault messages are allowed. Option is enabled after an output message has been added to the Operation.
Collapse/Expand	Toggles to hide or display the sub-node structure, when sub-nodes are present (option not shown when no sub-nodes are present).
Delete	Removes the node and all sub-nodes.

## Properties

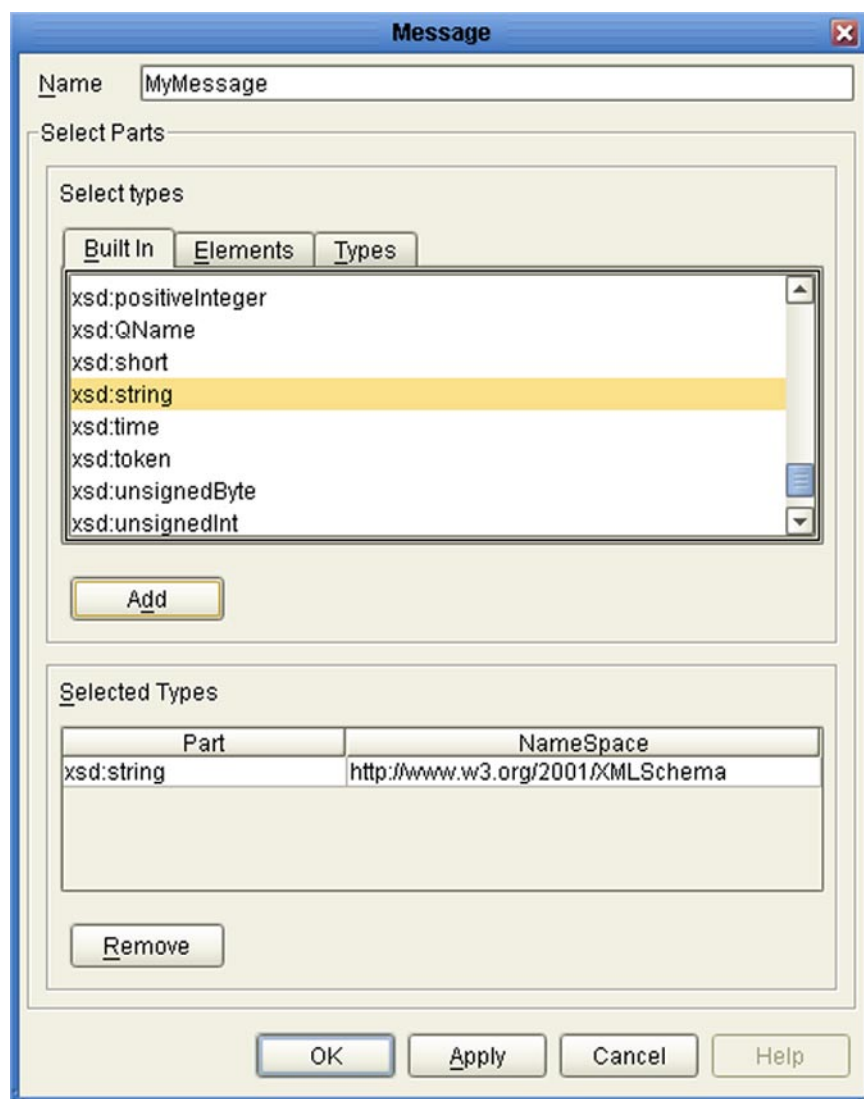
Properties for Input, Output, and Fault messages are shown in Figure 424. Click in the **Message Type** value field to activate it for editing, then select a message type from the drop-down menu, if an appropriate type has been defined.

**Figure 424** Message Properties

Name	Value
Name	InputMessage1
Message Type	tns:Message1
	tns:Message1
	tns:Deadline

To add a message type, click the ellipsis (...) button, which opens the *Message* dialog, shown in Figure 425. You can select a type based on standard simple types, or other types existing in internal or imported documents. Naming the message type (for example, *Deadline*) and clicking **OK** adds the message type to the list shown in Figure 424.

**Figure 425** Message Dialog

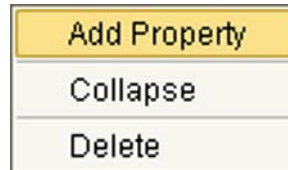


## Properties Folder

The Properties folder serves as a container for individual Properties, which can be virtually any type of classification or identification information.

### Context Menu

**Figure 426** Properties Folder Menu



**Table 131** Properties Folder Menu Options

Command	Function
Add Property	Adds a new property under the Properties folder (duplicates the action of the <b>Add Property</b> icon).
Collapse/Expand	Toggles to hide or display the sub-node structure, when sub-nodes are present (option not shown when no sub-nodes are present).
Delete	Removes the node and all sub-nodes.

## Property

Properties are usually expressed as text or XML. References to resources are entered as property aliases (see [Property Alias](#) on page 487).

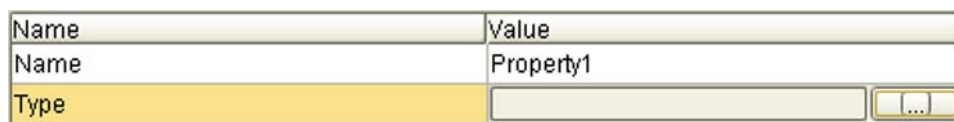
### Context Menu

This node cannot have sub-nodes assigned to it; its context menu only offers the **Delete** option.

### Properties

Properties for the Property element are shown in Figure 427.

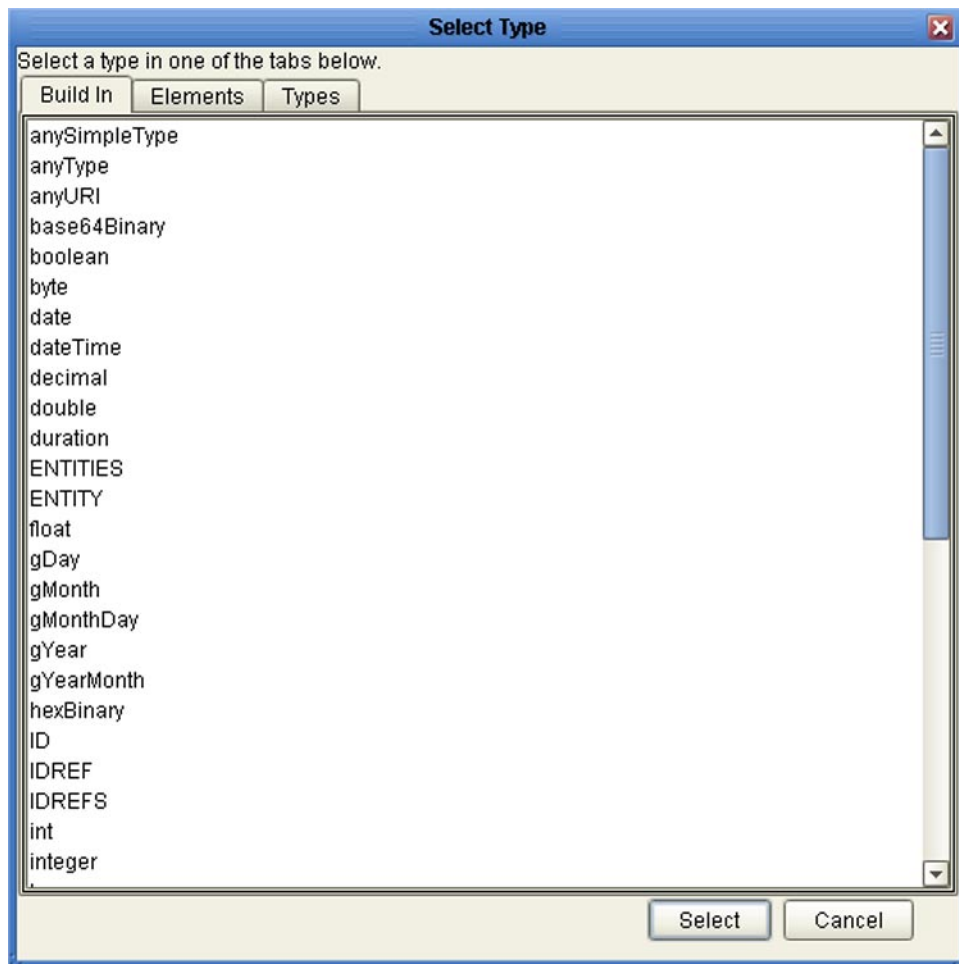
**Figure 427** Property Properties



Click in the **Type** value field to activate it for editing, then , click the ellipsis (...) button, which opens the *Select Type* dialog, shown in Figure 428.



**Figure 428** Select Type Dialog

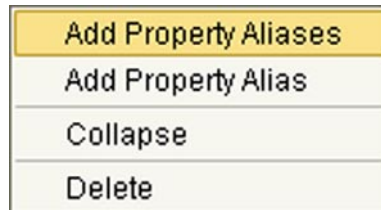


## Property Aliases Folder

**Note:** This functionality is not used in Java CAPS 5.1.

### Context Menu

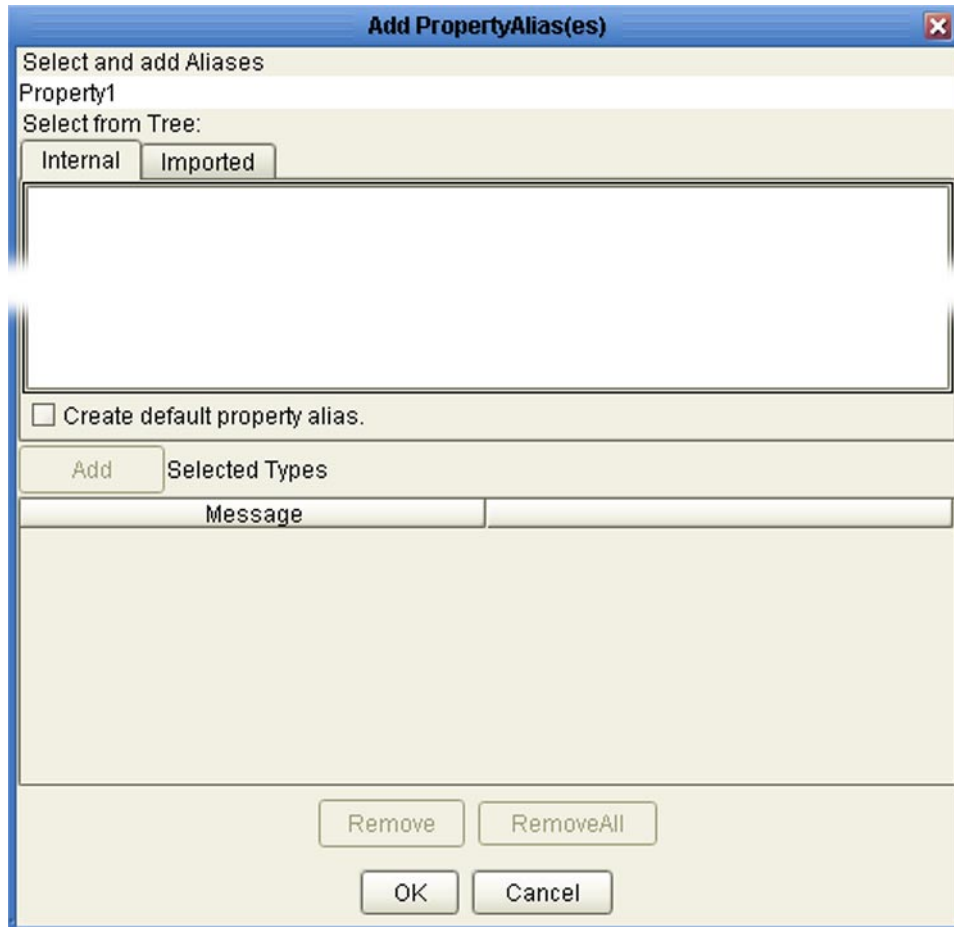
**Figure 429** Property Aliases Folder Menu



**Table 132** Properties Folder Menu Options

Command	Function
Add Property Aliases	Presents a dialog (see Figure 342) with which you can define multiple property aliases.
Add Property Alias	Adds a new property alias under the Property Aliases folder (duplicates the action of the <b>Add Property Alias</b> icon).
Collapse/Expand	Toggles to hide or display the sub-node structure, when sub-nodes are present (option not shown when no sub-nodes are present).
Delete	Removes the node and all sub-nodes.

**Figure 430** Add Property Aliases Dialog



The *Add Property Aliases* dialog allows you to assemble a list of property aliases based on existing internal or imported documents.

Clicking **Add** adds the highlighted type to the list; clicking **OK** adds the list to the property. You can delete items from the list using the **Remove** or **Remove All** buttons.

## Property Alias

***Note:** This functionality is not used in Java CAPS 5.1.*

### Context Menu

This node cannot have sub-nodes assigned to it; its context menu only offers the **Delete** option.

## Service Link Type Folder

**Note:** This functionality is not used in Java CAPS 5.1.

### Context Menu

**Figure 431** Service Link Type Folder Menu



**Table 133** Service Link Type Folder Menu Options

Command	Function
Add Service Link Role	Adds a new role under the Service Link Type folder (duplicates the action of the <b>Add Role</b> icon).
Collapse/Expand	Toggles to hide or display the sub-node structure, when sub-nodes are present (option not shown when no sub-nodes are present).
Delete	Removes the node and all sub-nodes.

## Service Link Role

**Note:** This functionality is not used in Java CAPS 5.1.

### Context Menu

**Figure 432** Role Menu



**Table 134** Role Menu Options

Command	Function
Add Service Link PortType	Adds a new PortType for the selected Service Link Role (duplicates the action of the <b>Add PortType</b> icon). <b>Note:</b> Only one PortType can be added as a sub-node. After the PortType is added, this option is removed from the menu.
Collapse	Hides the sub-node structure. <b>Note:</b> After a PortType is added, this option is added to the menu.

**Table 134** Role Menu Options

Command	Function
Delete	Removes the node and all sub-nodes.

## Types Folder

The Types folder contains the metadata for the data messages in the form of XML Schemas, which are available in the in-line XML Schema. The node can be controlled by means of its context menu, shown in Figure 433.

### Context Menu

**Figure 433** Schema Type Menu



**Table 135** Schema Type Menu Options

Command	Function
Collapse	Hides the sub-node structure.
Expand	Displays the sub-node structure.
Delete	Removes the node and all sub-nodes.

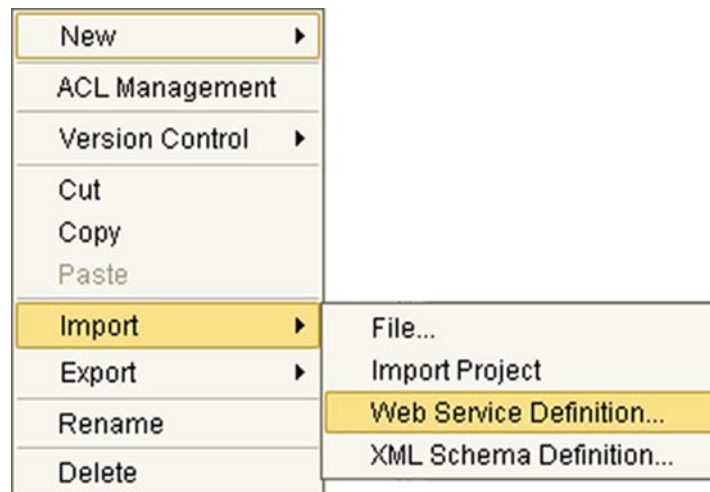
## 12.6.4 Importing Existing Web Service Definitions

You can import an existing WSDL Object by selecting the **Import > Web Service Definition** option from the *Project* context menu, which adds the WSDL Object to your Project and automatically opens the Web Service Definition Editor. You can use the Web Service Definition Editor to edit the WSDL Object as described in the previous section. See the *Sun SeeBeyond eInsight™ Business Process Manager User's Guide* for additional information.

### To import a WSDL Object

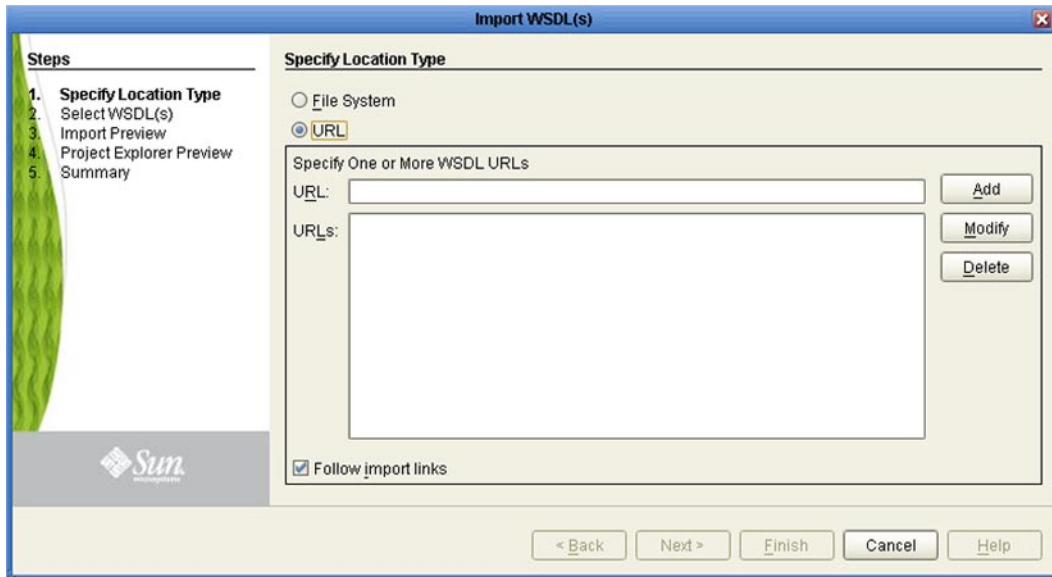
- 1 In Project Explorer, right-click the selected Project to display its context menu (see Figure 434).

