SUN SEEBEYOND

eGATE™ INTEGRATOR USER'S GUIDE

Release 5.1.2

Sun microsystems

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 intellectuels relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuels 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 règlementation 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.

Part Number: 819-7393-10

Version 20061011181514

Contents

Figures 15

Tables 29

Chapter 1

Introduction 34

About eGate Integrator 34

What’s New in This Release 34

About This User’s Guide 35

What’s in This User’s Guide 35

Scope 36

Intended Audience 36

Text Conventions 36

Screen Captures 36

Related Documents 37

Sun Microsystems, Inc. Web Site 37

Documentation Feedback 37

Chapter 2

eGate Integrator Overview 38

Introduction 38

Integration Model 39

System Architecture 40

Repository 41

Run-Time Environments 42

Enterprise Designer 43

Enterprise Manager 44

Chapter 3

Enterprise Designer Basics 45

Overview 45
Enterprise Explorer 46
Enterprise Designer Editors 46
Additional Tools 46

Starting Enterprise Designer 47

Navigating Enterprise Designer 48
Menus 48
File Menu 48
Edit Menu 48
Tools Menu 49
View Menu 49
Window Menu 50
Help Menu 50
Toolbar 51
Wizards 52
Browser Buttons 53
Property Configuration Dialogs 54

Configuring Enterprise Designer 55
Setting Up Options 55
Options Setup 55
Language 56
Build Options 56
Preferences 57
Proxy Configuration 57
Customizing the User Interface 59
Resizing Panes 59
Component Docking 59
Changing the Default Font Size 60

Chapter 4

Global Features 61

Overview 61

The Repository 62
Using the Repository Context Menu 62
Project Explorer Version 62
Environment Explorer Version 63

Using Export/Import Features 65
Exporting a Project or Environment 65
Exporting Using Enterprise Designer 65
Exporting Using the Command Line 69
Importing a Project or Environment 70
Importing Using Enterprise Designer 71
Importing Using the Command Line 77

Using Version Control Features 79
Managing Project Versions 79
Creating Branches 79
Changing Branches 80
Managing Component Versions 81
Accessing Component Version Control Features 82
Checking a Component In 83
Checking a Component Out 85
Viewing a Component’s Version History 87
Viewing a Read-Only Version of a Component 89
Retrieving a Component to Your Workspace 90
Checking a Component In Without Revisions 92
Checking In a Previous Version as the Latest Version 93
Tagging Components 95
Merging Changes from Another Version 97
Command-line Utilities 99
Moving and Replicating Repository Objects 100
Copying and Pasting 100
Projects 100
Components 100
Cutting and Pasting 101
Projects 101
Components 101
Analyzing Component Dependencies 102
About Component Dependencies 102
Using the Impact Analyzer 104
Chapter 5
Projects 105
Overview 105
About Project Explorer 107
Project Explorer Icons 108
Project Components 109
Using the Project Context Menu 109
Using Project Component Context Menus 113
Creating a Project 115
Project Nesting 116
Services 117
Business Processes 117
Collaborations 117
Linking Services and Destinations 118
Message Destinations 119
Naming a Message Destination 120
Standard Characters 120
Special Characters 120
External Applications 121

Web Service External Applications 122

Schedulers 123
Configuring a Scheduler 123
Schedule 123
Time Zone 124
Naming a Scheduler 124
Standard Characters 124
Special Characters 124

Constants and Variables 125
Project Constants 126
Project Variables 127
Variables & Constants Object Group 128

Connectivity Maps 129
Using the Connectivity Map Editor 130
Connectivity Map Editor Toolbar 131
Connecting Components 132
Configuring JMS Clients 133
Root Properties 133
Basic Properties 134
Redelivery Handling 135
Advanced Properties 136

Deployment Profiles 138
Using the Deployment Editor 138
Creating a Deployment Profile 140
Automapping 144
Mapping Variables 145
Version Control 146
Building an Application File 148
Building an Application File From the Command Line 149
Deploying the Project 154
Deploying to a Sun Java System Application Server 154

Chapter 6

Object Type Definitions I 155

Understanding OTDs 155
OTD Fundamentals 156
Metadata and OTDs 156
Data Structure 157
Data Types 158
Name Properties 158
OTD Types 159
Externally-Defined OTDs 159
User-Defined OTDs 159
Predefined OTDs 159

Specifying Data Conversion and Encoding 160
Encoding Options 160
Marshaling and Unmarshaling Methods 161
marshal() 161

marshalToString() 162
marshalToBytes() 163
unmarshal() 164
unmarshalFromString() 165
unmarshalFromBytes() 166
Setting Delimiters 167

Using the OTD Editor 168
OTD Editor Toolbar Icons 169
OTD Tester 171
Input Panel 172
Output Panel 173
Status Panel 173
Verbose Panel 174
OTD Tester Controls 175
Marshal/Unmarshal Processes 177
Data Encoding 177
Using the OTD Tester 178

Chapter 7

Object Type Definitions II 181

Overview 181

Using the New OTD Wizard 182

Creating Externally-Defined OTDs 183
Using the DTD Wizard 183
Using the XSD Wizard 190

Creating User-Defined OTDs 200
Using the User-Defined OTD Wizard 200
Using the UD OTD from File Wizard 202

Using Predefined OTDs 205
Using the JMS OTD 206
Using Scheduler OTDs 209
Using the CBO OTD Library 211
Installing the CBO OTD Library 211

Using the OTD Context Menu 212

Chapter 8

Object Type Definitions III 214

Overview 214

Creating and Managing OTD Nodes 215
Moving Nodes 217
Cutting, Copying, and Pasting Nodes 218

Creating and Using Templates 219
External Templates 219

Internal Templates 221
OTD Node Representations 222
Element Nodes 222
Field Nodes 223
Editing OTD Properties 224
Root Node Properties 224
Element Node Properties 226
Order Property 227
Field Node Properties 228
Specifying the Node Type 230
Default Values 231
Specifying Read/Write Access 232
Matching Data Patterns 233
Defining Byte Patterns 233
Specifying Pattern Alignment 233
Specifying Delimiters 235
Delimiter List 235
Delimiter Properties 236
Delimiter Levels 237
Delimiter Type 239
Escape Option 239
Delimiter Bytes 240
Escape Sequences 240
Multiple Delimiters 241
Precedence 242
Optional Mode 243
Terminator Mode 245
Creating a Delimiter List 246

Chapter 9

Collaboration Definitions (Java) I 252
Overview 252
Using the Collaboration Definition Context Menu 254
Using the Collaboration Editor (Java) 256
Collaboration Editor Toolbar Icons 258
Business Rules Editor 261
Business Rules Editor Toolbar Icons 261
Business Rules Tree 263
Java Source Editor 264
Java Source Editor Toolbar Icons 264
Business Rules Designer 266
Mapping Between Repeating Nodes 266
Business Rules Designer Toolbar Icons 267
Node Menu 268
Method-Access Nodes 269
Collaboration Method Menus 270

Collaboration Method Boxes 271
Placing Collaboration Methods 272
Editing or Adding Literals 272

Graphical Collaboration Methods (Java) 273
Array Methods 273
Assignment 275
Boolean Methods 278
Comparison Methods 280
Math Methods 282
Object Methods 285
Operators 287
String Methods 289

Using the Class Browser 292
Class Browser Toolbar Icons 293

Using the Method Dialog 297

Auto Type Conversion 300
Character Conversion 302
String to Number Conversion 302
Character Encoding 304
Byte-to-String Conversion 304
String-to-Byte Conversion 305

Using the Collaboration Tester (Java) 306
Breakpoints 307
Code Validation 308

Using the Java Debugger 313
Starting the Java Debugger 313
Setting Breakpoints 315
Inspecting and Editing the Source Code 318
Stepping Into, Over, or Out 319
Inspecting Java Threads and Methods 320
Inspecting a Local Variable or Method 321
Saving and Resuming Debug Sessions 324

Chapter 10

Collaboration Definitions (Java) II 325

Creating a Java-based Collaboration Definition 326
New Web Service Operation 328
Existing Web Service Operation 332

Editing Collaboration Definition Properties 334

Creating Classes 337

Creating Methods 339
Creating Parameters 341

Creating Constructors 343
Creating Parameters 345

Creating Fields 346
Creating Arrays 348
Creating and Modifying Variables 350
Creating and Modifying Literals 354
Using the Toolbar Menus 354
Using the Method Dialog 354
Generating Alerts 355
Creating Log Entries 359
Importing Third-Party Java Classes 362
Adding Java classes to Collaboration Definitions 364
Copying OTD Nodes 366
copyNode() 366
Using Try-Catch 368
Using Byte-Array Converters 371
Defining Encoding Converter Methods 372
Validating Java-based Collaboration Definitions 386

Chapter 11

Collaboration Definitions (XSLT) 387
Overview 387
Using the Collaboration Definition Context Menu 388
Creating an XSLT-based Collaboration Definition 390
New Web Service Operation 392
Existing Web Service Operation 395
Using the Collaboration Editor (XSLT) 396
XSLT Toolbar Icons 398
XSLT Tester Icons 399
Collaboration Method Menus 400
Collaboration Method Palette (XSLT) 401
Collaboration Method Boxes (XSLT) 401
Graphical Collaboration Methods (XSLT) 402
Boolean Methods 402
Nodes Methods 404
Number Methods 407
Operator Methods 409
String Methods 412
Design Procedures 415
Creating a For-Each Construct 415

Chapter 12

Web Services 418

Overview 418
Terminology 419
Web Services Security 420
HTTP Basic Authentication 420
Transport Layer Security (TLS) 420
SOAP Message Security 420

Developing Web Services 422
Exposing Collaboration Definitions as Web Services 422
Creating a Web Service Project 423
Procedure 423
Publishing a Web Service Project 424
Procedure 424
Accessing Web Service Definitions 426

Creating XML Schema Definitions 427
Using the Context Menu 427

Using the XML Schema Editor 428
Overview 428
Toolbar 429
XML Schema Definition Panel 429
Properties Panel 429
XML Source Code Panel 429
XML Schema Editor Toolbar Icons 430
Previewing and Exporting XSD Nodes 433
Preview/Export Options 433
XML Schema Compilation 435
Building an XML Schema Definition 437
XML Schema Definition Root Node 438
Attribute Node 442
AttributeGroup Node 444
Element Node 446
Union Node 450
Group Node 452
ComplexType Node 453
SimpleType Node 455
SimpleType Subnode 458
ListType Node 460
Sequence Node 461
Choice Node 462
All Node 463
Any Node 464
ComplexContent Node 466
SimpleContent Node 467
Extension Node 468
Restriction Node 469
Importing Existing XML Schema Definitions 470