**Figure 434** Project Context Menu - Import Web Service Definition Option



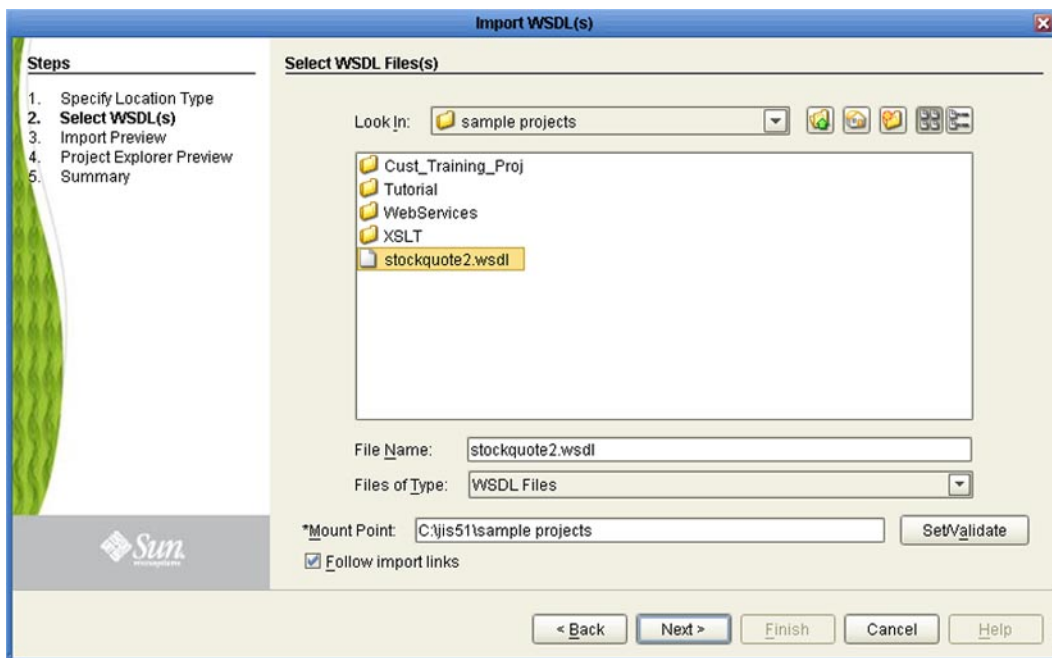
- 2 Select **Import > Web Service Definition** to display the *WSDL Import Wizard*, which begins with the dialog shown in Figure 435.

**Figure 435** WSDL Import Wizard - Step 1



- 3 In Step 1, you indicate whether the WSDL file is located in your local (or networked) file system or available from an external URL. In the case of a URL, you may specify multiple URLs, if necessary.
- 4 Click **Next** to proceed to the next dialog (see Figure 436, which shows a local file system).

**Figure 436** WSDL Import Wizard - Step 2

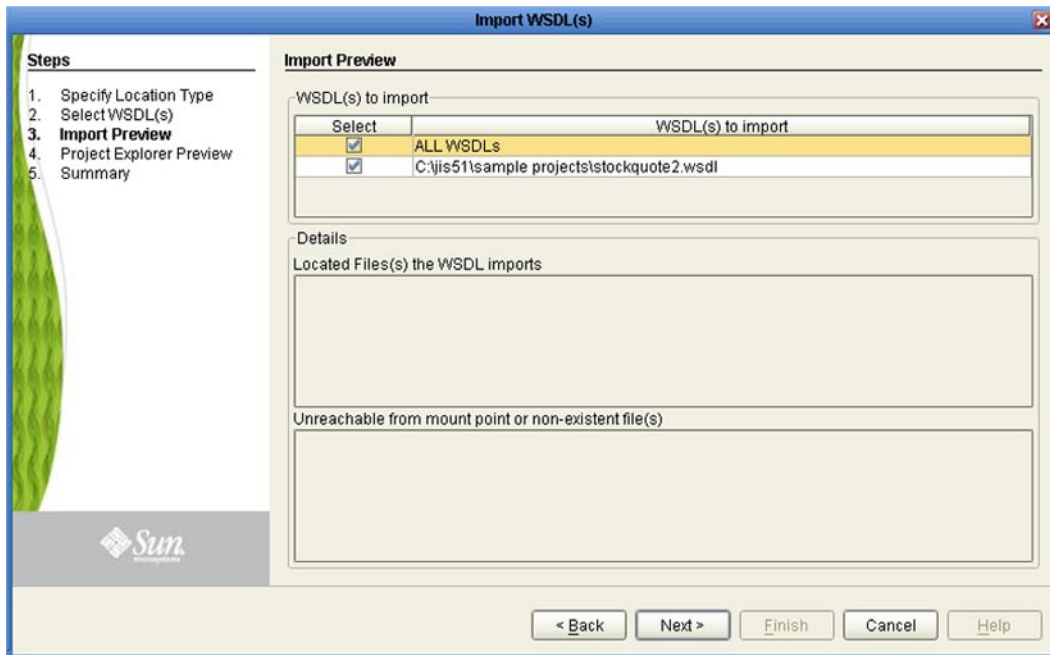


- 5 In Step 2, browse to locate the desired WSDL files and select the file.



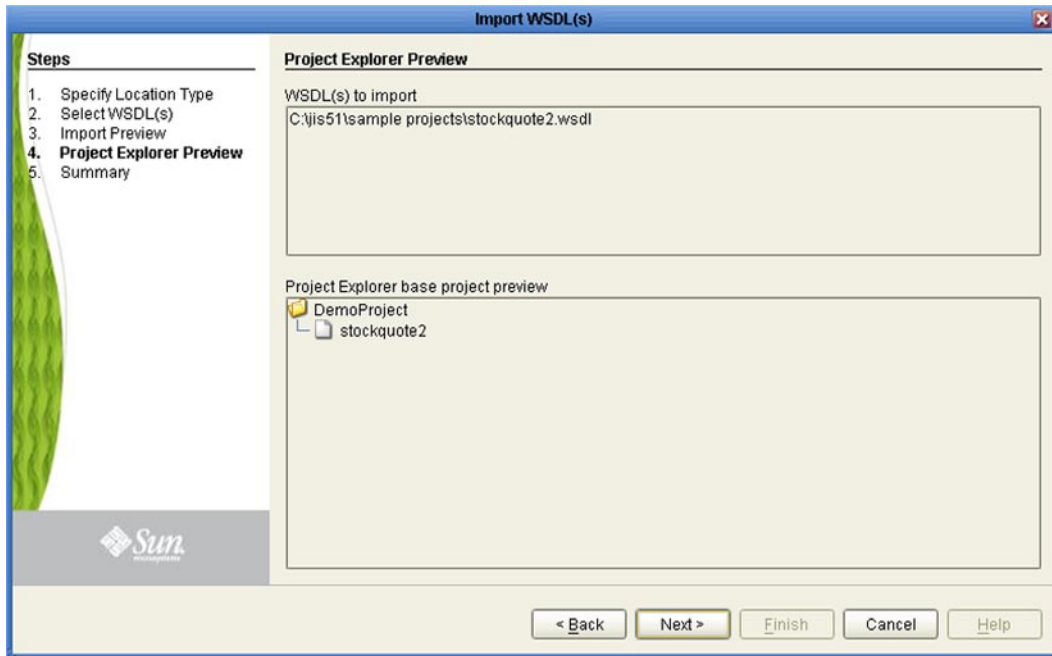
- 6 Click **Next** to proceed to the next dialog (see Figure 437).

**Figure 437** WSDL Import Wizard - Step 3



- 7 In Step 3 you can specify whether to import the specific WSDL file only, or include all referenced WSDL files. The *Details* panel shows whether the referenced files were or were not accessible.
- 8 Click **Next** to proceed to the next dialog (see Figure 438).

**Figure 438** WSDL Import Wizard - Step 4



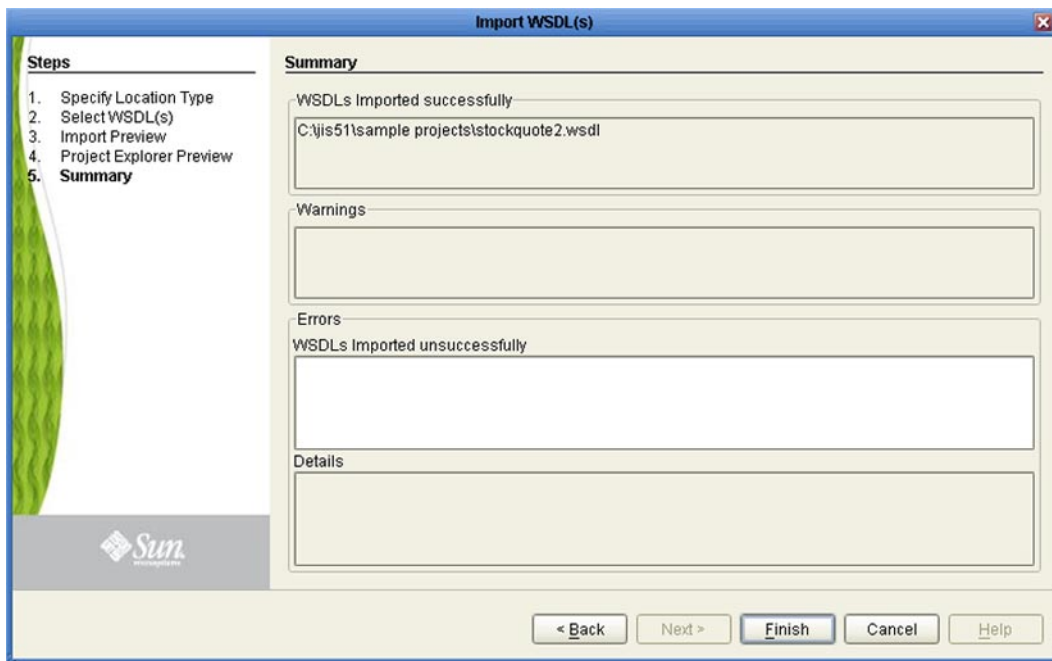
- 9 In Step 4, you are provided with a preview of the files to import and their destinations in the Project tree.
- 10 If the preview is correct, click **Next** to display the confirmation dialog shown in Figure 439.

**Figure 439** WSDL Import Wizard - Step 4 Confirmation



- 11 Assuming everything is correct, click **Yes** to initiate the import process.
- 12 Once the import process has completed, the *Summary* dialog will appear (see Figure 440).

**Figure 440** WSDL Import Wizard - Step 5



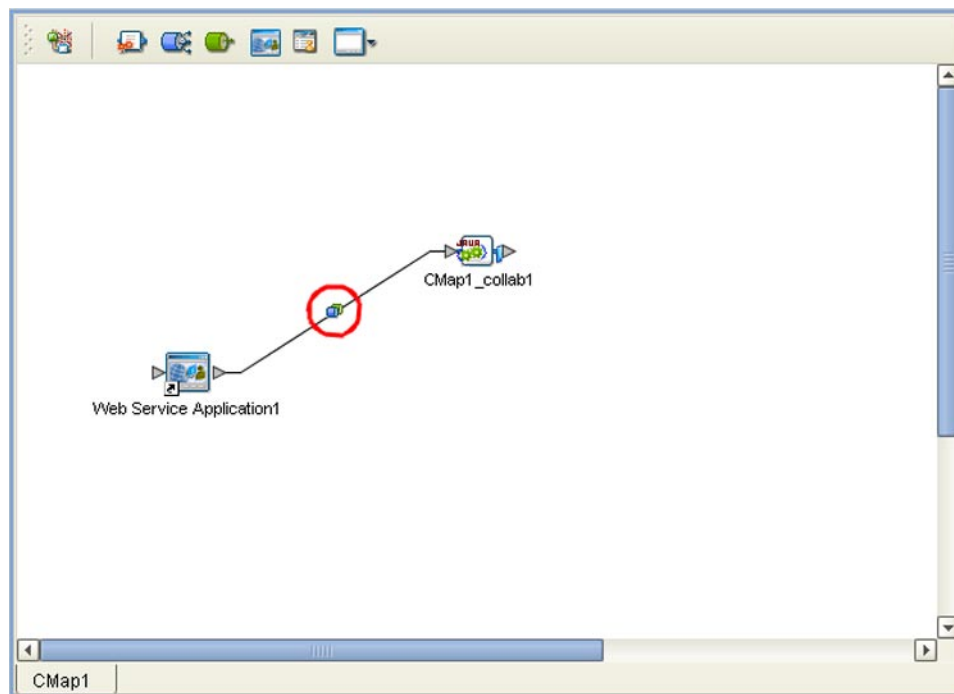
- 13 Assuming your files were imported successfully, click **Finish** and the Web Service Definition Editor opens.

## 12.7 Configuring WSDL Binding Properties

You must explicitly specify the transport bindings that are available for given WSDL operations. You can do this by using the WSDL Binding Editor to configure certain properties of the Web Service External Application Connector.

The WSDL Binding Editor is invoked by double-clicking the Web Service External Application Connector icon in the Connectivity Map (see Figure 441). The editor appears as shown in the following sections; icons appearing in the toolbar are described in Table 136.


**Figure 441** Web Service Connection



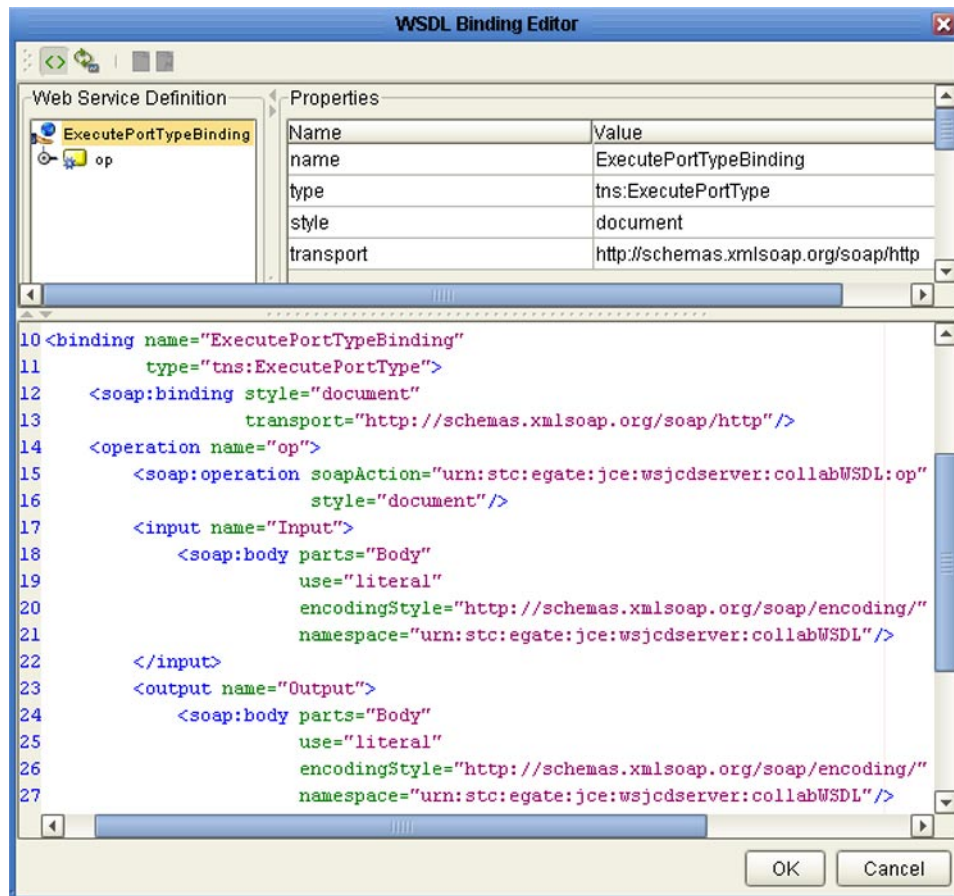
**Table 136** WSDL Binding Editor Toolbar Icons

Icon	Command	Action
	Show Web Service Definition Code	Displays the WSDL code (after generation) in the lower half of the WSDL Binding Editor (see <a href="#">Figure 442 on page 497</a> ).
	Generate Web Service Definition	Generates a Web Service Definition based on the properties set in the WSDL Binding Editor.
	Add Header	Adds a header to the selected input or output message. (Enabled only for input or output messages; see <a href="#">Figure 445 on page 500</a> .)

**Table 136** WSDL Binding Editor Toolbar Icons

Icon	Command	Action
	Add Header Fault	Adds a fault (error message) to the selected header. (Enabled only for input or output message headers; see <a href="#">Figure 447 on page 502.</a> )

**Figure 442** WSDL Binding Editor Displaying Code

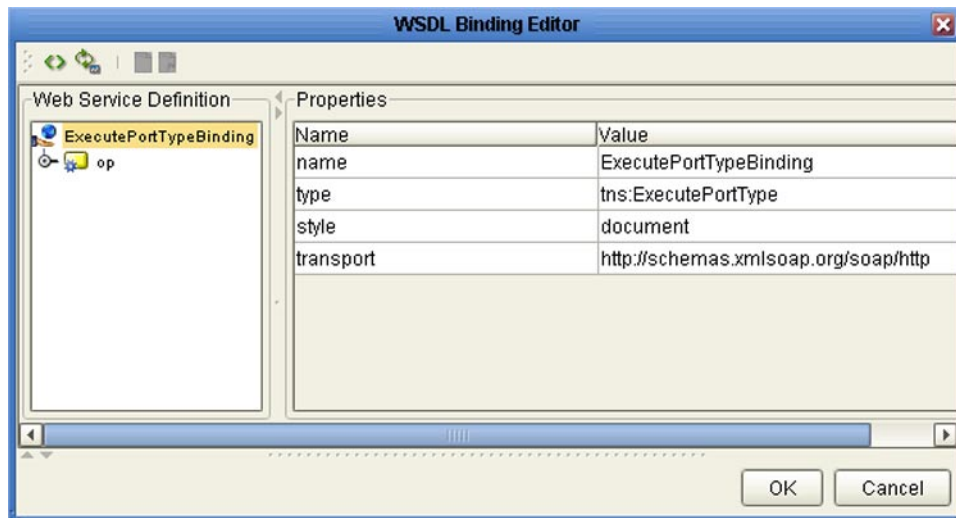


## 12.7.1 Configuring the PortType Binding

### PortTypes

PortTypes represent logical groupings, or sets, of operations. A Web Service External Application Connector represents a single PortType with a single operation.

**Figure 443** WSDL Binding Editor - PortType Binding Properties



**Table 137** PortType Binding Property Options

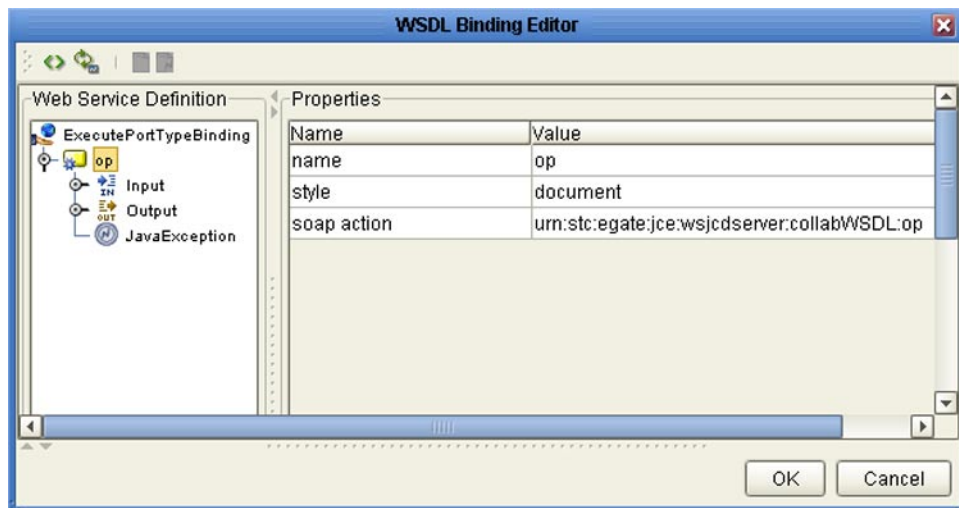
Name	Description
name	Name of the PortType Binding. The default is <b>ExecutePortTypeBinding</b> .
type	The type of binding. The default is <b>tns:ExecutePortType</b> , and cannot be changed.
style	Declares whether the interaction is document passing ( <b>document</b> ) or request/response ( <b>rpc</b> ). The default is <b>document</b> .
transport	Specifies the URL where the transport implementation can be found. The default is <b>http://schemas.xmlsoap.org/soap/http</b> .

## 12.7.2 Configuring the Operation

### Operations

Operations refer to the type of operation that the web service is to perform on behalf of a message or set of messages. Adding only an input message defines the operation to be *one-way*. Adding only an output message defines a *solicit response* operation. Adding both an input and an output message defines a *request/response* operation. Both are added by default in the Binding Editor.

**Figure 444** WSDL Binding Editor - Operation Properties



**Table 138** Operation Property Options

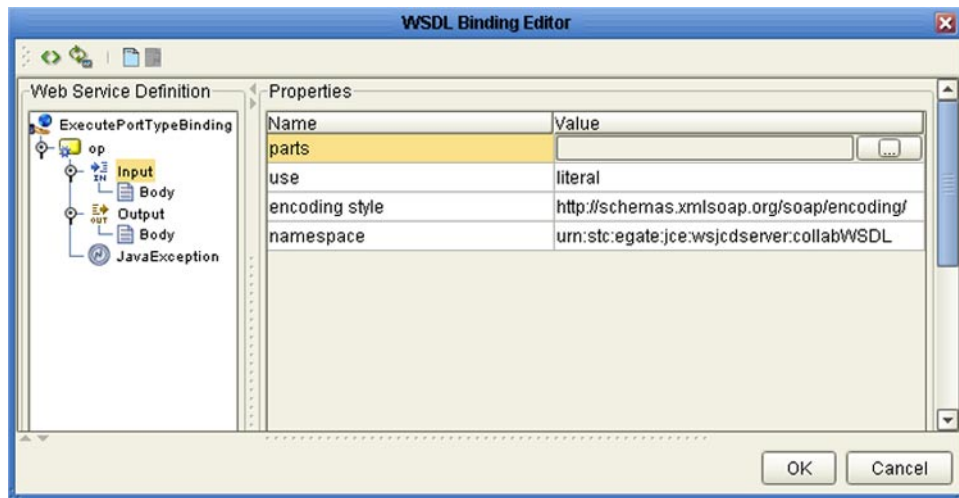
Name	Description
name	Name of the operation. The default is <b>op</b> , and cannot be changed.
style	Declares whether the interaction is document passing ( <b>document</b> ) or request/response ( <b>rpc</b> ). The default is <b>document</b> .
soap action	Specifies the Uniform Resource Name (URN) for the action. The default is <b>urn:stc:egate:jce:wsjcdserver:collabWSDL:op</b> .

## 12.7.3 Configuring Messages

### SOAP Messages

Messages, also referred to as message envelopes, represent abstract definitions of the data. They may be entire documents, or arguments that can be mapped to method invocations. Messages are either *Input* or *Output*; both have the same the property options. Clicking the **Add Header** icon in the toolbar adds a header to the message.

**Figure 445** WSDL Binding Editor - Message Properties



**Table 139** Message Property Options

Name	Description
part	Clicking the ellipsis (...) button displays a dialog containing a list of existing parts, from which you can select a part to associate with this message.
use	Allowed values are <b>encoded</b> and <b>literal</b> . The default is <b>literal</b> .
encoding style	Specifies the Uniform Resource Locator (URL) referencing the optional SOAP encoding style to be used in marshaling the associated message. The default is <b>http://schemas.xmlsoap.org/soap/encoding/</b> .
namespace	Specifies the Uniform Resource Name (URN) for the application-defined namespace. The default is <b>urn:stc:egate:jce:wsjcdserver:collabWSDL</b> .

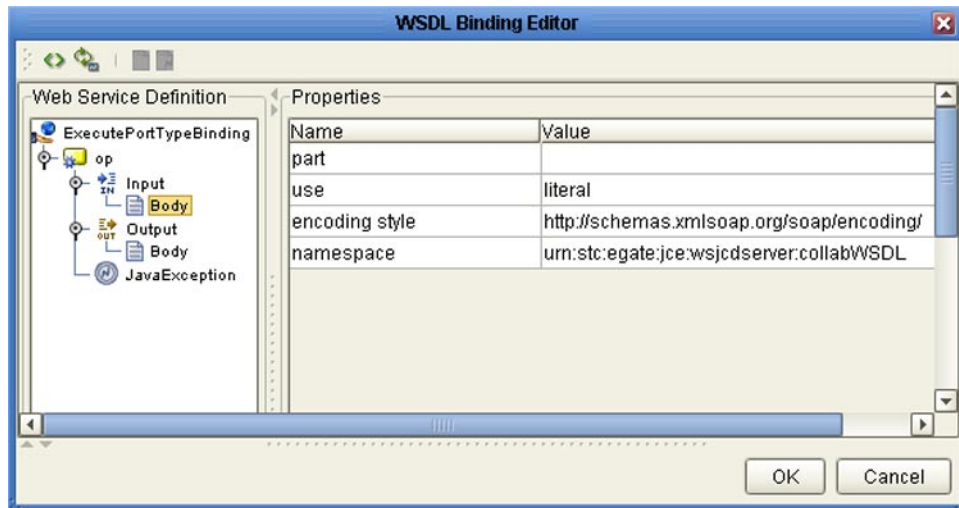


## 12.7.4 Configuring the Message Body

### Message Body

The message body contains the application-defined XML data in the SOAP message, and is the only mandatory part of the message envelope.

**Figure 446** WSDL Binding Editor - Message Body Configuration



**Table 140** Message Body Property Options

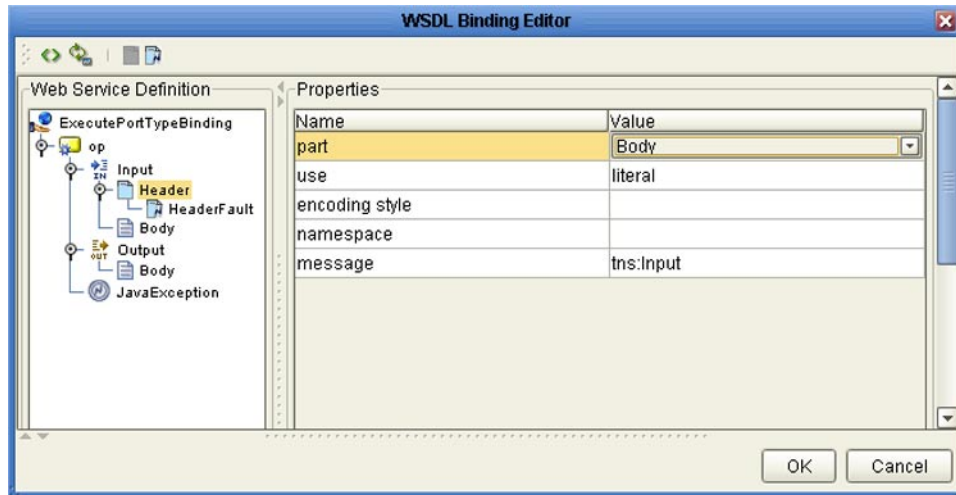
Name	Description
part	Clicking the ellipsis (...) button displays a dialog containing a list of existing parts, from which you can select a part to associate with this message body.
use	Allowed values are <b>encoded</b> and <b>literal</b> . The default is <b>literal</b> .
encoding style	Specifies the Uniform Resource Locator (URL) referencing the optional SOAP encoding style to be used in marshaling the message body. The default is <b>http://schemas.xmlsoap.org/soap/encoding/</b> .
namespace	Specifies the Uniform Resource Name (URN) for the application-defined namespace. The default is <b>urn:stc:egate:jce:wsjcdserver:collabWSDL</b> .

## 12.7.5 Configuring the Message Header

### Message Header

The SOAP header is an optional part of the SOAP message, and can occur multiple times. Clicking the **Add Header Fault** icon in the toolbar adds a fault to the header.

**Figure 447** WSDL Binding Editor - Input Header Properties



**Table 141** Header Property Options

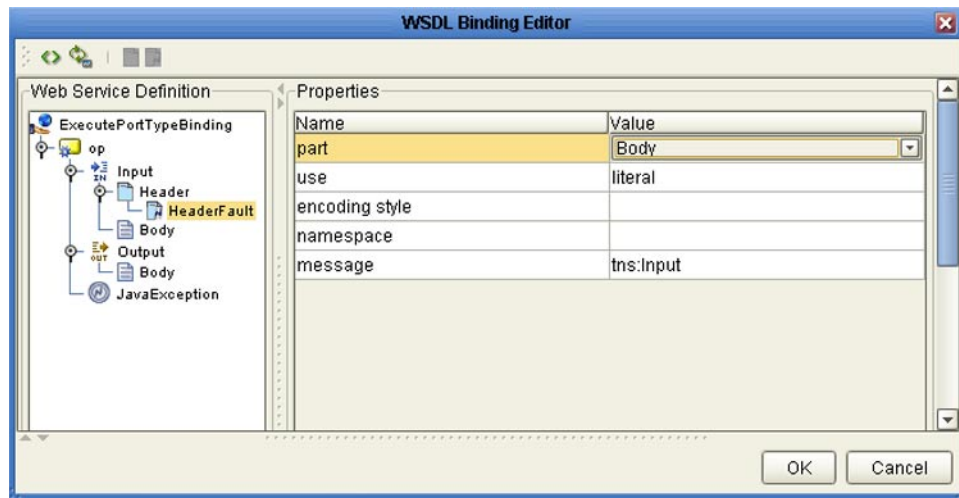
Name	Description
part	Clicking the arrow button displays a menu listing allowed parts to associate with this message header.
use	Allowed values are <b>encoded</b> and <b>literal</b> . The default is <b>literal</b> .
encoding style	Specifies the Uniform Resource Locator (URL) referencing the optional SOAP encoding style to be used in marshaling the associated message header. There is no default value.
namespace	Specifies the Uniform Resource Name (URN) for the namespace. There is no default value.
message	Specifies the message type. The default values are <b>tns:Input</b> and <b>tns:Output</b> .

## 12.7.6 Configuring a Header Fault

### Header Fault

Fault messages can optionally be added to a message header.

**Figure 448** WSDL Binding Editor - Header Fault Properties



**Table 142** Header Fault Property Options

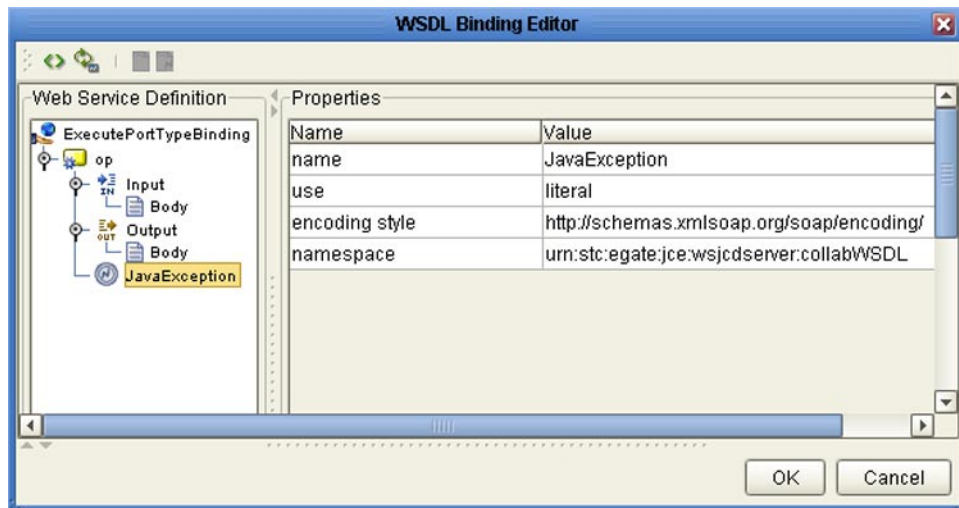
Name	Description
part	Clicking the arrow button displays a menu listing allowed parts to associate with this header fault.
use	Allowed values are <b>encoded</b> and <b>literal</b> . The default is <b>literal</b> .
encoding style	Specifies the Uniform Resource Locator (URL) referencing the optional SOAP encoding style to be used in marshaling the associated fault message. There is no default value.
namespace	Specifies the Uniform Resource Name (URN) for the namespace. There is no default value.
message	Edit this field to define the error message.

## 12.7.7 Configuring a Java Exception

### Java Exception

Java Exception messages can optionally be added to a message header.

**Figure 449** WSDL Binding Editor - Java Exception Properties



**Table 143** Java Exception Property Options

Name	Description
name	Name of the exception. The default is <b>JavaException</b> , and cannot be changed.
use	Allowed values are <b>encoded</b> and <b>literal</b> . The default is <b>literal</b> .
encoding style	Specifies the Uniform Resource Locator (URL) referencing the optional SOAP encoding style to be used in marshaling the exception message. The default is <b>http://schemas.xmlsoap.org/soap/encoding/</b> .
namespace	Specifies the Uniform Resource Name (URN) for the application-defined namespace. The default is <b>urn:stc:egate:jce:wsjcdserver:collabWSDL</b> .

# Run-time Environments

This chapter describes the process of defining run-time Environments, and the various components of an Environment.

## What's in This Chapter

- [Overview](#) on page 505
- [Using the Environment Editor](#) on page 506
- [Using Environment Explorer](#) on page 507
- [Environment Components](#) on page 509
- [Environmental Constants](#) on page 512
- [Keystores](#) on page 514
- [Logical Hosts and Domains](#) on page 515
- [Integration Servers](#) on page 518
- [Message Servers](#) on page 521
- [Schedulers](#) on page 523
- [SOAP/HTTP Web Service External Systems](#) on page 524
- [UDDI External Systems](#) on page 538

---

## 13.1 Overview

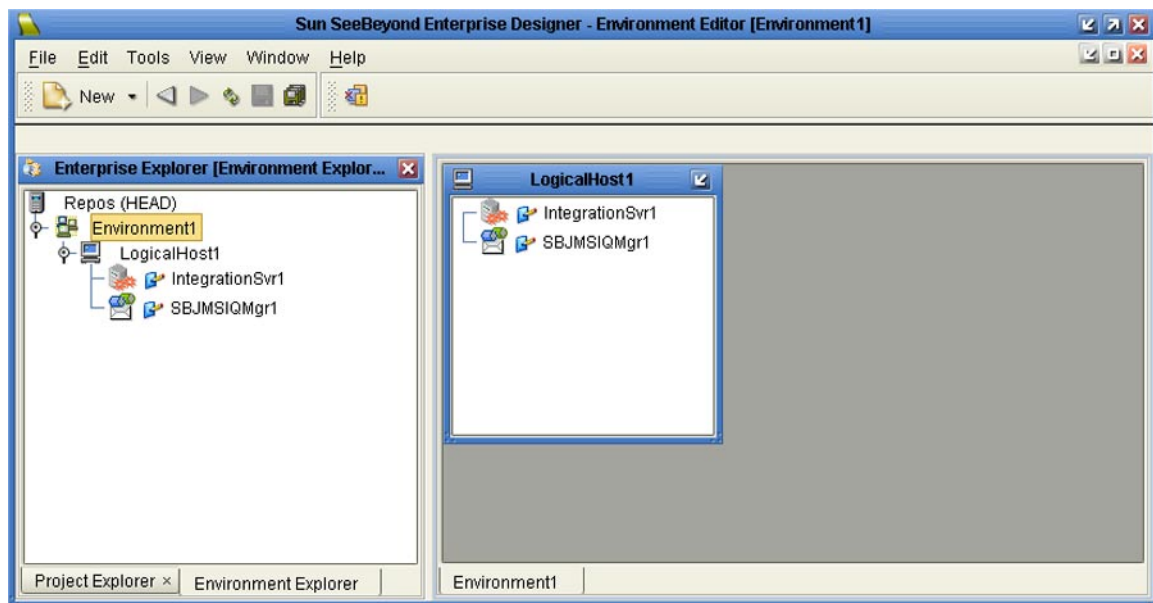
Projects are run within *domains*, which contain the logical resources required by the Project at run time. Each domain contains a single instance of an application server and a single instance of a message server. The domains, in turn, are defined within run-time *Environments*, which represent the physical resources required to implement the Project. The Environment also contains information about external systems with which the Project interacts.

## 13.2 Using the Environment Editor

The Environment Editor provides a canvas in which you create and customize a run-time Environment. Here you can see the various components (Logical Hosts, servers, and external systems) included in the selected Environment.

Clicking an Environment icon in the Environment Explorer invokes the Environment Editor, which provides a canvas in which you can create and customize an Environment (see Figure 450).

**Figure 450** Environment Editor

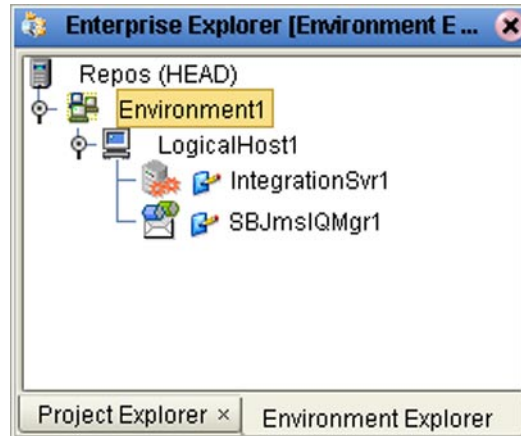


Here you can see the various components (Logical Hosts, servers, and external systems) included in the selected Environment. New Environments are added through the use of the Repository context menu (see [Using the Repository Context Menu](#) on page 59). Components are added to the Environment by selecting options in the Environment and Logical Host context menus (see [Environment Components](#) on page 509 and [Logical Hosts and Domains](#) on page 515, respectively).

## 13.3 Using Environment Explorer

Environment Explorer deals with physical resources, including integration and message servers (see Figure 451). The Environment it represents also contains information about external systems which may be involved with an eGate Integrator configuration. Environment Explorer is used in conjunction with the Environment Editor's canvas to create and configure the components of a run-time Environment.