Creating Web Service Definitions 475
Overview 475

Using the Context Menu 475
Using the Web Service Definition Editor 476
Overview 476
Toolbar 477
Web Service Definition Panel 477
Properties Panel 477
Source Code Panel 477
Web Service Definition Editor Toolbar Icons 478
Building a Web Service Definition 481
Web Service Definition Root Node 482
Import Folder 483
Messages Folder 485
Messages 485
Message Parts 487
PortTypes Folder 489
PortTypes 489
Operations 490
Properties Folder 493
Property 493
Property Aliases Folder 495
Property Alias 496
Service Link Type Folder 497
Service Link Role 497
Types Folder 498
Importing Existing Web Service Definitions 499
Configuring WSDL Binding Properties 504
Configuring the PortType Binding 506
PortTypes 506
Configuring the Operation 507
Operations 507
Configuring Messages 508
SOAP Messages 508
Configuring the Message Body 509
Message Body 509
Configuring the Message Header 510
Message Header 510
Configuring a Header Fault 511
Header Fault 511
Configuring a Java Exception 512
Java Exception 512
Invoking Web Services From Collaboration Definitions 513
Chapter 13
Run-time Environments 516
Overview 516
Using the Environment Editor 517
Using Environment Explorer 518
Environment Explorer Icons 519

Environment Components 520
Using the Environment Context Menu 520
Using Environment Component Context Menus 522
Environmental Constants 523
Using the Constant Context Menu 524
Keystores 525
Using the Keystore Context Menu 525
Logical Hosts and Domains 526
Using the Logical Host Context Menu 526
Configuring Domains 528
Integration Servers 529
Using the Integration Server Context Menu 529
Configuring an Integration Server 530
Sun SeeBeyond Integration Server Configuration 530
eInsight Engine Configuration 531
Message Servers 532
Configuring a Message Server 533
Sun SeeBeyond JMS IQ Manager Configuration 533
Schedulers 535
SOAP/HTTP Web Service External Systems 536
Configuring a SOAP/HTTP Client 537
Web Service Security: HTTP Basic Authentication 537
Web Service Security: HTTP Proxy Server Authentication 538
Web Service Security: SOAP Message Security 539
Web Service External System Configuration 542
Configuring a SOAP/HTTP Server 543
Web Service Security: HTTP Basic Authentication 543
Web Service Security: Transport Layer Security 544
Web Service Security: SOAP Message Security 545
Web Service External System Configuration 549
UDDI External Systems 550
Configuring the UDDI External System 551

Appendix A

Troubleshooting 552
Error Messages 552
OutOfMemory 552
CodeGen Validation Errors 553
Operations … cannot be exposed to external web services 553
The XSD OTD … is missing target namespace 553
The WSDL … is out of date … 553
Unexpected Behavior 554
Node Names Change when Relaunching XSD OTDs 554
Java Method Usage 555

System.getProperties() Causes Exception 555

Glossary 556

Index 561

Figures

Figure 1	eGate Integrator	38
Figure 2	eGate Integrator Implementation Model	40
Figure 3	Distributed eGate Integrator System	41
Figure 4	Enterprise Designer	43
Figure 5	Enterprise Manager	44
Figure 6	Enterprise Designer User Interface	45
Figure 7	Login Dialog	47
Figure 8	Options Setup - Heap Size Dialog	55
Figure 9	Options Setup - Language Dialog	56
Figure 10	Options Setup - Build Options Dialog	56
Figure 11	Options Setup - Preferences Dialog	57
Figure 12	Options Setup - Proxy Configuration Dialog	58
Figure 13	Options Setup - Proxy Configuration Dialog - Error	58
Figure 14	Draggable Splitters	59
Figure 15	Component View Docking Options	60
Figure 16	Repository Context Menu (Project Explorer)	62
Figure 17	Repository Context Menu (Environment Explorer)	63
Figure 18	Export Manager Dialog (1a)	65
Figure 19	Export Manager Dialog (1b)	66
Figure 20	Export Manager Dialog (2)	67
Figure 21	Save As Dialog	67
Figure 22	Enter File Name Dialog (2)	68
Figure 23	Export Status Message Box	68
Figure 24	Repository Context Menu - Import Project Option	71
Figure 25	Project Context Menu - Import Options	71
Figure 26	Import Message Box	72
Figure 27	Import Manager Dialog (1)	72
Figure 28	Open File Dialog	73
Figure 29	Import Manager Dialog	74
Figure 30	Project Import Confirmation Dialog	75
Figure 31	Create New Version Confirmation Dialog	75
Figure 32	Import Status Message Box	76

Figure 33 Repository HEAD Branch 79
Figure 34 Repository Context Menu - Change Branch 80
Figure 35 Change Branch Dialog 80
Figure 36 Checked In Icon (OTD Example) 81
Figure 37 Checked Out Icon 81
Figure 38 Retrieved Icon 81
Figure 39 Connectivity Map Context Menu 82
Figure 40 Example Context Menu - Check In Option 83
Figure 41 Version Control - Check In Dialog 84
Figure 42 Example Context Menu - Check Out Option 85
Figure 43 Version Control - Check Out Dialog 86
Figure 44 Example Context Menu - Version History Option 87
Figure 45 Version Control - History Dialog 87
Figure 46 Version Control - History Dialog 89
Figure 47 Version Control - History Dialog 90
Figure 48 Access File Dialog 91
Figure 49 Confirm Version Replace Dialog 91
Figure 50 Example Context Menu - Undo Check Out Option 92
Figure 51 Version Control - Undo Check Out Dialog 92
Figure 52 Example Context Menu - Make Latest Option 93
Figure 53 Make Latest Dialog 93
Figure 54 Confirm Latest Version Override Dialog 94
Figure 55 Version Control Tag Dialog 95
Figure 56 Example Context Menu - Tag Option 96
Figure 57 Version Control Tag Dialog 96
Figure 58 Collaboration Definition Context Menu - Create Diff Option 97
Figure 59 Version Control - Create Diff Dialog 97
Figure 60 Collaboration Definition Context Menu - Merge Diff Option 98
Figure 61 Version Control - Merge Changes Dialog 98
Figure 62 Objects Referenced by Deployment Profile 102
Figure 63 Objects Referencing Deployment Profile 102
Figure 64 Objects Referenced by Connectivity Map 103
Figure 65 Objects Referencing Connectivity Map 103
Figure 66 Objects Referenced by Collaboration Definition 103
Figure 67 Objects Referencing Collaboration Definition 103
Figure 68 Impact Analyzer Dialog 104
Figure 69 Project Explorer 107
Figure 70 Project Context Menu 109

Figure 71 Project Component Context Menu 113
Figure 72 Repository Context Menu - New Project 115
Figure 73 Project Context Menu - New Connectivity Map 115
Figure 74 Project Nesting 116
Figure 75 Defining the Service Component 117
Figure 76 Linking Multiple Destinations 118
Figure 77 Topic Icon 119
Figure 78 Queue Icon 119
Figure 79 Project Context Menu - New Queue 119
Figure 80 External Application Drop-Down Menu 121
Figure 81 Web Service External Application Icon 122
Figure 82 Project Context Menu - New Web Services Application 122
Figure 83 Scheduler Icon 123
Figure 84 Scheduler Properties Dialog - Schedule 123
Figure 85 Scheduler Properties Dialog - Time Zone 124
Figure 86 Project Context Menu - New Variable or Constant 125
Figure 87 Creating a Project Constant 126
Figure 88 Creating a Project Variable 127
Figure 89 Variables and Constants Object Group 128
Figure 90 Project Context Menu - New Connectivity Map 129
Figure 91 Connectivity Map Editor 130
Figure 92 Connection Icons in a Connectivity Map - Not Configured 132
Figure 93 Connection Icons in a Connectivity Map - Configured 133
Figure 94 JMS Client Configuration Properties Dialog (Subscriber) 133
Figure 95 JMS Client Basic Configuration Properties Dialog (Producer) 134
Figure 96 JMS Client Basic Configuration Properties Dialog (Consumer) 134
Figure 97 JMS Client Redelivery Handling Properties Dialog (Consumer) 135
Figure 98 JMS Client Advanced Configuration Properties Dialog (Subscriber) 136
Figure 99 eGate Integrator Implementation Model 138
Figure 100 Deployment Editor 139
Figure 101 Example Project Connectivity Map 140
Figure 102 Web Client Example Environment 141
Figure 103 Example Deployment Profile (1) 141
Figure 104 Example Deployment Profile (2) 142
Figure 105 Example Deployment Profile (3) 143
Figure 106 Automap Icon 144
Figure 107 Automap Results Dialog Box 144
Figure 108 Automap Options Dialog Box 144

Figure 109 Map Variables Button 145
Figure 110 Deployment Profile Mappings 145
Figure 111 Project Variable Value Entry 145
Figure 112 Spreadsheet View Icon 146
Figure 113 Spreadsheet View 146
Figure 114 Snapshot Dialog 147
Figure 115 Global Settings Dialog 147
Figure 116 Build Button 148
Figure 117 Deploy Button 154
Figure 118 OTD Generation 155
Figure 119 Message Processing 156
Figure 120 OTD Node Nomenclature 157
Figure 121 marshal() 161
Figure 122 marshalToString() 162
Figure 123 marshalToBytes() 163
Figure 124 unmarshal() 164
Figure 125 unmarshalFromString() 165
Figure 126 unmarshalFromBytes() 166
Figure 127 Setting Delimiters 167
Figure 128 OTD Editor 168
Figure 129 Tester Icon 171
Figure 130 OTD Tester Interface 171
Figure 131 Data Field Display 172
Figure 132 Output Panel 173
Figure 133 Status Panel 173
Figure 134 Verbose Panel 174
Figure 135 Input View Encoding Option Menu 177
Figure 136 Data Display - Status Panel 178
Figure 137 OTD Tester Node Table 179
Figure 138 Serialized Data in Output Panel 179
Figure 139 Saving Test Data 180
Figure 140 OTD Wizard Selection Dialog 182
Figure 141 OTD Wizard Selection: DTD Wizard 183
Figure 142 Select DTD File(s) Dialog 184
Figure 143 Cannot Create OTD Warning Box 184
Figure 144 Select Encoding Dialog 185
Figure 145 Select Document Elements Dialog 186
Figure 146 Select OTD Options Dialog 187