**Figure 451** Enterprise Explorer: Environment Explorer View













Each component in Environment Explorer has an icon to identify the component type (see [Environment Explorer Icons](#) on page 508).

Right-clicking on a component in the Environment Explorer displays a context menu for that component, from which you can select various options. Only those menu options that are allowed for the component in its current state are activated.

### 13.3.1 Environment Explorer Icons

The icons described in Table 144 appear in the Environment Explorer.

**Table 144** Environment Icons

Icon	Function
	Represents the <b>Repository</b> , which is the central database where all Project information is saved. Binary files required at run time are also stored here.
	Represents a run-time <b>Environment</b> , which contains Logical Hosts and information about external systems.
	Represents a <b>Logical Host</b> or <b>Domain</b> , which contains the various logical components and files that are required at run time. See <a href="#">Logical Hosts and Domains</a> on page 515.
	Represents an <b>Environmental Constant</b> , which you can use to automate eWay and message destination configuration changes. See <a href="#">Environmental Constants</a> on page 512.
	Represents a <b>Scheduler</b> external system, which is an Environmental container for a Scheduler adapter. See <a href="#">Schedulers</a> on page 119.
	Represents a <b>SOAP/HTTP Web Service External System</b> , which is a web service or web services client. See <a href="#">SOAP/HTTP Web Service External Systems</a> on page 524.
	Represents a <b>UDDI External System</b> , which represents a UDDI Registry server. See <a href="#">UDDI External Systems</a> on page 538.
	Represents an <b>External System</b> (such as a file) that is accessed through an eWay. See the <i>Sun SeeBeyond eGate™ Integrator System Administration Guide</i> for information on keystores, and the individual eWay User's Guides regarding the external systems.
	Represents a <b>Sun SeeBeyond Integration Server</b> or third-party <b>application server</b> , which manages the Collaborations and other process interactions of a Project. The integration server is deployed to a Logical Host. See <a href="#">Integration Servers</a> on page 518.
	Represents a <b>Sun SeeBeyond JMS IQ Manager</b> or third-party <b>message server</b> , which is used to store and forward eGate Integrator system messages. The message server is deployed to a Logical Host. See <a href="#">Message Servers</a> on page 521.



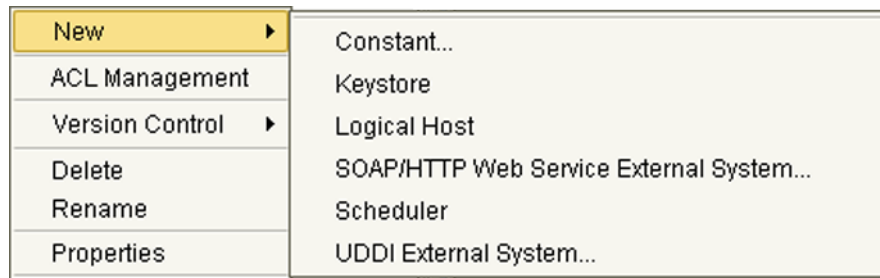
## 13.4 Environment Components

You create an Environment by selecting the **New Environment** option from the *Repository* context menu in Environment Explorer (see [Using the Repository Context Menu](#) on page 59). The different components available to you to incorporate into a run-time Environment are represented in the context menu.

### 13.4.1 Using the Environment Context Menu

Right-clicking an **Environment** in Environment Explorer displays the context menu shown in Figure 452, which shows the basic set of components. Additional external systems may be displayed in your user interface, depending upon which eWays you have installed.

**Figure 452** Environment Context Menu



**Table 145** Environment Context Menu Options

Option		Function
New	Constant	Presents a dialog with which you can add a constant to the selected Environment. See <a href="#">Environmental Constants</a> on page 512.
	Keystore	Adds a new keystore to the selected Environment. See <a href="#">Keystores</a> on page 514.
	Logical Host	Adds a new Logical Host to the selected Environment. See <a href="#">Logical Hosts and Domains</a> on page 515.
	SOAP/HTTP Web Service External System	Adds a SOAP/HTTP Web Service External System to the selected Environment. See <a href="#">SOAP/HTTP Web Service External Systems</a> on page 524.
	Scheduler	Presents a dialog with which you can add a new scheduling component to the selected Environment. See <a href="#">Schedulers</a> on page 523.
	UDDI External System	Adds a UDDI External System to the selected Environment. See <a href="#">UDDI External Systems</a> on page 538.

**Table 145** Environment Context Menu Options

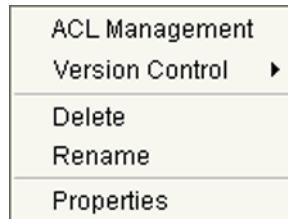
Option		Function
ACL Management		Presents a dialog with which an Administrator can assign read/write privileges to users for the selected Environment. See the <i>Sun SeeBeyond eGate™ Integrator System Administration Guide</i> .
Version Control	Check In	Presents a dialog with which you can check in a new version of the selected Environment. Refer to <a href="#">Checking a Component In</a> on page 80 for more details.
	Check Out	Presents a dialog with which you can check out the current version of the selected Environment. See <a href="#">Checking a Component Out</a> on page 82 for more information.
	Tag	Presents a dialog with which you can specify a tag to attach to the selected Environment.
Delete		<p>Deletes the selected Environment, subject to the following conditions:</p> <ul style="list-style-type: none"> <li>▪ You have <i>write</i> privileges for Environments (see ACL Management, above).</li> <li>▪ The Environment is not checked out by anyone other than yourself.</li> </ul> <p>If these conditions are met, a dialog is presented in which you confirm that you want to delete the selected Environment. Clicking <b>Yes</b> then deletes the Environment.</p>
Rename		Activates the field, allowing you to rename the selected Environment.
Properties		Presents a dialog showing the configuration properties of the selected Environment.

**Important:** All Environment component names should contain only alphanumeric characters (letters and numbers), dashes, and underscores.

## 13.4.2 Using Environment Component Context Menus

Most Environment components have context menus similar to that shown in Figure 453. Those context menus that are different are described in the sections for the specific components.

**Figure 453** Environment Component Context Menu



**Table 146** Environment Component Context Menu Options

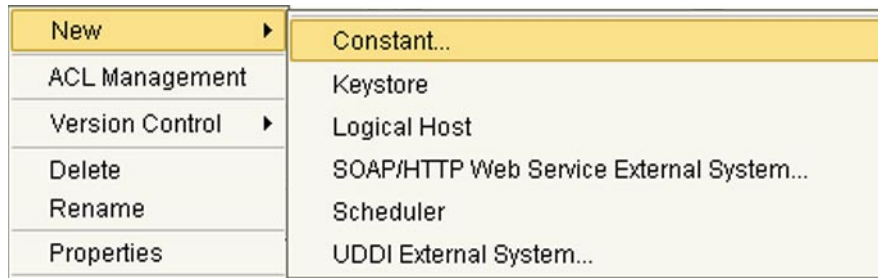
Option	Function
ACL Management	Presents the ACL Properties dialog, with which an Administrator can assign read/write privileges to users for the selected component. See the <i>Sun SeeBeyond eGate™ Integrator System Administration Guide</i> .
Version Control	Accesses the version control features for the selected component. See <a href="#">Accessing Component Version Control Features</a> on page 79 for additional information.
Delete	Deletes the selected component, subject to the following conditions: <ul style="list-style-type: none"> <li>▪ You have <i>write</i> privileges for the component (see ACL Management, above).</li> <li>▪ You have the component checked out to your workspace.</li> <li>▪ The component is not checked out by anyone other than yourself.</li> </ul> If these conditions are met, a dialog is presented in which you confirm that you want to delete the selected component. Clicking <b>Yes</b> then deletes the component.
Rename	Activates the field, allowing you to rename the selected component.
Properties	Presents a dialog showing the configuration properties of the selected Environment.

**Important:** All Environment component names should contain only alphanumeric characters (letters and numbers), dashes, and underscores.

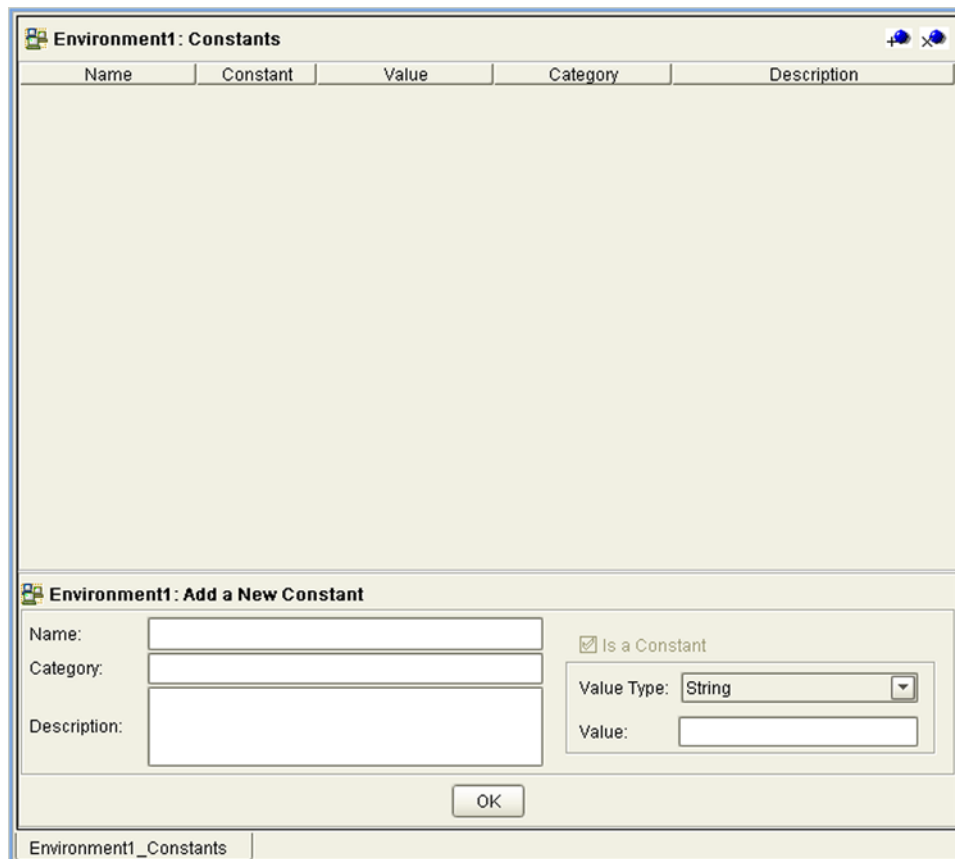
## 13.5 Environmental Constants

Environmental constants are name-value pairs that are visible across the Environment. Selecting the **New > Constant** option from the Environment context menu (see Figure 454) displays the *Environmental Constants* panel shown in Figure 455.

**Figure 454** Environment Context Menu - New Constant





**Figure 455** Environmental Constants Panel



All constants defined for the specific Environment are listed in the *Constants* section of the panel, along with their various properties. New constants are added using the *Add a New Constant* section of the panel.

**Note:** When you create an *Environmental constant*, you assign a permanent value to it which cannot be overridden.

**Table 147** Environmental Constants Panel Icons

Icon	Name	Function
	Add a New Constant	Adds a new constant to the list.
	Delete a Highlighted Constant	Deletes the selected constant from the list.

### 13.5.1 Using the Constant Context Menu

Right-clicking the Constant in Environment Explorer displays its context menu (see [Using Environment Component Context Menus](#) on page 511).

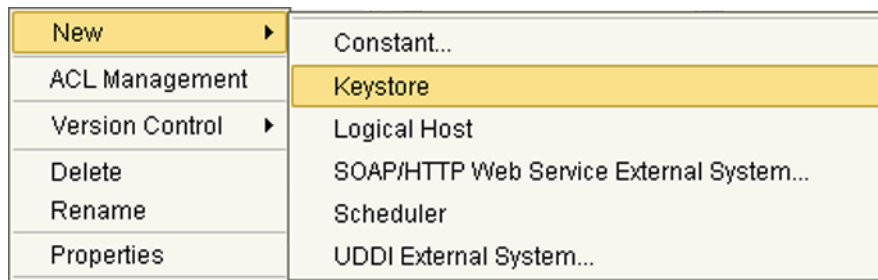
## 13.6 Keystores

The *keystore* is a security database, containing keys and certificates, that is used to implement the Secure Messaging Extension (SME) encryption protocol. SME is not currently supported in Java CAPS.

Selecting the **New > Keystore** option from the Environment context menu (see Figure 456) first displays a dialog in which you provide a name for the keystore, then adds the keystore to your Environment.

**Important:** All Environment component names should contain only alphanumeric characters (letters and numbers), dashes, and underscores.

**Figure 456** Environment Context Menu - New Keystore



### 13.6.1 Using the Keystore Context Menu

Right-clicking the Keystore in Environment Explorer displays its context menu (see Figure 457).

**Figure 457** Keystore Context Menu



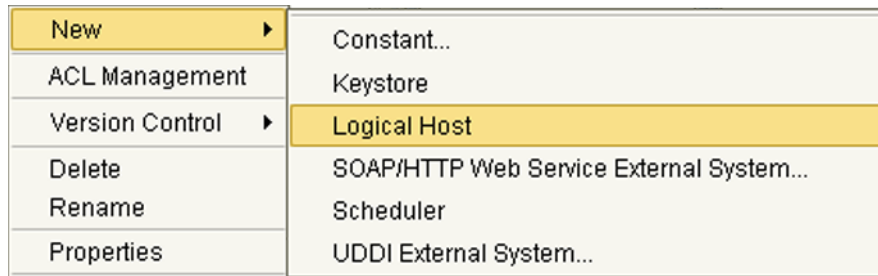
**Table 148** Keystore Context Menu Options

Option	Function
Rename	Activates the field, allowing you to rename the selected Environment.
Manage Private Keys	Presents a dialog with which you can edit the private keys in the selected keystore.
Manage Public Certificates	Presents a dialog with which you can edit the public certificates in the selected keystore.
Manage Trust Stores	Presents a dialog with which you can edit the trust stores in the selected keystore.

## 13.7 Logical Hosts and Domains

You create this component by selecting the **New > Logical Host** option from the *Environment* context menu in Environment Explorer (see Figure 458).

**Figure 458** Environment Menu - New Logical Host



A *domain* is an instance of a *Logical Host*; that is, the Logical Host used in Enterprise Designer acts as a directory for the domains, which are the actual run-time components. Each domain consists of two main components: an integration (or application) server and a message server.

**Note:** The eGate run-time components, which include domains and any Projects or components deployed to them, provide their full functionality independent of the Repository.

### 13.7.1 Using the Logical Host Context Menu

Right-clicking on a Logical Host in Environment Explorer displays the context menu shown in Figure 459.

**Figure 459** Logical Host Context Menu



**Table 149** Logical Host Context Menu Options

Option		Function
New	Sun Java System JMS Server	Adds a new Sun Java System JMS Server to the selected Logical Host. See the <i>Sun SeeBeyond eGate™ Integrator JMS Reference Guide</i> .
	Sun SeeBeyond JMS IQ Manager	Adds a new Sun SeeBeyond JMS IQ Manager to the selected Logical Host. See <a href="#">Message Servers</a> on page 521.
	Sun Java System Application Server	Adds a new Sun Java System Application Server to the selected Logical Host. See the <i>Sun SeeBeyond eGate™ Integrator JMS Reference Guide</i> .
	WebSphere MQ	Adds a new IBM WebSphere MQ message server to the selected Logical Host. See <a href="#">Message Servers</a> on page 521.
	Sun SeeBeyond Integration Server	Adds a new Sun SeeBeyond Integration Server to the selected Logical Host. See <a href="#">Integration Servers</a> on page 518.
ACL Management		Presents a dialog with which an Administrator can assign read/write privileges to users for the selected Logical Host. See the <i>Sun SeeBeyond eGate™ Integrator System Administration Guide</i> .
Version Control		Accesses the version control features for the selected component. See <a href="#">Accessing Component Version Control Features</a> on page 79 for additional information.
Delete		<p>Deletes the selected Logical Host, subject to the following conditions:</p> <ul style="list-style-type: none"> <li>▪ You have <i>write</i> privileges for the component (see ACL Management, above).</li> <li>▪ You have the component checked out to your workspace.</li> <li>▪ The component is not checked out by anyone other than yourself.</li> </ul> <p>If these conditions are met, a dialog is presented in which you confirm that you want to delete the selected component. Clicking <b>Yes</b> then deletes the component.</p>
Rename		Activates the field, allowing you to rename the selected Logical Host.

**Important:** All Environment component names should contain only alphanumeric characters (letters and numbers), dashes, and underscores.



## 13.7.2 Configuring Domains

There is no configuration performed on the Logical Host, since it is a directory. All configuration of the individual domains is performed from either the Domain Manager or the command line. See the *Sun SeeBeyond eGate™ Integrator System Administration Guide*.

## 13.8 Integration Servers

The Logical Host contains one or more instances of a J2EE-compatible integration server. This server is the engine that runs Collaborations for processing business logic, and eWays for communicating with external applications. The integration server provides services for security, transactions, business rules execution, and connectivity management. eGate Integrator contains the Sun SeeBeyond Integration Server and the Sun Java System Application Server; see the *Sun SeeBeyond eGate™ Integrator System Administration Guide* for security-related configuration information.

**Note:** You can also deploy custom EAR and WAR files to the integration servers; see the *Sun Java Application Server* documentation for detailed information.

You create this component by selecting the **New > [Integration Server Type]** option from the *Logical Host* context menu in Environment Explorer (for example, a Sun SeeBeyond Integration Server as shown in Figure 460).

**Figure 460** Logical Host Menu - New Integration Server



### 13.8.1 Using the Integration Server Context Menu

Right-clicking an integration server in Environment Explorer displays its context menu (see [Using Environment Component Context Menus](#) on page 511). In addition to the standard menu options described in the referenced section, the integration server context menu contains the *Java Debugger* option, shown in Figure 461. Selecting this option displays the Java Debugger, which enables you to debug Java-based Collaboration Definitions as deployed within the integration server, during the development phase (see [Using the Java Debugger](#) on page 303). This menu is used for all supported integration servers.

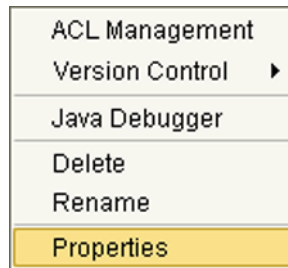
**Figure 461** Integration Server Context Menu - Java Debugger



## 13.8.2 Configuring an Integration Server

Selecting **Properties** from the integration server context menu (see Figure 462) displays the dialog set shown in this section, in which you assign the desired values for the listed properties.

**Figure 462** Integration Server Context Menu - Properties



## Sun SeeBeyond Integration Server Configuration

The properties shown in Figure 463 cover the basic configuration of the Sun SeeBeyond Integration Server for purposes of generating application files only (see [Building an Application File](#) on page 139). They do not affect operation of the integration server itself.

This dialog is used for all supported integration servers. The configuration property values, however, are different for the different servers.

**Figure 463** Sun SeeBeyond Integration Server Configuration Properties



**Table 150** Sun SeeBeyond Integration Server Configuration Properties List

Property	Description
Integration Server URL	URL of the Integration Server, showing the host and port on which the Integration Server is running. The default value is <b>stcis://localhost:18000</b> .
Username	User name for accessing the Integration Server. The default value is <b>Administrator</b> .

**Table 150** Sun SeeBeyond Integration Server Configuration Properties List

Property	Description
Password	User password for accessing the Integration Server. There is no default value.
Debug port	The port on which the Java Debugger will connect to the integration server. The value should match the actual debug port, and debugging must be enabled (see Configuring the Sun SeeBeyond Integration Server in the <i>Sun SeeBeyond eGate™ Integrator System Administration Guide</i> ).
Application Workspace Directory	The directory where application data will be kept at run time. There is no default value.
Number of Local Transaction Resources in a Transaction	Specifies the number of local resource managers that are allowed to participate in a J2EE container-managed transaction, if there is a XA resource manager involved in that transaction. There is no default value.

**Important:** User names and passwords should contain only alphanumeric characters (letters and numbers), dashes, and underscores.

## eInsight Engine Configuration

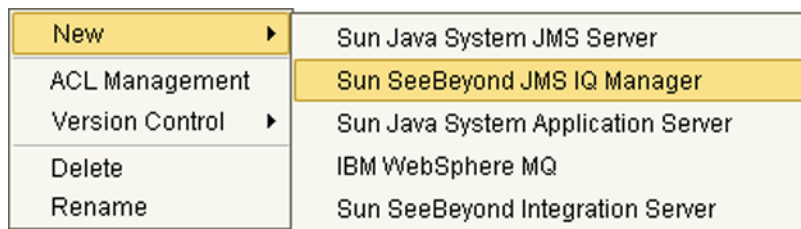
This configuration node is operational only if you have eInsight Business Process Manager installed on your system. The configuration properties relate to the BPEL engine's database cache; see the *Sun SeeBeyond eInsight™ Business Process Manager User's Guide* for information regarding these properties.

## 13.9 Message Servers

Each domain supports a single message server, which provides the Java Messaging Service (JMS) implementation. eGate Integrator includes the Sun SeeBeyond JMS IQ Manager, which conforms to the Java Message specification 1.1 and supports both topic (publish-and-subscribe) and queue (point-to-point) messaging styles. eGate Integrator also supports the Sun Java System JMS Server and IBM WebSphere MQ. For comprehensive information on the Sun SeeBeyond JMS IQ Manager, see the *Sun SeeBeyond eGate™ Integrator JMS Reference Guide*.

You create this component by selecting the **New > [Message Server Type]** option from the *Logical Host* context menu in Environment Explorer (for example, a Sun SeeBeyond JMS IQ Manager as shown in Figure 468).

**Figure 464** Logical Host Menu - New Message Server

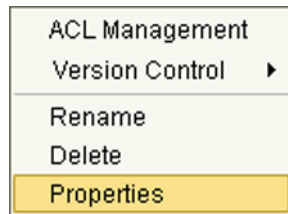


Entering a name and clicking **OK** adds the UDDI External System to the selected Environment. Right-clicking the UDDI External System in Environment Explorer displays its context menu (see [Using Environment Component Context Menus](#) on page 511).

### 13.9.1 Configuring a Message Server

Selecting **Properties** from the message server context menu (see Figure 465) displays the dialog set shown in this section, in which you assign the desired values for the listed properties.

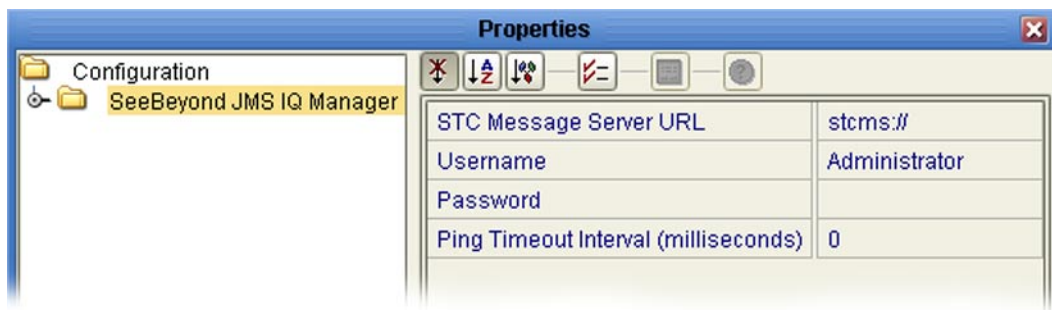
**Figure 465** Message Server Context Menu - Properties



The properties shown in Figure 466 cover the basic configuration of the Sun SeeBeyond JMS IQ Manager for purposes of generating application files only (see [Building an Application File](#) on page 139). They do not affect operation of the JMS server itself.

This dialog is used for all supported message servers. The configuration property values, however, are different for the different servers.

**Figure 466** Sun SeeBeyond JMS IQ Manager Configuration Properties



**Table 151** Sun SeeBeyond JMS IQ Manager Configuration Properties List

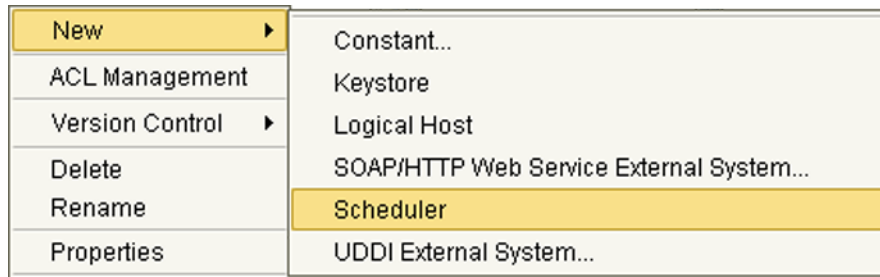
Property	Description
STC Message Server URL	URL of the message server, showing the host and port on which the message server is running. Note that <b>stcms://</b> is a valid URL, and points to the JMS message server managed by the Logical Host.
Username	User name for accessing the message server.
Password	User password for accessing the message server.
Ping Timeout Interval	Sets the timeout interval for “ping” functionality, in milliseconds.

**Important:** User names and passwords should contain only alphanumeric characters (letters and numbers), dashes, and underscores.

## 13.10 Schedulers

In an Environment, a *Scheduler* represents an external system to which you deploy a *Scheduler Adapter* created in a Project (see [Schedulers](#) on page 119). You create this component by selecting the **New > Scheduler** option from the *Environment* context menu in Environment Explorer (see Figure 467).

**Figure 467** Environment Context Menu - New Scheduler

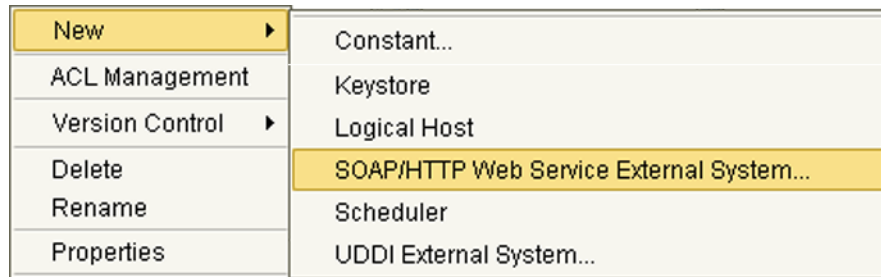


Selecting **New > Scheduler** adds the Scheduler external system to the selected Environment. Right-clicking the Scheduler in Environment Explorer displays its context menu (see [Using Environment Component Context Menus](#) on page 511).

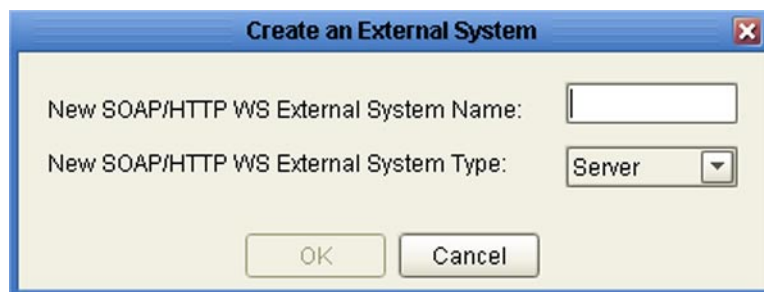
## 13.11 SOAP/HTTP Web Service External Systems

A SOAP/HTTP Web Service External System represents a system containing a Web Services Application. You create this component by selecting the **New > SOAP/HTTP Web Service External System** option from the *Environment* context menu in Environment Explorer (see Figure 468), which displays the dialog shown in Figure 469.

**Figure 468** Environment Context Menu - New SOAP/HTTP Web Service External System



**Figure 469** Create External System Dialog



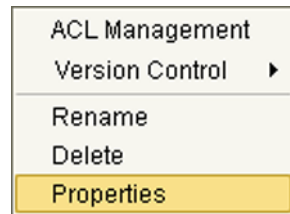
Entering a name, specifying whether it is a **Client** or **Server**, and clicking **OK** adds the UDDI External System to the selected Environment. Right-clicking the SOAP/HTTP Web Service External System in Environment Explorer displays its context menu (see [Using Environment Component Context Menus](#) on page 511).



### 13.11.1 Configuring a SOAP/HTTP Client

Selecting **Properties** from the **SOAP/HTTP Web Service External System (Client)** context menu (see Figure 470) displays the dialog set shown in this section, in which you assign the desired values for the listed properties.

**Figure 470** SOAP/HTTP Web Service External System Context Menu - Properties



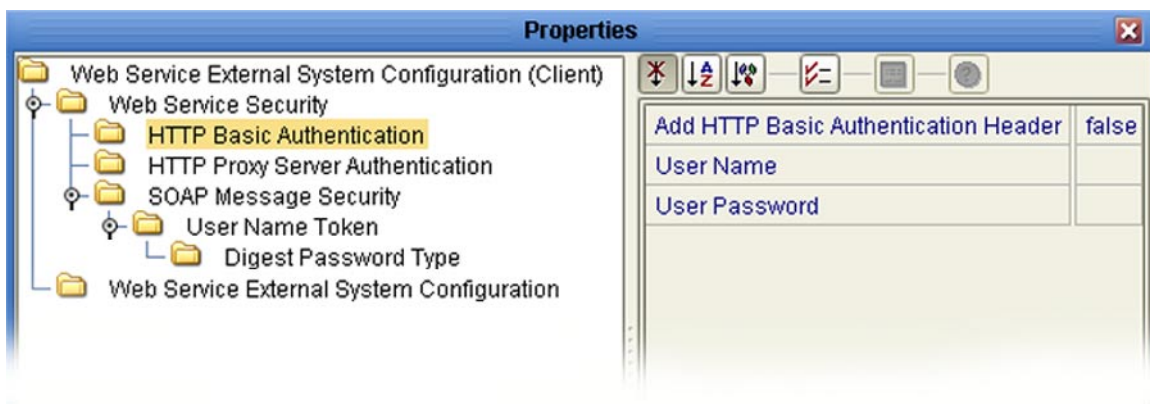
## Web Service Security: HTTP Basic Authentication

**Note:** You must have created Logical Host users prior to configuring SOAP/HTTP Web Service External System security.

With HTTP basic authentication, a client requests access to a protected resource on the server. The server sends back a request for the client's user name and password. The client sends this information. If the server verifies that the client is allowed to access the protected resource, then the server returns the resource to the client.

**Note:** By itself, this mechanism is limited. The user name and password are not encrypted; therefore, malicious users who eavesdrop on the communications between the client and server can learn the user name and password.

**Figure 471** HTTP Basic Authentication Configuration



**Table 152** HTTP Basic Authentication Configuration Properties

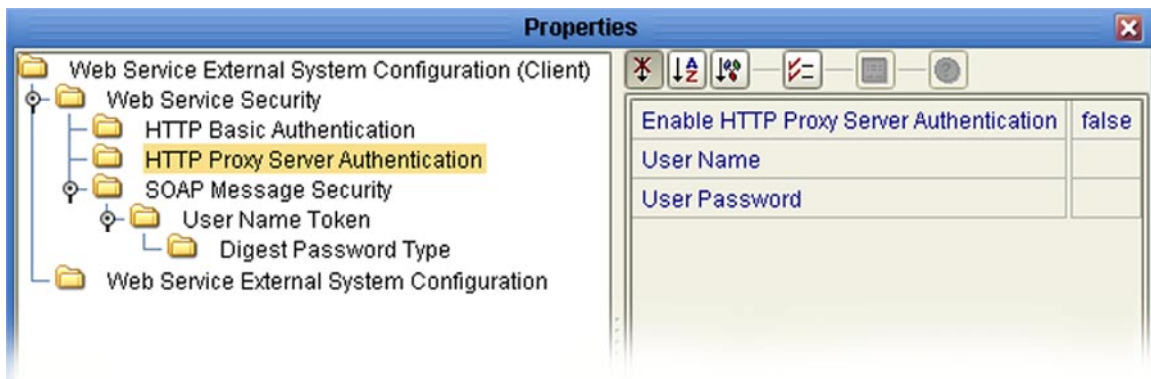
Property	Description
Add HTTP Basic Authentication Header	Specifies whether or not you want to use HTTP Basic Authentication for this web service. The default value is <b>false</b> .
User Name	Specifies the name of the Logical Host user to whom you want to grant access to the web service. This information will appear in the header.
User Password	Specifies the password for the Logical Host user to whom you want to grant access to the web service. This information will appear in the header.

## Web Service Security: HTTP Proxy Server Authentication

**Note:** You must have created Logical Host users prior to configuring SOAP/HTTP Web Service External System security.

HTTP proxy server authentication enables you to control access to the web service by means of a proxy server. First, an administrator must configure a host name and port name for the proxy server using the Integration Server Administration console. Then you can configure one or more SOAP/HTTP Web Service External Systems in client mode.

**Figure 472** HTTP Proxy Server Configuration Dialog



**Table 153** HTTP Proxy Server Configuration Properties

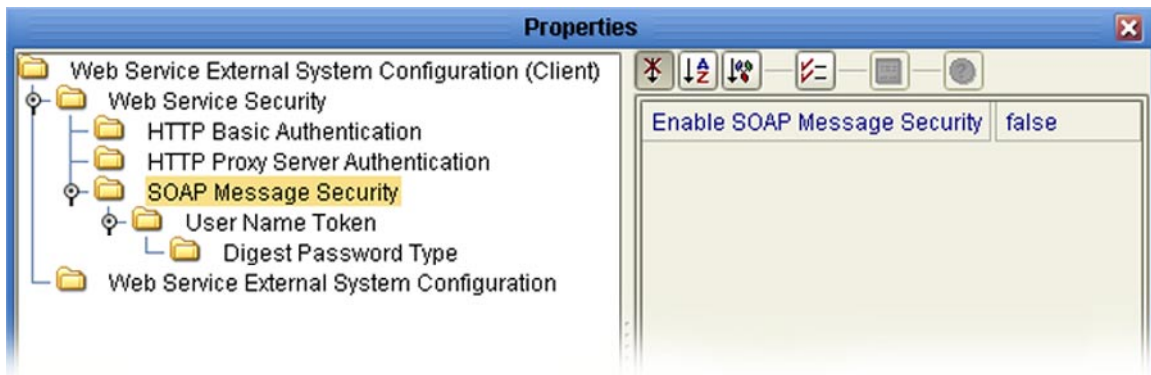
Property	Description
Enable HTTP Proxy Server Authentication	Specifies whether or not HTTP Proxy Server Authentication is to be used for this web service. The default value is <b>false</b> .
User Name	Specifies the name of the proxy server user.
User Password	Specifies the password for the proxy server user.

## Web Service Security: SOAP Message Security

**Note:** You must have created Logical Host users prior to configuring SOAP/HTTP Web Service External System security.

The sender of a SOAP message can place security-related information in a security header block. For example, the header can include a *token*, which contains one or more declarations about an entity. There are two possible types of tokens: *User Name* tokens and *Security Assertion Markup Language (SAML)* tokens. SAML tokens enable you to secure web services across heterogeneous security domains.

**Figure 473** SOAP Message Security Configuration Dialog



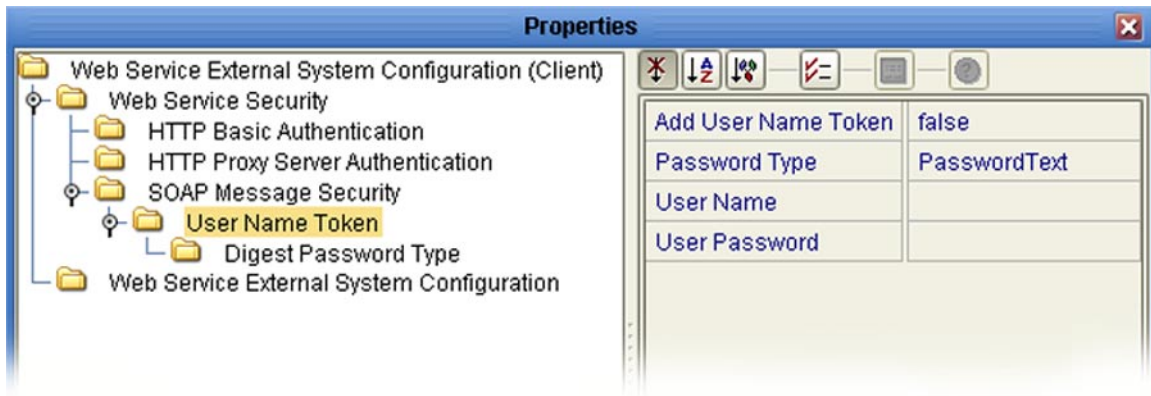
**Table 154** SOAP Message Security Configuration Properties

Property	Description
Enable SOAP Message Security	Specifies whether or not a security header block is to be used for this web service. The default value is <b>false</b> .

## User Name Token

A user name token contains a user name and password, which can be of two types: *text* and *digest*.