Figure 147 OTD Context Menu 189
Figure 148 OTD Wizard Selection: XSD Wizard 190
Figure 149 XSD Wizard: Select XSD File(s) 191
Figure 150 Cannot Create OTD Warning Box 191
Figure 151 Select Encoding Dialog 192
Figure 152 Select Document Elements Dialog 193
Figure 153 Select OTD Options Dialog 194
Figure 154 OTD Context Menu 199
Figure 155 OTD Wizard Selection: User-Defined OTD 200
Figure 156 Enter OTD Name 201
Figure 157 OTD Wizard Selection: UD OTD from File 203
Figure 158 Select a File Dialog 204
Figure 159 Project Explorer - Sun SeeBeyond Folder 205
Figure 160 Project Explorer - JMS OTD 206
Figure 161 JMS OTD Property Nodes 206
Figure 162 Existing Web Service: Select Operation 207
Figure 163 Existing Web Service: Select OTDs 208
Figure 164 Project Explorer - Scheduler OTD 209
Figure 165 Existing Web Service: Select OTDs 210
Figure 166 CBO OTD Library in Project Explorer 211
Figure 167 OTD Context Menu 212
Figure 168 OTD Node Context Menu 215
Figure 169 Node-moving Options 217
Figure 170 Node Context Menu - Paste Options 218
Figure 171 Import Dialog 219
Figure 172 Element Node Created for OTD 222
Figure 173 Element Node from External Template 222
Figure 174 Element Node from Internal Template 222
Figure 175 Field Node Created for OTD 223
Figure 176 Field Node from External Template 223
Figure 177 Internal Template - Field Node 223
Figure 178 Root Node Properties (delim nodeType) 224
Figure 179 Element Node Properties (delim nodeType) 226
Figure 180 Order Property Example 227
Figure 181 Field Node Properties (delim nodeType) 228
Figure 182 nodeType Property Menu (Root/Element Example) 230
Figure 183 Access Property Menu 232
Figure 184 Match Property 233

Figure 185 Align Property Menu 234
Figure 186 Delimiter List Editor 235
Figure 187 OTD Hierarchical and Delimiter Levels 237
Figure 188 Delimiter Type - Escape 239
Figure 189 Multiple Delimiter Example 241
Figure 190 Optional Mode Property (Example 1) 243
Figure 191 Optional Mode Property (Example 2) 244
Figure 192 Terminator Mode Property Example 245
Figure 193 Sample OTD Tree 246
Figure 194 Initial Root Node Properties 246
Figure 195 Activated delim Value Field 247
Figure 196 Delimiter List Editor - Delimiters Not Set 247
Figure 197 Delimiter List Editor - Insert Level 248
Figure 198 Delimiter List Editor - Add Delimiter 248
Figure 199 Delimiter List Editor - Add Levels and Delimiters 249
Figure 200 Root Node - Delimiter Specified 249
Figure 201 Element_1 Node Properties 250
Figure 202 Element_2 Node Properties 250
Figure 203 Field_1 Node Properties 251
Figure 204 Collaboration Definition Context Menu 254
Figure 205 Collaboration Editor (Java) 256
Figure 206 Collaboration Editor (Java) - Source Code Mode 257
Figure 207 Business Rules Editor 261
Figure 208 Java Source Editor 264
Figure 209 Business Rules Designer 266
Figure 210 Node Menu 268
Figure 211 Method-Access Nodes 269
Figure 212 Business Rules Designer: Method Menus and Boxes 270
Figure 213 Collaboration Method Palette (Java) 271
Figure 214 Expanded Method Box 271
Figure 215 Collapsed Method Box 271
Figure 216 Method Box Context Menu - Literal 272
Figure 217 Array Methods Menu 273
Figure 218 Assignment Menu 275
Figure 219 Boolean Methods Menu 278
Figure 220 Comparison Methods Menu 280
Figure 221 Math Methods Menu 282
Figure 222 Object Methods Menu 285

Figure 223 Operators Menu 287
Figure 224 String Methods Menu 289
Figure 225 Class Browser Icon 292
Figure 226 Class/Method Browser 292
Figure 227 Class and Field 294
Figure 228 Class Browser - Select Constructor 295
Figure 229 Constructor Box on Mapping Canvas 296
Figure 230 Node Context Menu 297
Figure 231 Method Dialog (for type TextMessage) 298
Figure 232 Method List for Text 299
Figure 233 Method Box in Mapping Panel 299
Figure 234 Exception Handling Options 300
Figure 235 Automatic Conversion Option 300
Figure 236 Type Conversion - String to Number 302
Figure 237 Type Conversion - Character Encoding 304
Figure 238 Test Mode Icon 306
Figure 239 Collaboration Tester (Java) 306
Figure 240 Breakpoints in Java Code 307
Figure 241 Breakpoint Menu 308
Figure 242 Validation Panel 308
Figure 243 Source Code Mode Icon 309
Figure 244 Test Mode Icon 309
Figure 245 Initial Collaboration Tree 309
Figure 246 Root Node Context Menu 310
Figure 247 Selecting Input Data File 311
Figure 248 Populated Value Fields 312
Figure 249 Java Debugger 313
Figure 250 File Menu 314
Figure 251 Attach to JVM Dialog 314
Figure 252 Java Source Code Display 315
Figure 253 Breakpoint Example 315
Figure 254 Debug Menu 316
Figure 255 Stop in Method Dialog 316
Figure 256 Choose Exception Dialog 317
Figure 257 Break on Exception Dialog 317
Figure 258 Breakpoint Indicator 318
Figure 259 Stepping Into, Over, and Out Commands 319
Figure 260 Java Thread and Method Display 320

Figure 261 Local Variables Tab 321
Figure 262 Evaluate Local Variable 322
Figure 263 Evaluate Method 323
Figure 264 Save Debugger Session Dialog 324
Figure 265 Resume Debugger Session Dialog 324
Figure 266 Project Context Menu - New Web Service Definition 326
Figure 267 Initial Wizard Dialog 327
Figure 268 New Web Service: Operation Name 328
Figure 269 New Web Service: Input Message 329
Figure 270 New Web Service: Output Message 330
Figure 271 New Web Service: Auxiliary OTD 331
Figure 272 Existing Web Service: Select Operation 332
Figure 273 Existing Web Service: Select OTD 333
Figure 274 Collaboration Definition (Java) Properties 334
Figure 275 Collaboration Definition (Java) Properties 335
Figure 276 Collaboration Definition (Java) Properties 336
Figure 277 Class Icon 337
Figure 278 Create Class Dialog 337
Figure 279 Method Icon 339
Figure 280 Create Method Dialog 339
Figure 281 Create Parameter Dialog 341
Figure 282 Constructor Icon 343
Figure 283 Create Constructor Dialog 343
Figure 284 Create Parameter Dialog 345
Figure 285 Field Icon 346
Figure 286 Create Field Dialog 346
Figure 287 New Array Icon 348
Figure 288 New Array Dialog 348
Figure 289 Local Variable Icon 350
Figure 290 Create Variable Dialog 350
Figure 291 Create Variable Dialog 351
Figure 292 Class Browser - Select Class 352
Figure 293 Business Rules with Variable 352
Figure 294 Variable Dialog 353
Figure 295 Modify Variable Dialog 353
Figure 296 Object Icon 354
Figure 297 String Icon 354
Figure 298 Empty File Test - If Condition 355

Figure 299 Empty File Test - Alerter Node Menu 356
Figure 300 Alerter Severity Method List 357
Figure 301 Empty File Test - critical Method 357
Figure 302 Empty File Test - Pass Alert Message 358
Figure 303 Logging Example - Logger Node Menu 359
Figure 304 Logging Level Method List 360
Figure 305 Logging Example - debug Method Box 360
Figure 306 Logging Example - Pass Log Message 361
Figure 307 Project Context Menu: Import File 362
Figure 308 File Import Dialog 363
Figure 309 File Context Menu - Update 363
Figure 310 JAR File Icon 364
Figure 311 Add/Remove Jar File Dialog (1) 364
Figure 312 Add/Remove Jar File Dialog (2) 365
Figure 313 copyNode Operation 366
Figure 314 Try Icon 368
Figure 315 Try Statement 368
Figure 316 Catch Icon 368
Figure 317 Create Exception Variable Dialog 369
Figure 318 Class Browser Dialog 369
Figure 319 Catch Statement 370
Figure 320 Encoding Converter Icon 372
Figure 321 Encoding Converter Methods Dialog 372
Figure 322 Using the unmarshalFromBytes Method 375
Figure 323 Create a Variable Dialog 376
Figure 324 Class Browser Dialog 377
Figure 325 Adding the MyConverter Local Variable 378
Figure 326 Encoding Converter Icon 378
Figure 327 Encoding Converter Method Dialog 379
Figure 328 Creating the Encoding Converter Rule 380
Figure 329 Selecting a Method 381
Figure 330 Convert Method List 382
Figure 331 Data Mapping 383
Figure 332 Marshaling Output Data 384
Figure 333 Output Data 385
Figure 334 Validate Icon 386
Figure 335 Validating Java Code 386
Figure 336 Collaboration Definition Context Menu 388

Figure 337 Project Context Menu - New Web Service Definition 390
Figure 338 Collaboration Wizard (XSLT) Dialog 391
Figure 339 New Web Service: Operation Name 392
Figure 340 New Web Service: Input Message 393
Figure 341 New Web Service: Output Message 394
Figure 342 Existing Web Service: Select Operation 395
Figure 343 Collaboration Editor (XSLT) 396
Figure 344 Encoding Icon 399
Figure 345 Collaboration Definition Editor (XSLT): Method Menus and Boxes 400
Figure 346 Collaboration Method Palette (XSLT) 401
Figure 347 Expanded Method Box 401
Figure 348 Collapsed Method Box 401
Figure 349 Boolean Methods Menu 402
Figure 350 Nodes Methods Menu 404
Figure 351 Number Methods Menu 407
Figure 352 Operator Methods Menu 409
Figure 353 String Methods Menu 412
Figure 354 For-Each Construct - Step 1 415
Figure 355 For-Each Construct - Step 2 416
Figure 356 For-Each Construct - Step 3 416
Figure 357 Development Flow - Creating a Web Service Project 423
Figure 358 Development Flow - Publishing a Web Service Definition 424
Figure 359 Web Services Access Manager 426
Figure 360 Project Context Menu - New XML Schema Definition 427
Figure 361 XML Schema Editor 428
Figure 362 XSD Node Preview/Export Options - Output Encoding 434
Figure 363 XSD Node Preview/Export Options - Synthetic Element Name 434
Figure 364 XSD Tree Example 437
Figure 365 XSD Root Node Icon 438
Figure 366 XSD Root Node Menu 438
Figure 367 Select XSD Dialog 440
Figure 368 XSD Root Node Properties 440
Figure 369 Attribute Node Icon 442
Figure 370 Attribute Node Properties 442
Figure 371 Attribute Group Node Icon 444
Figure 372 AttributeGroup Node Menu 444
Figure 373 AttributeGroup Node Properties (Global) 444
Figure 374 AttributeGroup Node Properties (Non-global) 444