**Figure 474** User Name Token Configuration Dialog



**Table 155** User Name Token Configuration Properties

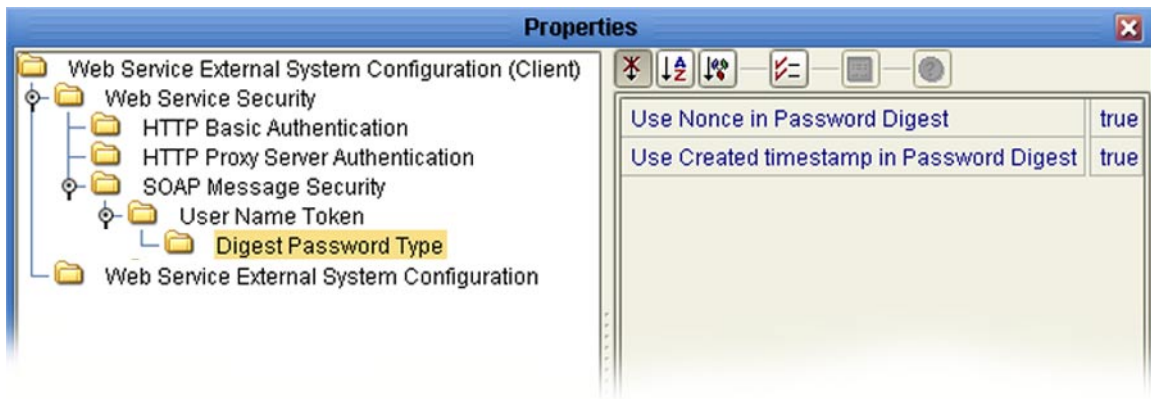
Property	Description
Add User Name Token	Specifies whether or not a User Name Token is to be used in the SOAP Message Security for this web service. The default value is <b>false</b> .
Password Type	Specifies whether you want to use a <b>digest</b> or <b>text</b> password in the token. In a <b>digest</b> type, the password is combined with the user name and (optionally) a nonce and/or creation timestamp, and then modified by an algorithm. The default value is <b>PasswordText</b> .
User Name	Specifies the name of the Logical Host user you want to use for the token.
User Password	If you specified <b>text</b> for the password type, this property specifies the password you want to use for the token. The password can be clear text or a password equivalent. A <i>clear text</i> password is the original, unmodified password. A <i>password equivalent</i> is created by applying an algorithm to the password.

## Digest Password Type

To guard against replay attacks, the security header can also contain a *nonce* and a *creation timestamp*. Using a nonce and creation timestamp in a digest password helps you to prevent *replay attacks*. In a replay attack, a malicious user eavesdrops on the communications between a sender and a receiver. The malicious user learns the sender's password (encrypted or unencrypted), and then impersonates the sender using the password.

**Important:** If you include a digest password, then you must set the default realm of the Integration Server to **wssfile**. For instructions, see "Configuring the Sun SeeBeyond Integration Server" in the Sun SeeBeyond eGate™ Integrator System Administration Guide.

**Figure 475** Digest Password Type Configuration Dialog



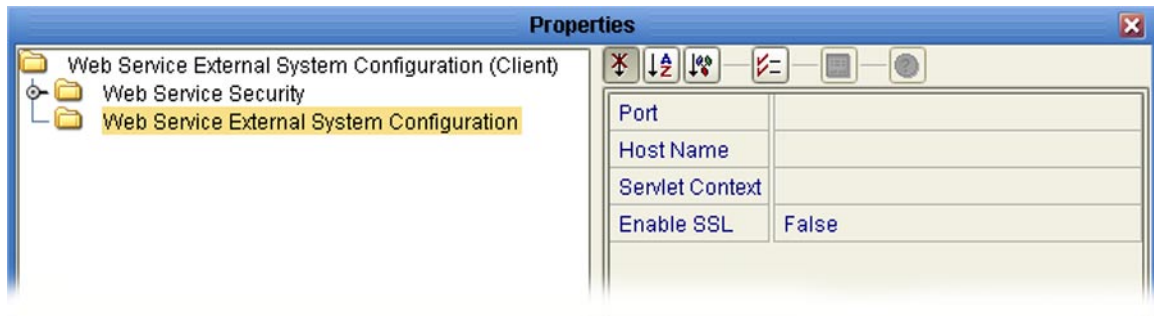
**Table 156** Digest Password Type Configuration Properties

Property	Description
Use Nonce in Password Digest	If you specified <b>PasswordDigest</b> for the User Name Token password type, this property specifies whether or not you want to use a nonce in the password digest. A <i>nonce</i> is a random value that is used only once. The default value is <b>true</b> .
Use Created Timestamp in Password Digest	If you specified <b>PasswordDigest</b> for the User Name Token password type, this property specifies whether or not you want to use a created timestamp in the password digest. The default value is <b>true</b> .

## Web Service External System Configuration

**Note:** The values you enter for these properties must match the settings on the integration server to which you will be deploying the Web Service.

**Figure 476** Web Service External System Configuration Dialog



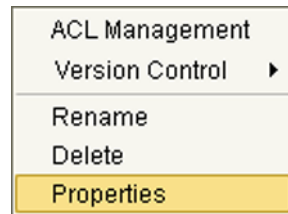
**Table 157** Web Service External System Configuration Properties

Property	Description
Port	Port on which the Web Services Application being deployed to this Web Service External System can be accessed. There is no default. <b>Note:</b> If your domain is running on the default port (18000), then the HTTP listener port is 18001. If you are using the Sun Java System Application Server, this port should match the listener port in the Application Server Administration Console.
Host Name	Name or IP address of the computer hosting the Web Services Application. There is no default.
Servlet Context	The path and name of the Web Services Application. This should be the <Servlet Context/Port Name> of the SOAP/HTTP Web Services External System (Server).
Enable SSL	Enables or disables support for Secure Socket Layer features. If you want to use the HTTPS protocol, set this property to <b>True</b> . The default is <b>False</b> .

## 13.11.2 Configuring a SOAP/HTTP Server

Selecting **Properties** from the **SOAP/HTTP Web Service External System (Server)** context menu (see Figure 477) displays the dialog set shown in this section, in which you assign the appropriate values for the listed properties.

**Figure 477** SOAP/HTTP Web Service External System Context Menu - Properties



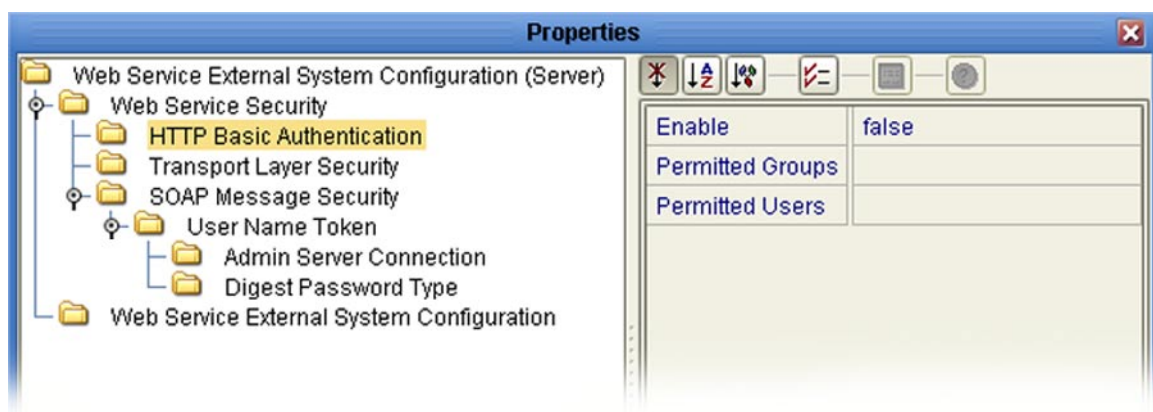
## Web Service Security: HTTP Basic Authentication

**Note:** You must have created Logical Host users prior to configuring SOAP/HTTP Web Service External System security.

With HTTP basic authentication, a client requests access to a protected resource on the server. The server sends back a request for the client's user name and password. The client sends this information. If the server verifies that the client is allowed to access the protected resource, then the server returns the resource to the client.

**Note:** By itself, this mechanism is limited. The user name and password are not encrypted; therefore, malicious users who eavesdrop on the communications between the client and server can learn the user name and password.

**Figure 478** HTTP Basic Authentication Configuration Dialog





**Table 158** HTTP Basic Authentication Configuration Properties

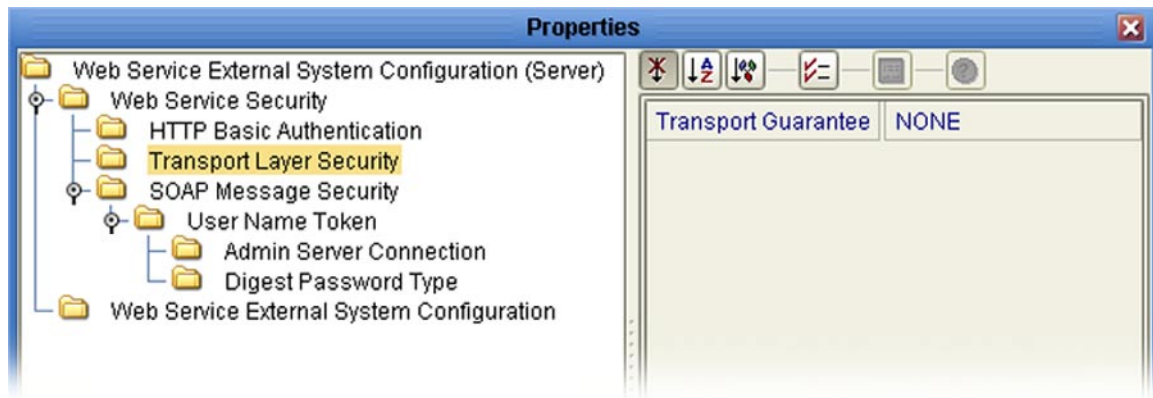
Property	Description
Enable	Specifies whether or not you want to use HTTP Basic Authentication for this web service. The default value is <b>false</b> .
Permitted Groups	Specifies the groups to which you want to grant access to the web service. Use commas to separate multiple groups.
Permitted Users	Specifies the Logical Host users to which you want to grant access to the web service. Use commas to separate multiple groups.

## Web Service Security: Transport Layer Security

**Note:** You must have created Logical Host users prior to configuring SOAP/HTTP Web Service External System security.

To secure web services using TLS, you must configure a SOAP/HTTP Web Service External System in server mode, and also on an HTTP listener as described in the *Sun SeeBeyond eGate™ Integrator System Administration Guide*. At runtime, clients are presented with a security alert.

**Figure 479** Transport Layer Security Configuration Dialog



**Table 159** Transport Layer Security Configuration Properties

Property	Description
Transport Guarantee	Specifies how strongly messages are protected in the Transport Layer. <ul style="list-style-type: none"> <li>▪ If you want messages to be visible but not modifiable, then set the value to <b>INTEGRAL</b>.</li> <li>▪ If you want messages to be neither visible nor modifiable, then set the value to <b>CONFIDENTIAL</b>.</li> <li>▪ The default value is <b>NONE</b>.</li> </ul>



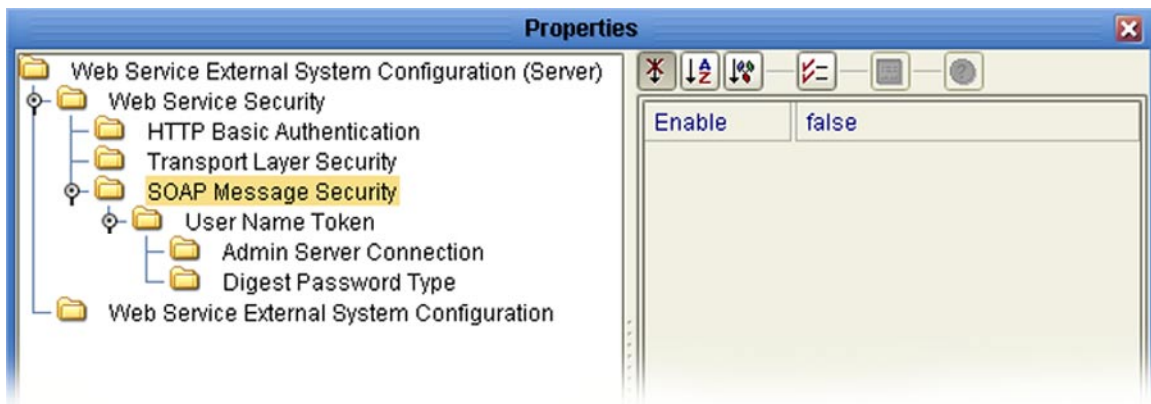
## Web Service Security: SOAP Message Security

**Note:** You must have created Logical Host users prior to configuring SOAP/HTTP Web Service External System security.

The sender of a SOAP message can place security-related information in a security header block. For example, the header can include a *token*, which contains one or more declarations about an entity. There are two possible types of tokens: *User Name* tokens and *Security Assertion Markup Language (SAML)* tokens. SAML tokens enable you to secure web services across heterogeneous security domains.

**Important:** If SOAP Message Security is enabled, the **Admin Server Connection** properties must be set correctly in order to acquire a connection for authentication and authorization of the User Name Token.

**Figure 480** SOAP Message Security Configuration Dialog



**Table 160** SOAP Message Security Configuration Properties

Property	Description
Enable	Specifies whether or not a security header block is to be used for this web service. The default value is <b>false</b> .

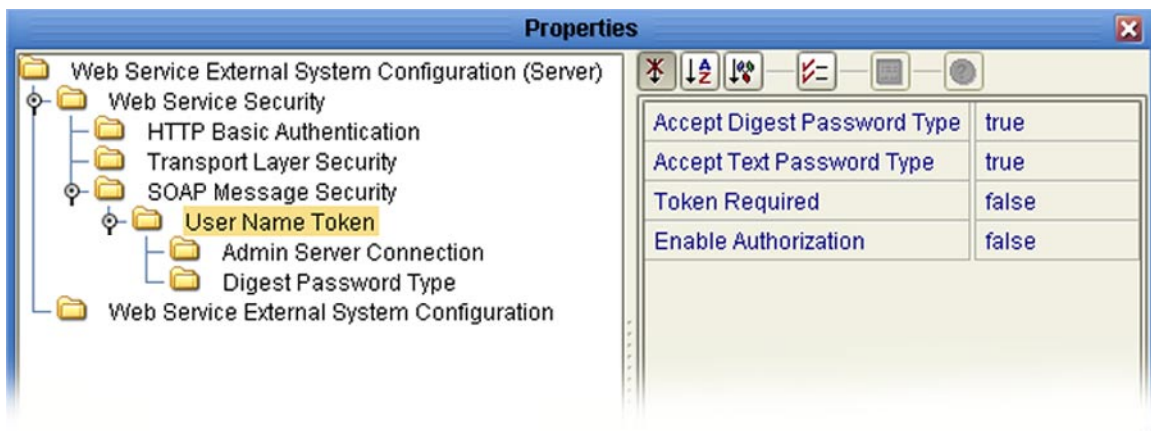
## User Name Token

A user name token contains a user name and password, which can be of two types: *text* and *digest*.

**Note:** *At least one of the **Accept <Digest/Text> Password Type** properties must be set to **true**.*

**Important:** *If SOAP Message Security is enabled, the **Admin Server Connection** properties must be set correctly in order to acquire a connection for authentication and authorization of the User Name Token.*

**Figure 481** User Name Token Configuration Dialog



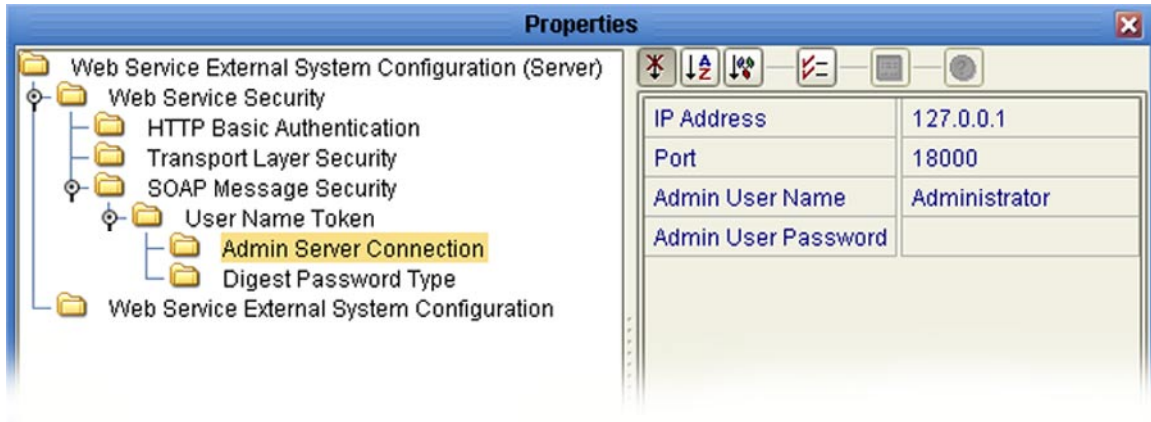
**Table 161** User Name Token Configuration Properties

Property	Description
Accept Digest Password Type	Specifies whether or not you want web service clients to be able to include a digest password. The default value is <b>true</b> .
Accept Text Password Type	Specifies whether or not you want web service clients to be able to include a text password. The default value is <b>true</b> .
Token Required	Specifies whether or not the security header must contain a user name token. A value of <b>true</b> indicates that if the security header does not contain a user name token, then a SOAP header fault is thrown. The default value is <b>false</b> .
Enable Authorization	Specifies whether or not you want messages to be authorized based on the settings in the Web Services Access Manager. The default value is <b>false</b> .

## Admin Server Connection

If SOAP Message Security is enabled, the following properties must be set correctly in order to acquire a connection for authentication and authorization of the User Name Token. (If no values are entered, then attempting to build a Project with the external system results in a compilation error.)

**Figure 482** Admin Server Connection Configuration Dialog



**Table 162** Admin Server Connection Configuration Properties

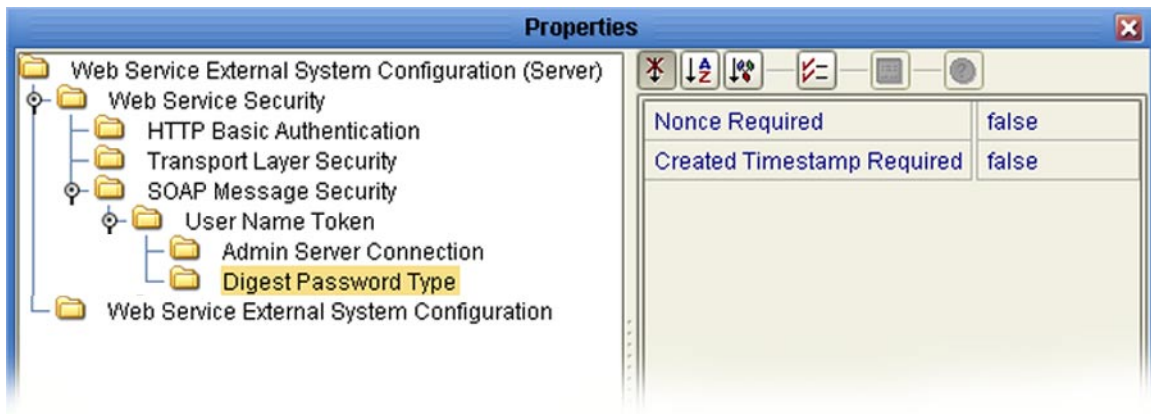
Property	Description
IP Address	Specifies the IP address or hostname for your domain. The default is <b>127.0.0.1</b> .
Port	Specifies the admin port for the integration server. The default is <b>18000</b> .
Admin User Name	Specifies the user name of the desired administrator. The default is <b>Administrator</b> .
Admin User Password	Specifies the password of the desired administrator. The default is <b>STC</b> .