Figure 375 Select Type Dialog 445
Figure 376 Element Node Icon 446
Figure 377 Element Node Menu 446
Figure 378 Element Node Properties (Global) 447
Figure 379 Element Node Properties (Non-global) 448
Figure 380 Union Node Icon 450
Figure 381 Union Node Properties 450
Figure 382 Select Member Types Dialog 451
Figure 383 Group Node Icon 452
Figure 384 Group Node Menu 452
Figure 385 ComplexType Node Icon 453
Figure 386 ComplexType Node Menu 453
Figure 387 SimpleType Node Icon 455
Figure 388 SimpleType Node Menu 455
Figure 389 SimpleType Node Properties 456
Figure 390 Select Final Value Dialog 457
Figure 391 SimpleType Subnode Menu 458
Figure 392 Enumeration Dialog 459
Figure 393 List Type Node Icon 460
Figure 394 ListType Node Properties 460
Figure 395 Sequence Node Icon 461
Figure 396 Sequence Node Menu 461
Figure 397 Choice Node Icon 462
Figure 398 Choice Node Menu 462
Figure 399 All Node Icon 463
Figure 400 All Node Menu 463
Figure 401 Any Node Icon 464
Figure 402 Any Node Properties 464
Figure 403 Complex Content Node Icon 466
Figure 404 ComplexContent Node Menu 466
Figure 405 Simple Content Node Icon 467
Figure 406 SimpleContent Node Menu 467
Figure 407 Extension Node Icon 468
Figure 408 Extension Node Properties 468
Figure 409 Restriction Node Icon 469
Figure 410 Restriction Node Properties 469
Figure 411 Project Context Menu - Import XML Schema Definition Option 470
Figure 412 Import XSD File Dialog - Step 1 471

Figure 413 Import XSD File Dialog - Step 2 471
Figure 414 Import XSD File Dialog - Step 3 472
Figure 415 Import XSD File Dialog - Step 4 473
Figure 416 Import XSD File Dialog - Step 4 Confirmation 473
Figure 417 Import XSD File Dialog - Step 5 474
Figure 418 Project Context Menu - New Web Service Definition 475
Figure 419 Web Service Definition Editor User Interface 476
Figure 420 WSDL Tree - All Options 481
Figure 421 Web Service Definition Root Node Menu 482
Figure 422 Import Folder Menu 483
Figure 423 Select WSDL, XSD or OTD Dialog 484
Figure 424 Messages Folder Menu 485
Figure 425 Message Menu 485
Figure 426 Add Parts Dialog 486
Figure 427 Message Part Properties (Initial) 487
Figure 428 Select Part Dialog 488
Figure 429 Message Value Properties (String) 488
Figure 430 PortTypes Folder Menu 489
Figure 431 PortType Menu 489
Figure 432 Operation Menu 490
Figure 433 Message Properties 491
Figure 434 Message Dialog 492
Figure 435 Properties Folder Menu 493
Figure 436 Property Properties 493
Figure 437 Select Type Dialog 494
Figure 438 Property Aliases Folder Menu 495
Figure 439 Add Property Aliases Dialog 496
Figure 440 Service Link Type Folder Menu 497
Figure 441 Role Menu 497
Figure 442 Schema Type Menu 498
Figure 443 Project Context Menu - Import Web Service Definition Option 499
Figure 444 WSDL Import Wizard - Step 1 500
Figure 445 WSDL Import Wizard - Step 2 500
Figure 446 WSDL Import Wizard - Step 3 501
Figure 447 WSDL Import Wizard - Step 4 502
Figure 448 WSDL Import Wizard - Step 4 Confirmation 502
Figure 449 WSDL Import Wizard - Step 5 503
Figure 450 Web Service Connection 504

Figure 451 WSDL Binding Editor Displaying Code 505
Figure 452 WSDL Binding Editor - PortType Binding Properties 506
Figure 453 WSDL Binding Editor - Operation Properties 507
Figure 454 WSDL Binding Editor - Message Properties 508
Figure 455 WSDL Binding Editor - Message Body Configuration 509
Figure 456 WSDL Binding Editor - Input Header Properties 510
Figure 457 WSDL Binding Editor - Header Fault Properties 511
Figure 458 WSDL Binding Editor - Java Exception Properties 512
Figure 459 Example WSD Object in Business Rules Designer 513
Figure 460 Example Java Operator - unmarshalFromString 513
Figure 461 Method Browser - Invoke Method 514
Figure 462 Connecting WSD Objects to Web Service External Applications 514
Figure 463 Java Exception 515
Figure 464 Environment Editor 517
Figure 465 Enterprise Explorer: Environment Explorer View 518
Figure 466 Environment Context Menu 520
Figure 467 Environment Component Context Menu 522
Figure 468 Environment Context Menu - New Constant 523
Figure 469 Environmental Constants Panel 523
Figure 470 Environment Context Menu - New Keystore 525
Figure 471 Keystore Context Menu 525
Figure 472 Environment Menu - New Logical Host 526
Figure 473 Logical Host Context Menu 526
Figure 474 Logical Host Menu - New Integration Server 529
Figure 475 Integration Server Context Menu - Java Debugger 529
Figure 476 Integration Server Context Menu - Properties 530
Figure 477 Sun SeeBeyond Integration Server Configuration Properties 530
Figure 478 Logical Host Menu - New Message Server 532
Figure 479 Message Server Context Menu - Properties 533
Figure 480 Sun SeeBeyond JMS IQ Manager Configuration Properties 533
Figure 481 Environment Context Menu - New Scheduler 535
Figure 482 Environment Context Menu - New SOAP/HTTP Web Service External System 536
Figure 483 Create External System Dialog 536
Figure 484 SOAP/HTTP Web Service External System Context Menu - Properties 537
Figure 485 HTTP Basic Authentication Configuration 537
Figure 486 HTTP Proxy Server Configuration Dialog 538
Figure 487 SOAP Message Security Configuration Dialog 539
Figure 488 User Name Token Configuration Dialog 540

Figure 489 Digest Password Type Configuration Dialog 541
Figure 490 Web Service External System Configuration Dialog 542
Figure 491 SOAP/HTTP Web Service External System Context Menu - Properties 543
Figure 492 HTTP Basic Authentication Configuration Dialog 543
Figure 493 Transport Layer Security Configuration Dialog 544
Figure 494 SOAP Message Security Configuration Dialog 545
Figure 495 User Name Token Configuration Dialog 546
Figure 496 Admin Server Connection Configuration Dialog 547
Figure 497 Digest Password Type Configuration Dialog 548
Figure 498 Web Service External System Configuration Dialog 549
Figure 499 Environment Context Menu - New UDDI External System 550
Figure 500 Create UDDI External System Dialog 550
Figure 501 UDDI External System Context Menu - Properties 551
Figure 502 UDDI System Configuration Properties Dialog 551

Tables

Table 1	Text Conventions	36
Table 2	File Menu Options	48
Table 3	Tools Menu Options	48
Table 4	Tools Menu Options	49
Table 5	View Menu Options	49
Table 6	Window Menu Options	50
Table 7	Help Menu Options	50
Table 8	Enterprise Designer Toolbar Icons	51
Table 9	Wizard Navigation Buttons	52
Table 10	Browser Buttons	53
Table 11	Configuration Dialog Toolbar Buttons	54
Table 12	Repository Context Menu Options (Project Explorer)	62
Table 13	Repository Context Menu Options (Environment Explorer)	63
Table 14	Connectivity Map Menu Options - Version Control	82
Table 15	Impact Analyzer Command Buttons	104
Table 16	Project Icons	108
Table 17	Project Context Menu Options	109
Table 18	Project Component Context Menu Options	113
Table 19	Project Variable Options	126
Table 20	Project Variable Options	127
Table 21	Connectivity Map Editor Toolbar Icons	131
Table 22	JMS Client Configuration Properties	134
Table 23	JMS Client Basic Configuration Properties	134
Table 24	JMS Client Redelivery Handling Configuration Properties	136
Table 25	JMS Client Advanced Configuration Properties	137
Table 26	Deployment Editor Toolbar Commands	139
Table 27	Commandline Codegen Properties	149
Table 28	Encoding Options According to Locale	160
Table 29	OTD Editor User Interface	168
Table 30	OTD Editor Toolbar Icons	169
Table 31	OTD Tester Controls	175
Table 32	Marshal and Unmarshal Process Differences	177

Table 33	DTD OTD Options	187
Table 34	XSD OTD Options	195
Table 35	Strict Validation - XSD Facets	197
Table 36	Text File Variables	202
Table 37	JMS OTD Properties	207
Table 38	OTD Context Menu Options	212
Table 39	OTD Node Context Menu Options	215
Table 40	Root Node Properties	224
Table 41	Element Node Properties	226
Table 42	Order Property Example	227
Table 43	Field Node Properties	228
Table 44	nodeType Property Options	230
Table 45	nodeType Default Values	231
Table 46	Access Parameter Options	232
Table 47	Align Parameter Options	234
Table 48	Delimiter List Editor Command Buttons	236
Table 49	Delimiter Properties	236
Table 50	Delimiter Type Options	239
Table 51	Escape Sequences	240
Table 52	Optional Mode Options	243
Table 53	Terminator Mode Options	245
Table 54	Collaboration Definition Context Menu Options	254
Table 55	Collaboration Editor Toolbar Icons	258
Table 56	Business Rules Editor Toolbar Buttons	261
Table 57	Java Source Editor Toolbar Icons	264
Table 58	Business Rules Designer Toolbar Icons	267
Table 59	Business Rules Designer Node Menu Options	268
Table 60	Method Access Nodes	269
Table 61	Array Operation Collaboration Methods (Java)	273
Table 62	Assignment (Java)	275
Table 63	Boolean Collaboration Methods (Java)	278
Table 64	Additional Boolean Collaboration Methods (Java)	279
Table 65	Comparison Collaboration Methods (Java)	280
Table 66	Math Collaboration Methods (Java)	282
Table 67	Object Collaboration Methods (Java)	285
Table 68	Operators (Java)	287
Table 69	String Collaboration Methods (Java)	289
Table 70	Method Browser Toolbar Icons	293

Table 71 Method Dialog Toolbar Icons 298
Table 72 Supported Datatype Conversions 301
Table 73 Special Characters 303
Table 74 Collaboration Tester (Java) Commands 307
Table 75 Breakpoint Menu Options 308
Table 76 Class Options 338
Table 77 Method Options 340
Table 78 Parameter Options 342
Table 79 Constructor Options 344
Table 80 Parameter Options 345
Table 81 Field Options 347
Table 82 Array Options 348
Table 83 Variable Options 350
Table 84 Literals in Toolbar Menus 354
Table 85 Byte-Array Encoding Converters - Japanese 371
Table 86 Byte-Array Encoding Converters - Korean 371
Table 87 Encoding Converter Methods Dialog Options 373
Table 88 Collaboration Definition Context Menu Options 388
Table 89 XSLT Toolbar Icons 398
Table 90 XSLT Tester Icons 399
Table 91 Boolean Collaboration Methods (XSLT) 402
Table 92 Nodes Collaboration Methods (XSLT) 404
Table 93 Number Collaboration Methods (XSLT) 407
Table 94 Operator Collaboration Methods (XSLT) 409
Table 95 String Collaboration Methods (XSLT) 412
Table 96 Metadata Forms 419
Table 97 Password Types for User Name Token 421
Table 98 XML Schema Editor Toolbar Icons 430
Table 99 XSD Node Preview/Export Options 433
Table 100 Duplicated Global Schema Component Options 435
Table 101 Enforce UPA Rule Options 435
Table 102 Enforce Particle Restriction Rule Options 436
Table 103 XSD Root Node Menu Options 438
Table 104 XSD Root Node Property Options 441
Table 105 Attribute Node Property Options 442
Table 106 AttributeGroup Node Menu Options 444
Table 107 AttributeGroup Node Property Options 445
Table 108 Element Node Menu Options 446

Table 109 Element Node Property Options (Global) 447
Table 110 Element Node Property Options 448
Table 111 Union Node Property Options 450
Table 112 Group Node Menu Options 452
Table 113 ComplexType Node Menu Options 453
Table 114 SimpleType Node Menu Options 455
Table 115 SimpleType Node Property Options 456
Table 116 SimpleType Subnode Menu Options 458
Table 117 ListType Node Property Options 460
Table 118 Sequence Node Menu Options 461
Table 119 Choice Node Menu Options 462
Table 120 All Node Menu Options 463
Table 121 Any Node Property Options 464
Table 122 ComplexContent Node Menu Options 466
Table 123 SimpleContent Node Menu Options 467
Table 124 Extension Node Property Options 468
Table 125 Restriction Node Property Options 469
Table 126 Web Service Definition Editor Toolbar Icons 478
Table 127 Web Service Definition Root Node Menu Options 482
Table 128 Import Folder Menu Options 483
Table 129 Messages Folder Menu Options 485
Table 130 Message Menu Options 485
Table 131 Message Value Node Property Options 487
Table 132 PortTypes Folder Menu Options 489
Table 133 PortType Menu Options 489
Table 134 Operation Menu Options 490
Table 135 Properties Folder Menu Options 493
Table 136 Properties Folder Menu Options 495
Table 137 Service Link Type Folder Menu Options 497
Table 138 Role Menu Options 497
Table 139 Schema Type Menu Options 498
Table 140 WSDL Binding Editor Toolbar Icons 504
Table 141 PortType Binding Property Options 506
Table 142 Operation Property Options 507
Table 143 Message Property Options 508
Table 144 Message Body Property Options 509
Table 145 Header Property Options 510
Table 146 Header Fault Property Options 511

Table 147 Java Exception Property Options 512
Table 148 Environment Icons 519
Table 149 Environment Context Menu Options 520
Table 150 Environment Component Context Menu Options 522
Table 151 Environmental Constants Panel Icons 524
Table 152 Keystore Context Menu Options 525
Table 153 Logical Host Context Menu Options 527
Table 154 Sun SeeBeyond Integration Server Configuration Properties List 530
Table 155 JMS Message Server Configuration Properties List 533
Table 156 HTTP Basic Authentication Configuration Properties 538
Table 157 HTTP Proxy Server Configuration Properties 538
Table 158 SOAP Message Security Configuration Properties 539
Table 159 User Name Token Configuration Properties 540
Table 160 Digest Password Type Configuration Properties 541
Table 161 Web Service External System Configuration Properties 542
Table 162 HTTP Basic Authentication Configuration Properties 544
Table 163 Transport Layer Security Configuration Properties 544
Table 164 SOAP Message Security Configuration Properties 545
Table 165 User Name Token Configuration Properties 546
Table 166 Admin Server Connection Configuration Properties 547
Table 167 Digest Password Type Configuration Properties 548
Table 168 Web Service External System Configuration Properties 549
Table 169 UDDI External System Configuration Properties 551

Chapter 1

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 34
- **What's New in This Release** on page 34
- **About This User's Guide** on page 35
- **Related Documents** on page 37
- **Sun Microsystems, Inc. Web Site** on page 37
- **Documentation Feedback** on page 37

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 34 describes the purpose of this User’s Guide, including writing conventions and a list of related documents.

2 **eGate Integrator Overview** on page 38 provides an overview of the general structure, architecture, and operation of eGate Integrator, and its place within the Sun Java™ Composite Application Platform Suite (Java CAPS).

3 **Enterprise Designer Basics** on page 45 provides an overview of the Enterprise Designer, including its structure and operation.

4 **Global Features** on page 61 describes features used to manage Repository objects in both Project and Environment realms.

5 **Projects** on page 105 describes the process of building a Java CAPS Project, including the use of Connectivity Maps and Deployment Profiles.

6 **Object Type Definitions I** on page 155 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 181 describes how to create Object Type Definitions using the OTD Wizards.

8 **Object Type Definitions III** on page 214 explores the details of User-Defined OTDs.

9 **Collaboration Definitions (Java) I** on page 252 describes how to build Collaboration Definitions described in Java™.

10 **Collaboration Definitions (Java) II** on page 325 describes a variety of specific design procedures used to create Java-based Collaboration Definitions.

11 **Collaboration Definitions (XSLT)** on page 387 describes how to build Collaboration Definitions described in XSLT.

12 **Web Services** on page 418 describes how to use eGate Integrator in concert with other Java CAPS components to create web services.

13 **Run-time Environments** on page 516 explains how to create and populate run-time Environments.

14 **Troubleshooting** on page 552 contains tips on troubleshooting problems in eGate Integrator.

In addition, the **Glossary** on page 556 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
Bold	Names of buttons, commands, menu selections, and file names, to accentuate required user actions.	▪ Click **OK** to save and close. ▪ From the File menu, select **Exit**. ▪ Select the **logicalhost.exe** file.
`Monospaced`	Command-line arguments and code samples (variables are shown within angle brackets).	`bootstrap -p` <password>
Blue bold	Hypertext links within document.	See **Text Conventions** on page 36
Blue underlined	Hypertext links for Web addresses (URLs) or email addresses.	http://www.seebeyond.com docfeedback@seebeyond.com

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 CAPS documentation is included in the *Sun Java™ Composite Application Platform Suite Primer*.

The documentation for Java CAPS 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

Chapter 2

eGate Integrator Overview

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

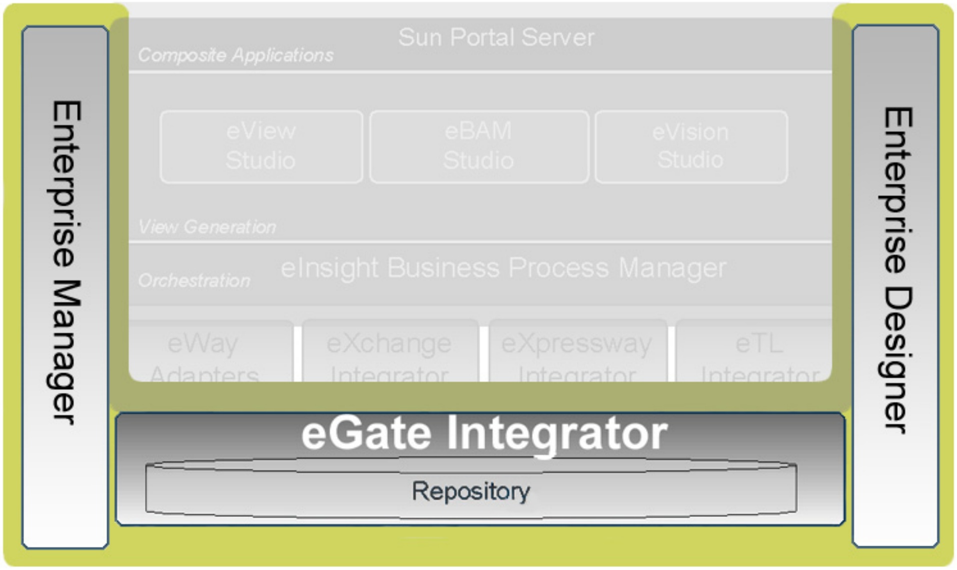
What's in This Chapter

- **Introduction** on page 38
- **Integration Model** on page 39
- **System Architecture** on page 40

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 together 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.

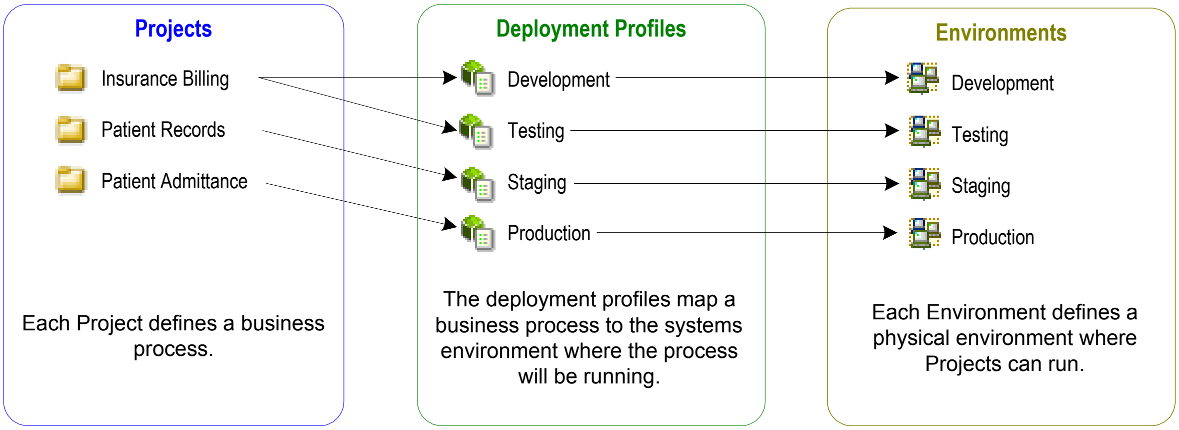
Projects are run within individual sets of system definitions, referred to as Logical Hosts. These are defined within Environments, which represent the physical resources required to implement the Project. Projects are mapped to the individual Environments by means of deployment profiles, which are defined 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


Projects
Insurance Billing
Patient Records
Patient Admittance
Each Project defines a business process.
Deployment Profiles
Development
Testing
Staging
Production
The deployment profiles map a business process to the systems environment where the process will be running.
Environments
Development
Testing
Staging
Production
Each Environment defines a physical environment where Projects can run.

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.