## Digest Password Type

To guard against replay attacks, the security header can also contain a *nonce* and a *creation timestamp*. Using a nonce and creation timestamp in a digest password helps you to prevent *replay attacks*. In a replay attack, a malicious user eavesdrops on the communications between a sender and a receiver. The malicious user learns the sender's password (encrypted or unencrypted), and then impersonates the sender using the password.

**Important:** If you include a digest password, then you must set the default realm of the Integration Server to **wssfile**. For instructions, see "Configuring the Sun SeeBeyond Integration Server" in the Sun SeeBeyond eGate™ Integrator System Administration Guide.

**Figure 483** Digest Password Type Configuration Dialog



**Table 163** Digest Password Type Configuration Properties

Property	Description
Nonce Required	Specifies whether or not you want to require that web service clients include a nonce in a digest password. The default value is <b>false</b> .
Created Timestamp Required	Specifies whether or not you want to require that web service clients include a creation timestamp in a digest password. The default value is <b>false</b> .

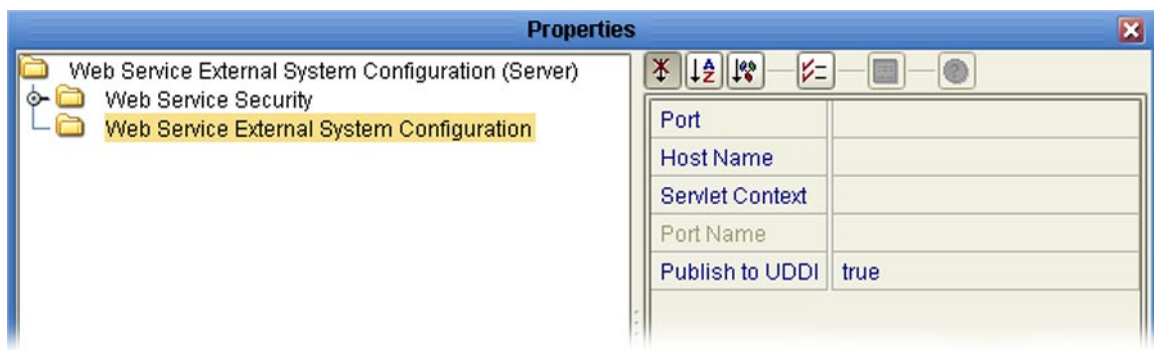
## Web Service External System Configuration

If you do not specify the properties for the web service, then they are assigned automatically upon deployment. The name automatically assigned to the **servlet context** property can be lengthy and complex, however, so you should rename it to something relatively simple. This property controls the name of the Web Service Application (.war) file.

Additionally, if you are creating more than one Web Services Application for deployment in any given Environment, you must ensure that the respective servlet context properties have different values so that each application will have a different URL. The URL format is **http://hostname:port/servlet context**.

**Note:** You must specify both the **Host Name** and **Port** if the Project is not deployed on the local system.

**Figure 484** Web Service External System Configuration Dialog



**Table 164** Web Service External System Configuration Properties

Property	Description
Port	Port on which the Web Services Application being deployed to this Web Service External System can be accessed. There is no default (see <b>Note</b> following table).
Host Name	Name or IP address of the computer hosting the Web Services Application. There is no default. <b>Note:</b> If you are using the HTTPS protocol, the value of the Host Name property must exactly match that used to create the certificates.
Servlet Context	The path and name of the Web Services Application. <b>Note:</b> For server mode, all slash characters (/) within the servlet context string are converted to underscores (_) except when they occur at the beginning of the string, where they are ignored.
Port Name	Specifies the port name used in the WSDL.
Publish to UDDI	Specifies whether or not to publish the Web Services Application to the UDDI Registry.

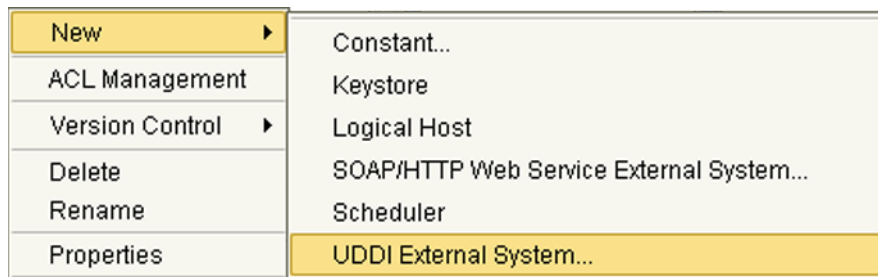
**Note:** If your domain is running on the default port (18000), then the HTTP listener port is 18001.

## 13.12 UDDI External Systems

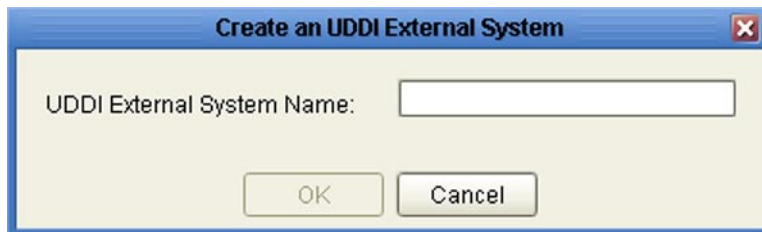
A UDDI External System represents a UDDI Registry server in a run-time Environment. You create this component by selecting the **UDDI External System** option from the *Environment* context menu in Environment Explorer (see Figure 485), which displays the dialog shown in Figure 486.

**Note:** You can create only one UDDI External System per run-time Environment.

**Figure 485** Environment Context Menu - New UDDI External System



**Figure 486** Create UDDI External System Dialog

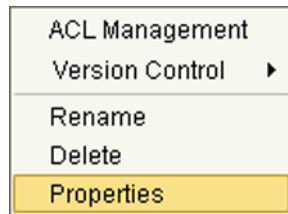


Entering a name and clicking **OK** adds the UDDI External System to the selected Environment. Right-clicking the UDDI External System in Environment Explorer displays its context menu (see [Using Environment Component Context Menus](#) on page 511).

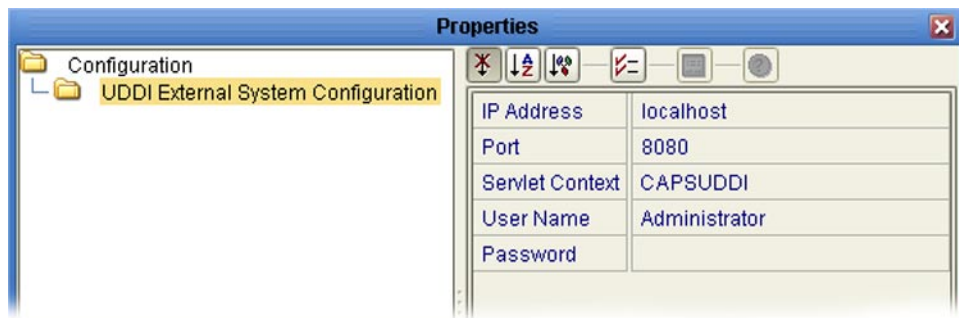
### 13.12.1 Configuring the UDDI External System

Selecting **Properties** from the **UDDI External System** context menu (see Figure 487) displays the dialog shown in Figure 488. You should assign the appropriate values for the listed properties.

**Figure 487** UDDI External System Context Menu - Properties



**Figure 488** UDDI System Configuration Properties Dialog



**Table 165** UDDI External System Configuration Properties

Property	Description
IP Address	The IP address of the UDDI External System. The default is localhost.
Port	Port on which the UDDI Registry can be accessed. The default is 8080.
Servlet Context	The URL for WSDL publication. The default is CAPSUDDI.
User Name	The user name for access to the UDDI Registry.
Password	The user password for access to the UDDI Registry.

**Important:** User names and passwords should contain only alphanumeric characters (letters and numbers), dashes, and underscores.

# Troubleshooting

This appendix contains descriptions of potential solutions for problems you may encounter when using eGate Integrator.

## What's in This Appendix

- [Error Messages](#) on page 540
- [Unexpected Behavior](#) on page 542

---

## A.1 Error Messages

### A.1.1 OutOfMemory

Out-of-memory errors occurring during Project design are most frequently overcome by increasing the *heap size* of the editor you are using, or of Enterprise Designer itself (see [Options Setup](#) on page 53). Try increasing the heap size in increments of 50 MB, starting with the appropriate editor.

If you are using an XSD OTD having a file size in excess of 1 MB, and experience out-of-memory errors that cannot be overcome by increasing the heap size, you may need to increase the *permanent generation* memory size. To do so, you must edit the following file:

```
<Sun_JavaCAPS_install_dir>\edesigner\bin\runed.bat
```

The required modification is to add the following command-line parameters after the command **runidew** (near the end of the file):

```
-J-XX:PermSize=192m -J-XX:MaxNewSize=128m -J-XX:MaxPermSize=192m
```



---

## A.2 CodeGen Validation Errors

### A.2.1 Operations ... cannot be exposed to external web services

If you are building a Web Service Definition that will be exposed externally as a web service through a SOAP/HTTP interaction, you must base all Operations containing input/output messages on XSD Objects created in the XML Schema Editor. Operations based on OTDs will not support SOAP/HTTP. See [Building a Web Service Definition](#) on page 472 for additional information.

### A.2.2 The XSD OTD ... is missing target namespace

All XSD OTDs that are used in SOAP-callable Java-based Collaboration Definitions must be derived from XML Schema Definitions that have explicitly declared target namespaces. If you receive this validation error, you should relaunch the OTD using an XML Schema Definition that has a declared target namespace. See [To redefine an existing XSD OTD](#) on page 190 for additional information.

### A.2.3 The WSDL ... is out of date ...

The WSDL in a SOAP-callable Java-based Collaboration Definition is generated when the JCD is created and persisted. If the XSD OTDs used in the JCD are changed following the creation of the JCD, the WSDL will be out of date and the application file will not be built. When this situation occurs, you must refresh the WSDL by opening the property dialog for the JCD and clicking the **OK** button (without changing anything in the dialog). See [Editing Collaboration Definition Properties](#) on page 324 for additional information.

---

## A.3 Unexpected Behavior

### A.3.1 Node Names Change when Relaunching XSD OTDs

If the names of existing XSD OTD nodes change when relaunching the OTD, you need to change the algorithm used for the Relaunch feature. This is particularly true if your XSD contains choice, sequence, any, or all node types. To enable the alternate algorithm, you must modify the following file:

```
<Sun_JavaCAPS_install_dir>\edesigner\bin\runed.bat
```

Simply locate the parameter

```
-J-DXsdOtdRelaunch.enabled
```

(it is near the end of the file), and set it to true (it is false by default). Using this functionality, you can delete, insert, or move an OTD node anywhere in the OTD structure, as long as you do not break the parent-child relationship of existing OTD nodes.

For additional information, see [To redefine an existing XSD OTD](#) on page 190.

# Glossary

**BPEL**

BPEL (Business Process Execution Language), also known as BPEL4WS (Business Process Execution Language for Web Services), is an XML-based language designed to enable task sharing for either a distributed or grid computing environment. It combines and replaces [WSDL](#) and Microsoft's XLANG specification.

**Collaboration**

A logical operation performed between some combination of message destinations and external applications. The operation is defined by a Collaboration Definition (see next entry), which can be encoded in either Java or [XSLT](#).

**Collaboration Definition**

The encoding of business rules, in Java or XSLT format. Typically, the encoding consists of operations on an [Object Type Definition \(OTD\)](#). Several Collaborations can have the same Collaboration Definition.

**Connection**

Consists of the configuration information that enables an eWay to connect to an external system.

**Connectivity Map**

Contains business logic and routing information about the data transmission. A Connectivity Map usually includes one or more [Collaborations](#), [Topics](#), [Queues](#), and [eWays](#). A Connectivity Map is created under a [Project](#). A Project may have multiple Connectivity Maps.

**Constant**

A static name-value pair that is visible across a [Project](#).

**Data Cleansing**

Data must be “cleansed” of errors in structure and content for accurate and effective use in a database or data management systems.

**Data Mapping**

Refers to establishing the relationship and data flow pattern between source and target objects, usually within the context of relational database management systems (RDBMSs).

**Data Normalization**

The process of removing redundant or incorrect data structure and organization, thereby creating a maintainable data set that can be cross-referenced. Normalized data is both easier to analyze and easier to expand.

**Data Transformation**

The process of converting the data extracted from its source into the form required by its target program or system. This process includes [Data Cleansing](#), [Data Mapping](#), [Data Normalization](#), and other sub-processes.

**DBCS**

Acronym for Double-Byte Character Set.

**Deployment Profile**

Contains the information about how the [Project](#) components will be deployed in an Environment. A Project can have multiple Deployment Profiles, but only one Deployment Profile can be activated for a Project in any one [Environment](#).

**Derived Collaboration**

A Collaboration that inherits operations from another, according to standard object-oriented practice.

**Domain**

A domain contains the eGate Integrator run-time components, including integration servers and message servers, that are installed on a host hardware platform. It is an instance of a [Logical Host](#).

**DTD**

A Document Type Definition (DTD) specifies how an associated document, written either in the Standard Generalized Markup Language (SGML) or of the Extensible Markup Language ([XML](#)), is to be processed.

**Enterprise Designer**

The [Project](#) design tool within eGate Integrator.

**Enterprise Service Bus (ESB)**

A category of software, incorporating native web services support, that provides a low-end alternative to a comprehensive integration broker suite — offering limited functionality, but less complexity and lower cost.

**Environment**

A collection of physical resources and their configurations that are used to host eGate Integrator [Project](#) components. An Environment contains Logical Hosts and external systems.

**eWay**

A link between a [Collaboration](#) and an external connection including the message server connection (topic or queue) or external application.

**External Application**

A logical representation of an application external to the Java™ Composite Application Platform Suite.

**External System**

A representation of a computer system hosting an application external to the Java™ Composite Application Platform Suite.

**HTML**

HTML (HyperText Markup Language) is the set of markup symbols or codes (tags) inserted in a file intended for display on a web page. HTML describes the content of the web page (primarily text and graphics) only in terms of how it is to be displayed and interacted with.

**HTTP**

HTTP (HyperText Transfer Protocol) is the set of rules for transferring files — text, graphics, audio, video — on the World Wide Web.

**Impact Analyzer**

A module within Enterprise Designer that analyzes and predicts the impact a specified change would have on other components in the [Project](#).

**Integration Server**

J2EE-compatible software platform that houses the business logic container used to run Collaborations and JCA connectors (eWays). Provides transaction services, persistence, and external connectivity.

**JMS IQ Manager**

JMS-compliant, guaranteed delivery store, forwarding, and queueing service.

**Link**

The JMS Connection between a [Collaboration](#) and a topic or queue in a JMS-compliant message server.

**Linked Message Destination**

A reference to a [Message Destination](#) defined in another [Connectivity Map](#).

**Logical Host**

A Logical Host is a template for a [Domain](#), which contains the eGate Integrator run-time components.

**Message Destination**

A general term for a topic or queue. Two or more Projects can share a message destination that has the same name and is deployed on the same message server. A single Project may also have a single message destination referenced in multiple [Connectivity Maps](#).

**Metadata**

Metadata describes the structure and format of a particular set of data.

**Object Type Definition (OTD)**

Object Type Definitions contain the data structure and rules that define an object. OTDs are used in [Collaboration Definitions](#) for creating data transformations and interfacing with external systems.

**Project**

Contains a collection of logical components, configurations, and files that are used to solve business problems. A Project organizes the files and packages and maintains the settings that comprise an eGate Integrator system in [Enterprise Designer](#).

**Queue**

A JMS queue is a shareable object that conforms to the *point-to-point* (p2p, or PTP) messaging domain, where one sender delivers a message to exactly one receiver. When the Sun SeeBeyond [JMS IQ Manager](#) sends a message to a queue, it ensures it is received once and only once, although there may be many receivers “listening” to the queue. This is equivalent to the subscriber pooling in other queue implementations. You can reference a queue that exists in another [Connectivity Map](#) or [Project](#).

**Relational Database (RDBMS)**

Short for Relational Database Management System, most often referred to as RDBMS, in which data is stored in related tables and can be viewed in many different ways. Relational databases differ from flat-file databases, in which each database is self-contained as a single file or table.

**Repository**

Stores and manages the setup, component, and configuration information for eGate Integrator [Projects](#). The Repository also provides monitoring services for Projects, which include version control and impact analysis.

**SBCS**

Acronym for Single-Byte Character Set.

**Schema Runtime Environment (SRE)**

An add-on feature in eGate Integrator 5.0 that allows [Collaborations](#) developed in e\*Gate 4.x to be used in and controlled from eGate Integrator 5.0, thereby providing an interim upgrade path for e\*Gate 4.x users.

**Service**

Contains the information about executing a set of business rules. These business rules can be defined in a Java or XSLT [Collaboration Definition](#), Business Process, eTL Definition, or other service. A Service also contains binding information for connecting to JMS [Topics](#), [Queues](#), [eWays](#), and other services.

**SOAP**

SOAP (Simple Object Access Protocol) enables a program running in one operating system to communicate with another program running in either the same or a different operating system, using [HTTP](#) and [XML](#) as the mechanisms for information exchange.

**Subproject**

An independent **Project** that is included as part of another Project, and is displayed in the Enterprise Explorer tree as a “branch” beneath the main Project.

**Table**

Refers to data arranged in rows and columns, as in a spreadsheet. In **Relational Database (RDBMS)** systems, all information is stored in tables.

**Topic**

A JMS topic is a shareable object that conforms to the *publish-and-subscribe* (pub/sub) messaging domain, where one publisher broadcasts messages to one or more subscribers. When the Sun SeeBeyond **JMS IQ Manager** publishes a message on a topic, it ensures that all subscribers receive the message.

**Transformation**

See **Data Transformation**.

**UDDI**

UDDI (Universal Description, Discovery, and Integration) is an **WSDL**-based registry that enables businesses to list themselves and their services on the Internet.

**Version Control**

Features that maintain the integrity of a program or **Project** by controlling the ability of an individual to modify the program or Project, and providing an audit trail for accepted modifications.