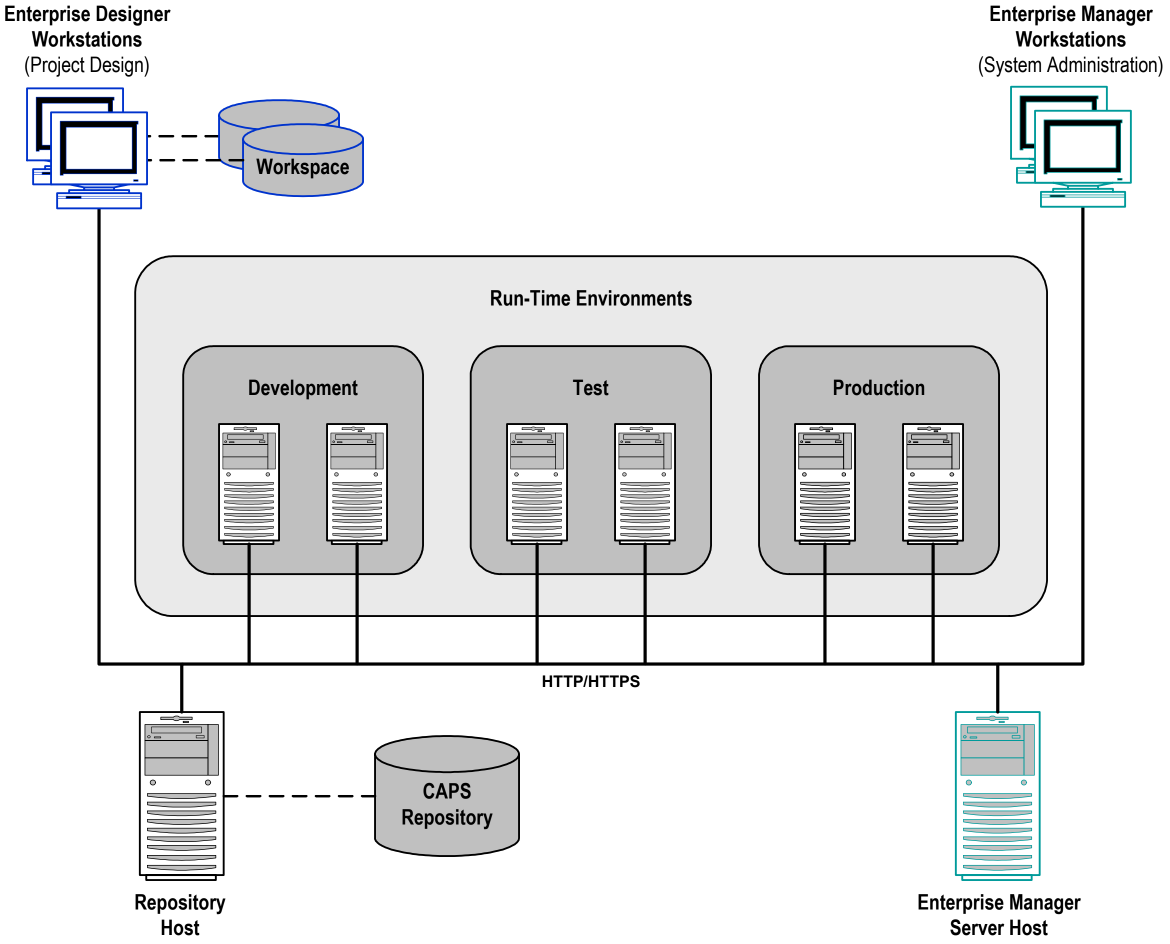
Typically, a particular Project will exist in different versions — a mature version in the production environment, and a follow-on version in the development environment, for example. Isolating these different versions from each other is handled by maintaining separate branches of the Repository.

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


Enterprise Designer Workstations (Project Design)
Workspace
Enterprise Manager Workstations (System Administration)
Run-Time Environments
Development
Test
Production
HTTP/HTTPS
CAPS Repository
Repository Host
Enterprise Manager Server Host

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.

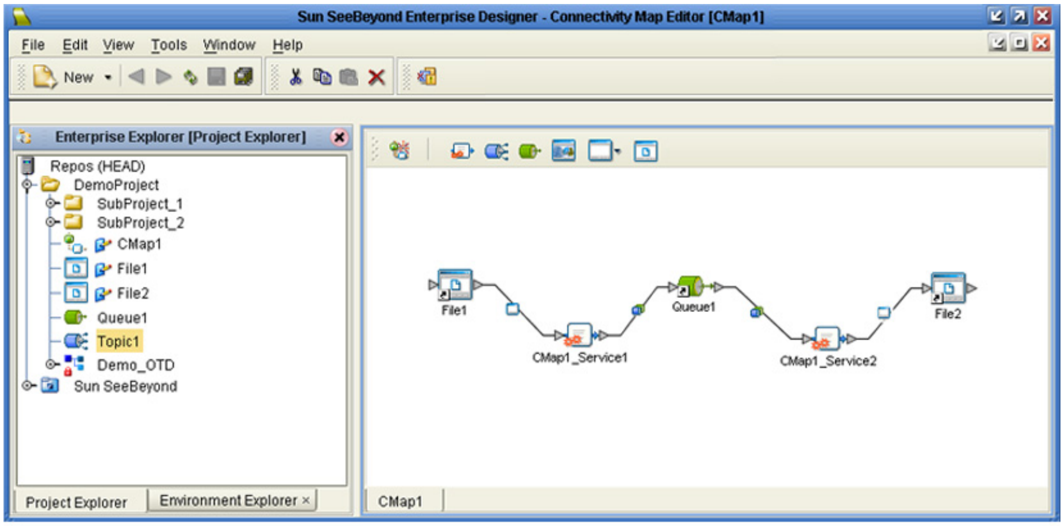
An integration Project is created within the development environment, then migrated to the test environment, and finally to the production environment. This migration path is a necessary and highly critical practice in implementing a working system.

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 105.

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.

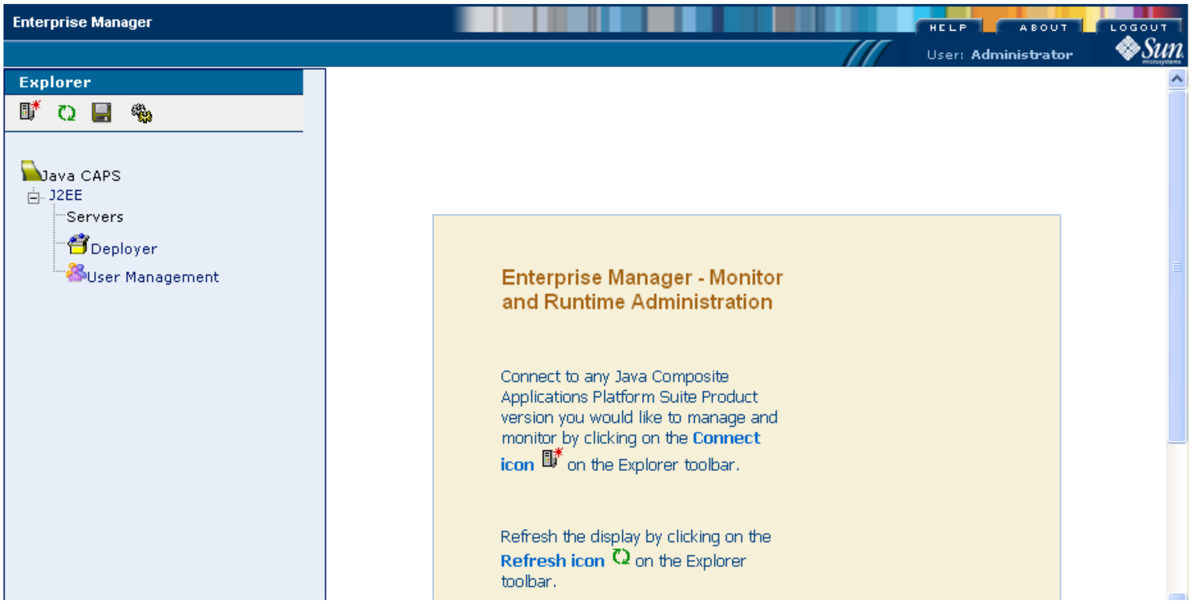
Enterprise Designer additionally supports security-related system administration tasks such as managing access to various components and features in the Java™ Composite Application Platform Suite. The use of Enterprise Designer in performing these tasks is described in the *Sun SeeBeyond eGate™ Integrator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



Enterprise Manager - Monitor and Runtime Administration

Connect to any Java Composite Applications Platform Suite Product version you would like to manage and monitor by clicking on the **Connect icon** on the Explorer toolbar.

Refresh the display by clicking on the **Refresh icon** on the Explorer toolbar.

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*.

Chapter 3

Enterprise Designer Basics

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

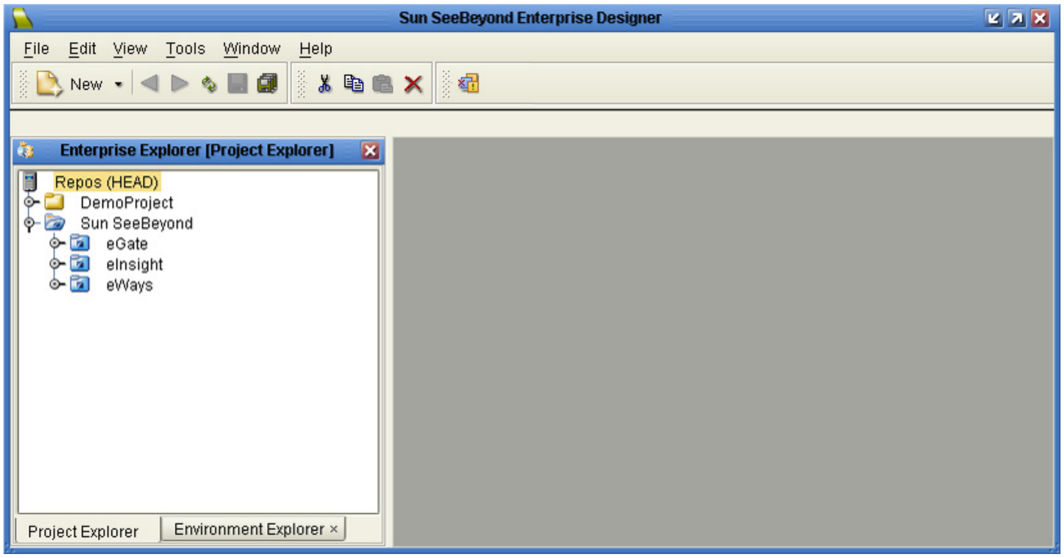
What's in This Chapter

- **Overview** on page 45
- **Starting Enterprise Designer** on page 47
- **Navigating Enterprise Designer** on page 48
- **Configuring Enterprise Designer** on page 55

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 107
- **Using Environment Explorer** on page 518

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 130
- **Using the Deployment Editor** on page 138
- **Using the OTD Editor** on page 168
- **Using the Collaboration Editor (Java)** on page 256
- **Using the Collaboration Editor (XSLT)** on page 396
- **Using the XML Schema Editor** on page 428
- **Using the Web Service Definition Editor** on page 476
- **Using the Environment Editor** on page 517
- **Using the Java Debugger** on page 313

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 65).
- Managing the versions of Project and Environment components (see **Using Version Control Features** on page 79).
- Analyzing the impact of removing components (see **Analyzing Component Dependencies** on page 102).