**WSDL**

WSDL (Web Services Flow Language) is an **XML**-based language, derived from **SOAP**, used to describe the services a business offers via the Internet. WSDL provides the means of expressing business services in the **UDDI** registry.

**XML**

XML (Extensible Markup Language) is a superset of **HTML**, which also describes the content in terms of what data is being described. XML is *extensible* because — unlike HTML — the markup symbols are unlimited and self-defining.

**XSD**

XSD (XML Schema Definition) specifies how to formally describe the elements in an **XML** document. It is more powerful than, and generally replaces, the older Document Type Definition (DTD). eGate Integrator makes use of **Object Type Definition (OTD)**s described in XSD, as well as DTD.

**XSLT**

XSLT (Extensible Stylesheet Language Transformation) is a language for transforming XML documents into other XML documents. It is designed for use as part of XSL, which is a stylesheet language for XML. eGate Integrator makes use of **Collaboration Definitions** coded in XSLT.

# Index

## A

Access Collaboration method 263  
 ACL properties 106, 109, 202, 244, 379, 511  
 Add and Assign Collaboration method 265  
 Add Collaboration method 272, 279  
 Add Element as Child command 161  
 Add Field command 161  
 Add Instance command 166  
 adding instances  
     classes 285  
     methods 289  
 addition Collaboration method 400  
 Advanced Mode command 249  
 ancestor node 148  
 AND and Assign Collaboration method 266  
 AND Collaboration method 268, 400  
 antecoding property 215  
 application file  
     command-line generation 140  
 Assign Collaboration method 264  
 Auto Layout command 257  
 automap feature 135

## B

boolean Collaboration method 393  
 BPEL  
     definition 543  
 branch, repository 76  
 Break command 253  
 business rules  
     tree 253  
 Business Rules on Left command 249  
 Business Rules on Top command 249  
 byte-array encoding 361–375

## C

Call Java Method command 257  
 Cast Collaboration method 275  
 Catch command 253  
 ceiling Collaboration method 398  
 CharAt Collaboration method 279  
 check in 80

    without revisions 89  
 check out 82, 87  
 child node 148  
 Choice node 149  
 Class command 251  
 classes  
     adding instances 285  
 Collaboration  
     definition 543  
     derived 544  
 Collaboration Definition  
     definition 543  
 Collaboration definition  
     Java 315  
     XSLT 377  
 Collaboration method box  
     Java 261  
     XSLT 392  
 Collaboration method palette  
     Java 260  
     XSLT 392  
 Collapse All Methods command 257  
 Collapse All Rules command 251  
 commandline codegen 140  
 Comment command 252  
 Commit Changes command 254  
 Commit Code Changes command 389  
 Complement operator 277  
 component version  
     checked in 78  
     checked out 78  
     history 84  
     latest version 90  
     retrieved 78  
 components  
     copying and pasting 97, 110  
     cutting and pasting 98  
 Concat Collaboration method 280  
 concat Collaboration method 403  
 connection 543  
 Connectivity Map  
     definition 543  
     Editor 126  
 constants 543  
     Environmental 512  
 Constructor command 252  
 contains Collaboration method 393  
 Continue command 253  
 conventions, document 34  
 convert Collaboration method 372–373  
 copying and pasting components 97, 110  
 copying and pasting Projects 97  
 count Collaboration method 395  
 Create Literal command 288



current Collaboration method 395  
 customizer 52  
 cutting and pasting components 98  
 cutting and pasting Projects 98

## D

data  
   antecoding 215  
   cleansing 543  
   decoding 215, 219  
   encoding 176, 183, 215, 219  
   mapping 543  
   normalization 544  
   postcoding 215  
   transformation 544  
 Data Encoding 151  
 data encoding 361–375  
 Data Type Definition (DTD) 174  
 database, relational 546  
 debugger  
   Collaboration (Java) 303  
 decoding property 215, 219  
 Decrement method 273  
 Delete Element or Field command 161  
 Delete Instance command 166  
 deployment  
   automapping 135  
 Deployment Editor 130  
 Deployment Profile 130  
   creating 132  
   definition 544  
   map variables 136  
   snapshot 138  
 derived Collaboration 544  
 descendant node 148  
 diff file  
   create 94  
   generate 94  
   merge 95  
 digest password 412  
 Divide and Assign Collaboration method 266  
 Divide Collaboration method 273  
 division Collaboration method 400  
 document Collaboration method 396  
 document conventions 34  
 Domain  
   definition 544  
 Do-While command 252  
 DTD  
   definition 544  
   Wizard 174

## E

EAR file  
   generation 140  
 Editor  
   Connectivity Map 126  
   Deployment 130  
   Java Collaboration 246  
   XSLT Collaboration 387  
 editor  
   OTD 159  
 element-available Collaboration method 394  
 encoding  
   byte-array 361–375  
 Encoding Converter method 362  
 encoding property 215, 219  
 EndsWith Collaboration method 280  
 Enterprise Designer  
   definition 544  
   font size 57  
   menu bar 46  
   starting 45  
 Enterprise Explorer  
   Environments 507  
   Projects 103  
 Enterprise Manager 42  
 Enterprise Service Bus (ESB)  
   definition 544  
 Environment 40  
   constants 512  
   definition 544  
 Environment Explorer 507  
 Equal Collaboration method 270  
 equal Collaboration method 401  
 Equals Collaboration method 275, 280  
 error messages  
   OutOfMemory 540  
 eWay  
   definition 544  
 Expand All Methods command 257  
 Expand All Rules command 251  
 Export Java Rule command 248  
 external application 117  
   definition 545  
 external system 40  
   definition 545

## F

False Collaboration method 268  
 false Collaboration method 394  
 Field command 252  
 Find command 251, 255  
 Find Next command 255

Find Previous command 255  
 floor Collaboration method 398  
 font size (Enterprise Designer)  
   changing 57  
 For command 252  
 format\_number Collaboration method 403  
 function\_available Collaboration method 394

## G

generate-id Collaboration method 396  
 Go command 167  
 Greater or equal Collaboration method 270  
 Greater than Collaboration method 270  
 greater\_or\_equal Collaboration method 401  
 greater\_than Collaboration method 401  
 Group node 149

## H

HEAD branch 76  
 heap size 53  
 HTTP  
   definition 545  
 HTTP basic authentication  
   described 411

## I

Icons  
   JCE Business Rules Designer toolbar 257  
   JCE Business Rules toolbar 251, 254  
   JCE Class Browser toolbar 283  
   JCE toolbar 248  
   OTD Editor toolbar 160  
   Web Service Definition Editor toolbar 469  
   XCE toolbar 389–390  
   XML Schema Editor toolbar 421  
 id Collaboration method 396  
 If-Then command 252  
 Impact Analyzer  
   definition 545  
   usage 99  
 Import a Local File command 248, 389–390  
 Import JAR File command 250  
 Import OTD command 160  
 Increment method 273  
 IndexOf Collaboration method 280  
 Instanceof Collaboration method 275  
 Integration Server  
   definition 545  
 Intersection operator 269, 277

## J

Java 2 Enterprise Edition (J2EE) 42  
 Java Collaboration  
   Debugger 303  
   definitions 315  
   Editor (JCE) 246  
   method box 261  
   method palette 260  
   Wizard 316  
 Java Collaboration methods  
   Access 263  
   Add 272, 279  
   Add and Assign 265  
   AND 268  
   AND and Assign 266  
   Assign 264  
   Cast 275  
   CharAt 279  
   Concat 280  
   convert 372–373  
   Decrement 273  
   Divide 273  
   Divide and Assign 266  
   EndsWith 280  
   Equal 270  
   Equals 275, 280  
   False 268  
   Greater or equal 270  
   Greater than 270  
   Increment 273  
   IndexOf 280  
   Instanceof 275  
   Length 264, 280  
   Less or equal 270  
   Less than 271  
   Literal Character 280  
   Literal Number 273  
   Literal String 280  
   marshal 152  
   marshalToBytes 154, 374  
   marshalToString 153  
   Multiply 272  
   Multiply and Assign 265  
   Negate 273  
   NOT 268  
   Not equal 271  
   Null 276  
   OR 268  
   OR and Assign 267  
   Positive 273  
   Predecrement 274  
   Preincrement 273  
   Remainder 273

- Remainder and Assign 266
  - Replace 281
  - reset 174, 181, 191
  - Shift Left and Assign 266
  - Shift Right and Assign 266
  - Substring 281
  - Subtract 272
  - Subtract and Assign 265
  - ToLowerCase 281
  - ToString 276
  - ToUpperCase 281
  - Trim 281
  - True 268
  - Unassigned Shift Right and Assign 266
  - unmarshal 155
  - unmarshalFromBytes 157, 365
  - unmarshalFromString 156
  - writeBytes 375
  - XOR and Assign 267
  - Java Collaboration operators
    - Complement 277
    - Intersection 269, 277
    - Shift Left 278
    - Shift Right 278
    - Symmetric Difference 269, 277
    - Union 269, 277
    - Unsigned Shift Right 278
  - JCE commands
    - Advanced Mode 249
    - Auto Layout 257
    - Break 253
    - Business Rules on Left 249
    - Business Rules on Top 249
    - Call Java Method 257
    - Catch 253
    - Class 251
    - Collapse All Methods 257
    - Collapse All Rules 251
    - Comment 252
    - Commit Changes 254
    - Constructor 252
    - Continue 253
    - Create Literal 288
    - Do-While 252
    - Expand All Methods 257
    - Expand All Rules 251
    - Export Java Rule 248
    - Field 252
    - Find 251, 255
    - Find Next 255
    - Find Previous 255
    - For 252
    - If-Then 252
    - Import a Local File 248, 389–390
    - Import JAR File 250
    - Local Variable 252
    - Method 251
    - New Array 257
    - Redo 255
    - Refresh Collaboration 250
    - Replace 255
    - Return 252, 288
    - Roll Back Changes 254
    - Rule 252
    - Source Code Mode 249
    - Standard Mode 249
    - Switch 252
    - Test Mode 249
    - Throw 252
    - Try 252
    - Undo 255
    - Validate 249
    - While 252
  - JMS IQ Manager
    - definition 545
- ## K
- key Collaboration method 396
- ## L
- language Collaboration method 394
  - last Collaboration method 396
  - Leaf nodes 149
  - Length Collaboration method 264, 280
  - Less or equal Collaboration method 270
  - Less than Collaboration method 271
  - lesser\_or\_equal Collaboration method 401
  - lesser\_than Collaboration method 401
  - link 545
  - Literal Character Collaboration method 280
  - Literal Number Collaboration method 273
  - Literal String Collaboration method 280
  - Local Variable command 252
  - local-name Collaboration method 397
  - Logical Host 40
    - definition 545
- ## M
- map variables 136
  - marshal Collaboration method 152
  - Marshal command 166
  - Marshal To Bytes command 166
  - Marshal To String command 166
  - marshalToBytes Collaboration method 154, 374

- marshalToString Collaboration method 153
- menu bar 46
- message destination
  - definition 545
  - linked 545
- metadata 545
- method box
  - Java Collaboration 261
  - XSLT Collaboration 392
- Method command 251
- method palette
  - Java Collaboration 260
  - XSLT Collaboration 392
- methods
  - adding instances 260, 289, 391
- multiplication Collaboration method 401
- Multiply and Assign Collaboration method 265
- Multiply Collaboration method 272

## N

- name Collaboration method 397
- namespace-uri Collaboration method 397
- Negate operator 273
- negative Collaboration method 402
- New Array command 257
- node
  - ancestor 148
  - child 148
  - choice 149
  - descendant 148
  - group 149
  - leaf 149
  - non-leaf 149
  - parent 148
  - repeating 256
  - root 148
  - sibling 148
- nonce 529
- non-leaf node 149
- normalize\_space Collaboration method 403
- NOT Collaboration method 268, 394
- Not equal Collaboration method 271
- not\_equal Collaboration method 402
- Null Collaboration method 276
- number Collaboration method 398
- number-literal Collaboration method 398

## O

- Object Type Definition 242, 377, 546
  - wizard 175, 182
- Object Type Definition (OTD) 146
- Open File command 166

- OR and Assign Collaboration method 267
- OR Collaboration method 268, 402
- OTD 546
  - editor 159
  - properties 214
  - tester 162
- OTD Editor commands
  - Add Element as Child 161
  - Add Field 161
  - Delete Element or Field 161
  - Go 167
  - Import OTD 160
  - Open File 166
  - Redo 161
  - Save as New Name 160
  - Save File 166
  - Show as hex 166
  - Tester 160
  - Toggle Reference Tab Panel 161
  - Undo 161
  - Word wrap 167
- OTD Tester commands
  - Add Instance 166
  - Delete Instance 166
  - Marshal 166
  - Marshal To Bytes 166
  - Marshal To String 166
  - Reset 166
  - Unmarshal 167
  - Unmarshal From Bytes 167
  - Unmarshal From String 167
- OTD Wizard
  - DTD 174
  - User-Defined 191, 193
  - XSD 181
- OutOfMemory error 540

## P

- palette
  - Java Collaboration methods 260
  - XSLT Collaboration methods 392
- parent node 148
- password equivalent 412, 528
- passwords
  - WS-Security 412
- position Collaboration method 397
- Positive method 273
- postcoding property 215
- Predecrement method 274
- Preincrement method 273
- Profile, Deployment 130
- Project
  - copying and pasting 97

- cutting and pasting 98
- definition 546
- Project Explorer 103
- properties
  - OTD 214

## Q

- queue
  - definition 546

## R

- rdbms 546
- Redo command 161, 255
- Refresh Collaboration command 250
- relational database 546
- Relaunch 180, 190
- Remainder and Assign Collaboration method 266
- Remainder Collaboration method 273
- remainder Collaboration method 402
- repeating nodes
  - group 149
  - mapping between 256
- Replace Collaboration method 281
- Replace command 255
- Repository 39, 59
  - definition 546
- repository
  - branches 76
- reset Collaboration method 174, 181, 191
- Reset command 166
- Return command 252, 288
- Roll Back Changes command 254
- Roll Back Code Changes command 389
- root node 148
- round Collaboration method 399
- Rule command 252

## S

- Save as New Name command 160
- Save File command 166
- Save XSL to a Local File command 389
- Scheduler 119
- Schema Runtime Environment (SRE) 42
  - definition 546
- Screen Captures 34
- security 518
- service
  - definition 546
- servlet context property 537
- Shift Left and Assign Collaboration method 266

- Shift Left operator 278
- Shift Right and Assign Collaboration method 266
- Shift Right operator 278
- Show as hex command 166
- Show Mapping Only command 389
- Show Maps and Code command 389
- Show XSLT Code Only command 389
- Sibling node 148
- snapshot 138
- SOAP
  - definition 546
- Source Code Mode command 249
- Standard Mode command 249
- starting
  - Enterprise Designer 45
- starts\_with Collaboration method 394
- string Collaboration method 404
- string\_length Collaboration method 404
- string\_literal Collaboration method 404
- subproject
  - definition 547
- Substring Collaboration method 281
- substring Collaboration method 404
- substring-after Collaboration method 404
- substring-before Collaboration method 404
- Subtract and Assign Collaboration method 265
- Subtract Collaboration method 272
- subtraction Collaboration method 402
- sum Collaboration method 399
- supporting documents 35
- Switch command 252
- Symmetric Difference operator 269, 277
- system-property Collaboration method 405

## T

- table 547
- tagging components 92
- Test Mode command 249
- tester
  - Collaboration (Java) 303
  - OTD 162
  - XSLT 387
- Tester command 160
- Throw command 252
- Toggle Reference Tab Panel command 161
- token
  - defined 411
- ToLowerCase Collaboration method 281
- topic
  - definition 547
- ToString Collaboration method 276
- ToUpperCase Collaboration method 281
- transformation 547

translate Collaboration method 405  
 Transport Layer Security  
   described 411  
 Trim Collaboration method 281  
 True Collaboration method 268  
 true Collaboration method 394  
 Try command 252

## U

UDDI  
   definition 547  
   registry 417  
 Unassigned Shift Right and Assign Collaboration  
 method 266  
 undo check-out 89  
 Undo command 161, 255  
 Union operator 269, 277  
 unmarshal Collaboration method 155  
 Unmarshal command 167  
 Unmarshal From Bytes command 167  
 Unmarshal From String command 167  
 unmarshalFromBytes Collaboration method 157,  
 365  
 unmarshalFromString Collaboration method 156  
 unparsed-entity-uri Collaboration method 405  
 Unsigned Shift Right operator 278  
 User-Defined OTD Wizard 191, 193

## V

Validate command 249  
 variables  
   mapping 136  
 version control 78  
   branches 76  
   check in 78, 80  
   check out 78, 82, 87  
   command-line utilities 96  
   create diff 94  
   definition 547  
   generate diff 94  
   history 84  
   latest version 90  
   merge diff 95  
   retrieve 78  
   tagging 92  
   undo check out 89

## W

While command 252  
 Wizard

Java Collaboration 316  
 OTD (DTD) 174  
 OTD (User-Defined) 191, 193  
 OTD (XSD) 181  
 XSLT Collaboration 381  
 Word wrap command 167  
 writeBytes Collaboration method 375  
 WSDL  
   definition 547  
 WS-Security  
   described 411

## X

XCE commands  
   Commit Code Changes 389  
   Roll Back Code Changes 389  
   Save XSL to a Local File 389  
   Show Mapping Only 389  
   Show Maps and Code 389  
   Show XSLT Code Only 389  
 XML  
   definition 547  
 XOR and Assign Collaboration method 267  
 XSD  
   definition 547  
   Wizard 181  
 XSLT  
   definition 547  
 XSLT Code Tester 387  
 XSLT Collaboration  
   definitions 377  
   Editor 387  
   method box 392  
   method palette 392  
   Wizard 381  
 XSLT Collaboration methods  
   addition 400  
   AND 400  
   boolean 393  
   ceiling 398  
   concat 403  
   contains 393  
   count 395  
   current 395  
   division 400  
   document 396  
   element-available 394  
   equal 401  
   false 394  
   floor 398  
   format\_number 403  
   function\_available 394  
   generate-id 396

- greater\_or\_equal 401
- greater\_than 401
- id 396
- key 396
- language 394
- last 396
- lesser\_or\_equal 401
- lesser\_than 401
- local-name 397
- multiplication 401
- name 397
- namespace-uri 397
- negative 402
- normalize\_space 403
- NOT 394
- not\_equal 402
- number 398
- number-literal 398
- OR 402
- position 397
- remainder 402
- round 399
- starts\_with 394
- string 404
- string\_length 404
- string\_literal 404
- substring 404
- substring-after 404
- substring-before 404
- subtraction 402
- sum 399
- system-property 405
- translate 405
- true 394
- unparsed-entity-uri 405