3.2 Starting Enterprise Designer

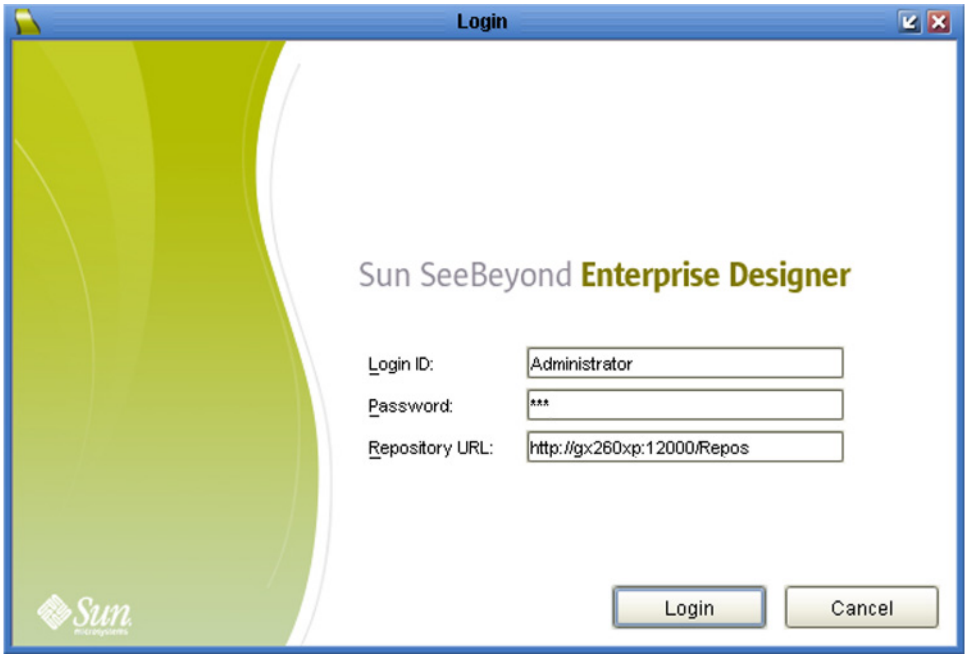
Before we proceed further, perhaps you would like to know how to start Enterprise Designer. It's described more completely in the *Sun Java™ Composite Application Platform Suite Installation Guide*, but here's a condensed version:

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



Login ID: Administrator
Password: ***
Repository URL: http://gx260xp:12000/Repos

2 Type your *Login ID* and *Password* into the respective boxes (the Login ID from the last successful login is entered automatically).

Important: *User names and passwords should contain only alphanumeric characters (letters and numbers), dashes, and underscores. Any leading or trailing spaces are automatically removed from the password before processing.*

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 **New** button in the toolbar and the **New** 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 **Moving and Replicating Repository Objects** on page 100 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 **Moving and Replicating Repository Objects** on page 100 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 **Moving and Replicating Repository Objects** on page 100 for additional information.
Delete	Deletes the selected object, subject to the following conditions: ▪ You have *write* privileges for the object. ▪ The object is not checked out by anyone other than yourself. If these conditions are true, a dialog is displayed in which you confirm that you want to delete the selected object. Clicking **Yes** then deletes the object. 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 **OK** button.
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 **Analyzing Component Dependencies** on page 102.
Options	Presents the Options Setup dialog, in which you can specify selected options such as heap sizes and language extensions. See **Setting Up Options** on page 55.
Update Center	Presents a series of dialogs in which you can check for program updates. See the *Sun Java™ Composite Application Platform Suite Installation Guide*.

View Menu

Table 5 View Menu Options

Option	Action
Environment Explorer	Activates the **Environment Explorer** tab on the Enterprise Explorer.
Project Explorer	Activates the **Project Explorer** 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

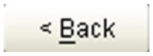
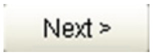
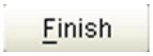
Table 8 Enterprise Designer Toolbar Icons

Icon	Command	Action
🗋	New	Displays a submenu that lists the objects that can be added to the selected node (duplicates the **New** menu in the menu bar and the **New** 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 *not* 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 *not* updated. This icon is inactive if no changes have been made.
✂	Cut	Duplicates the action of the **Cut** option in the **Edit Menu** on page 48.
⧉	Copy	Duplicates the action of the **Copy** option in the **Edit Menu** on page 48.
📋	Paste	Duplicates the action of the **Paste** option in the **Edit Menu** on page 48.
✕	Delete	Duplicates the action of the **Delete** option in the **Edit Menu** on page 48.
⚠	Impact Analyzer	Presents the *Impact Analyzer* 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 **Customizer** 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

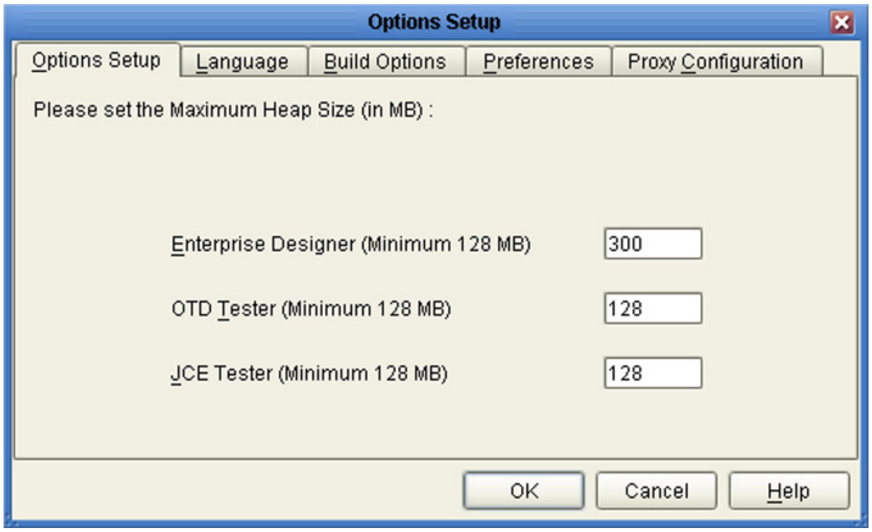
Enterprise Designer offers several global setup and configuration options, which are accessed by clicking **Options** on the *Tools* menu.

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 552.*

Figure 8 Options Setup - Heap Size Dialog



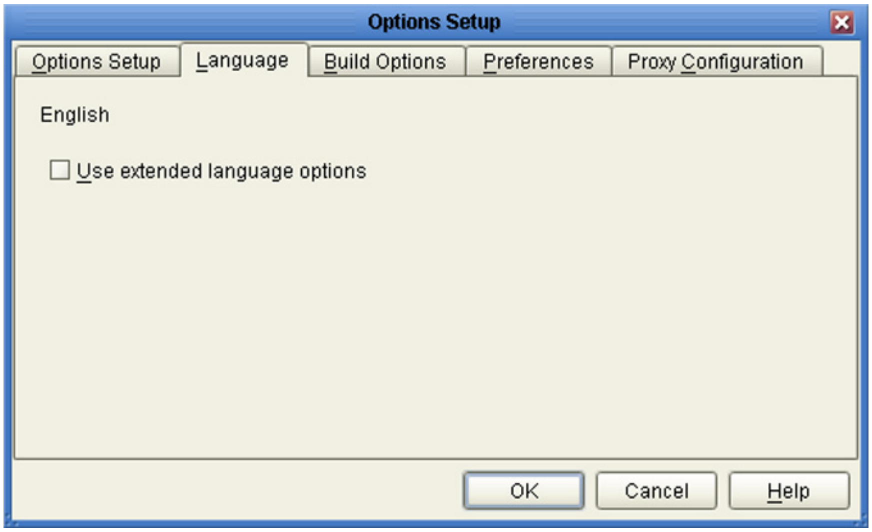
Please set the Maximum Heap Size (in MB) :

Enterprise Designer (Minimum 128 MB)	300
OTD Tester (Minimum 128 MB)	128
JCE Tester (Minimum 128 MB)	128

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



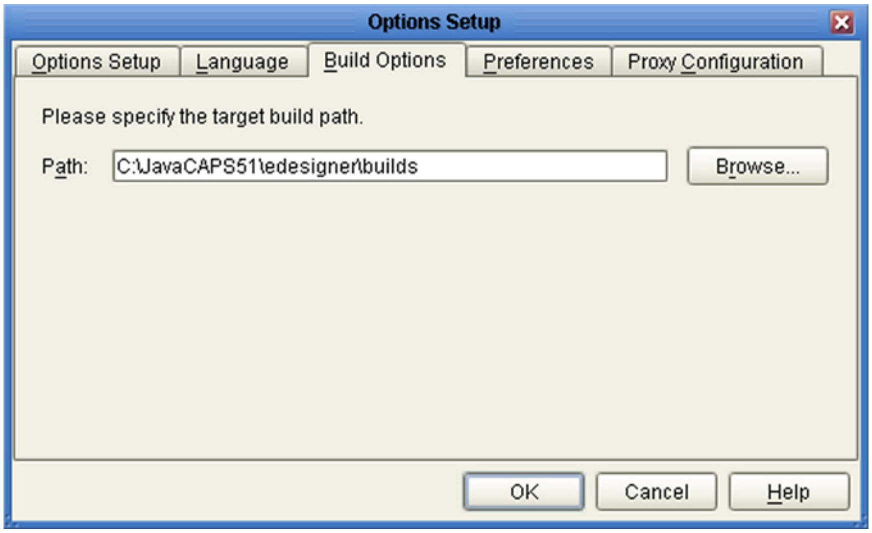
English

☐ Use extended language options

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



Please specify the target build path.

Path: C:\JavaCAPS51\edesigner\builds

Preferences

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

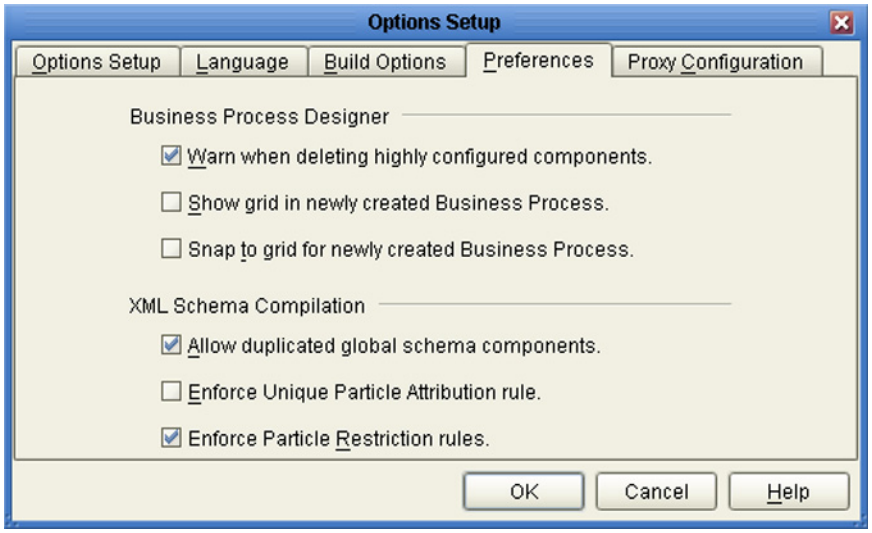
- Business Process Designer

  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 435).

Figure 11 Options Setup - Preferences Dialog



Business Process Designer

☑ Warn when deleting highly configured components.

□ Show grid in newly created Business Process.

□ Snap to grid for newly created Business Process.

XML Schema Compilation

☑ Allow duplicated global schema components.

□ Enforce Unique Particle Attribution rule.

☑ Enforce Particle Restriction rules.

Proxy Configuration

The Proxy Configuration allows Enterprise Designer to be used in an environment that requires that access to the internet be made through an HTTP Proxy Server. If you want to import WSDL or XSD files that reside on the internet, or are dependent upon other files that reside on the internet, you must configure the proxy server.

- **Proxy Host**

  The name of the HTTP Proxy Host, which must be resolvable to an IP address; an entry is required.

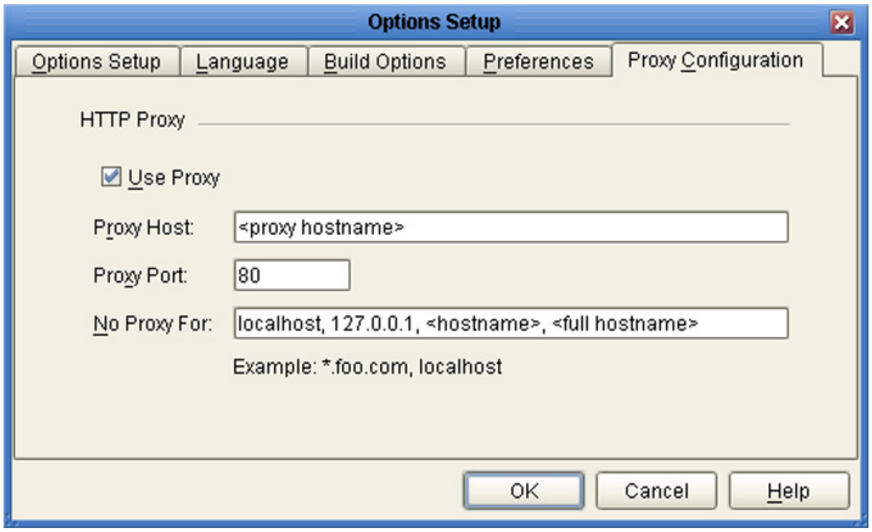
- **Proxy Port**

  The listening port of the HTTP Proxy Server, which must be a number between **1** and **65535** (inclusive).

- **No Proxy For**

  Specifies which host access should not go through the HTTP Proxy Server. By default, this field is initialized with various names representing the local host that is running Enterprise Designer. *You should not change these entries*, but you may append additional names to the list. Entries are separated by commas (,) and can be names or wildcard domains, such as those shown in the example beneath the field.

Figure 12 Options Setup - Proxy Configuration Dialog



HTTP Proxy

☑ Use Proxy

Proxy Host: <proxy hostname>

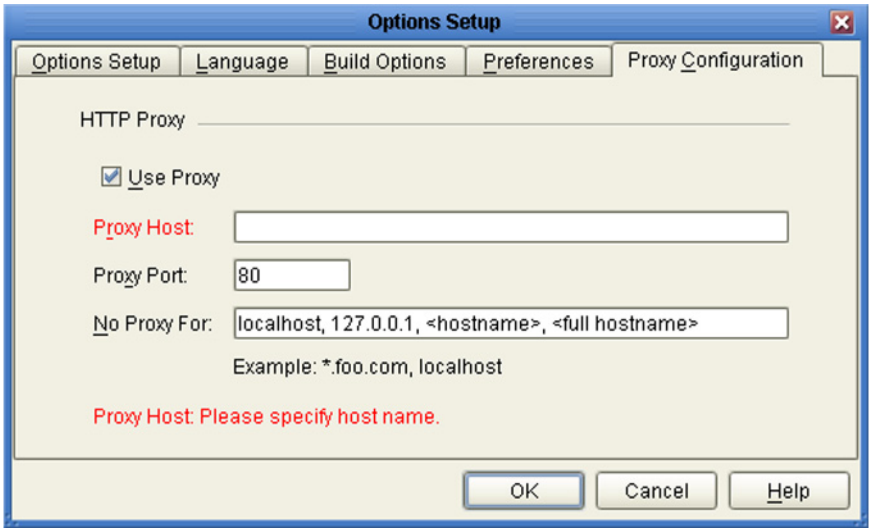
Proxy Port: 80

No Proxy For: localhost, 127.0.0.1, <hostname>, <full hostname>

Example: *.foo.com, localhost

It is imperitive that *valid* data be entered, since invalid data can not only circumvent access to the desired internet resource, but also to the Java CAPS Repository. Hence, validation of the entered data is performed automatically when the cursor focus is transferred away from the text field. An invalid entry results in the field label turning red, and a status line appearing, stating the nature of the error (see Figure 13).

Figure 13 Options Setup - Proxy Configuration Dialog - Error



HTTP Proxy

☑ Use Proxy

Proxy Host:

Proxy Port: 80

No Proxy For: localhost, 127.0.0.1, <hostname>, <full hostname>

Example: *.foo.com, localhost

Proxy Host: Please specify host name.

After you have entered the Proxy Name and Proxy Port, Enterprise Designer will attempt to establish a connection to the HTTP Proxy server. If the connection cannot be made, you will receive a message on the status line asking you to correct the data.

3.4.2 Customizing the User Interface

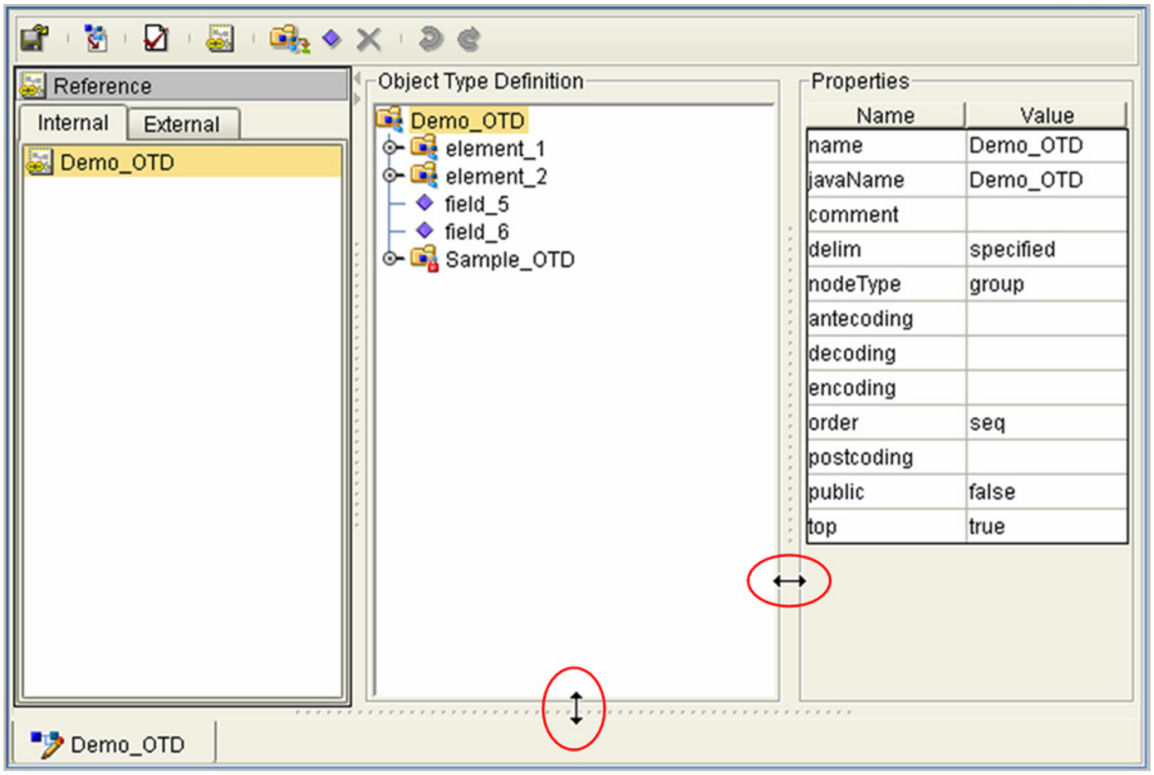
Enterprise Designer is based on the NetBeans windowing system, and offers most of the standard NetBeans window management features. These features allow you to customize the appearance of Enterprise Designer (within limits, of course) to suit your work style. Extensive modification of the interface is 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 14, for example.

Figure 14 Draggable Splitters

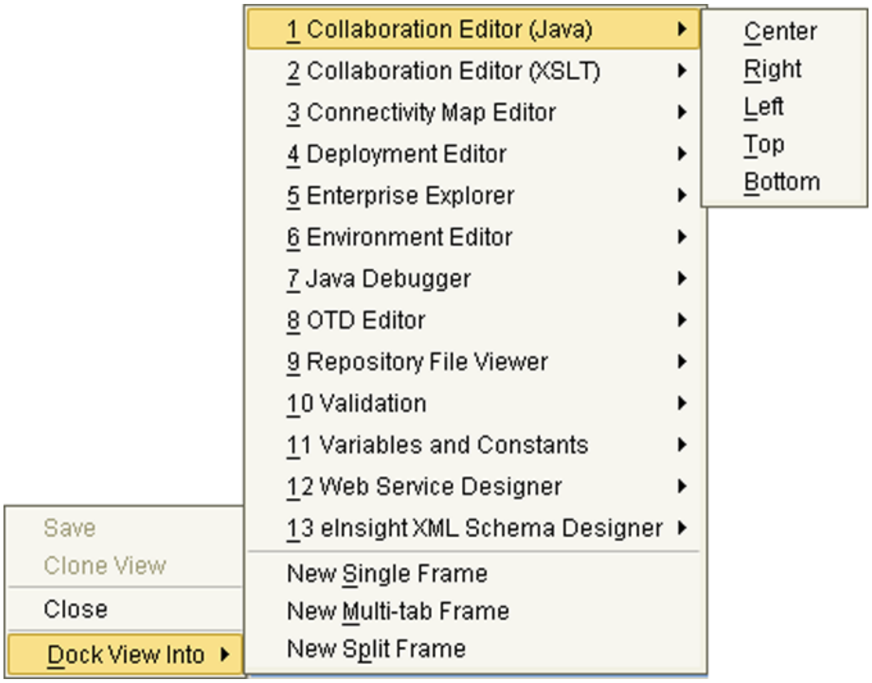


Name	Value
name	Demo_OTD
javaName	Demo_OTD
comment	
delim	specified
nodeType	group
antecoding	
decoding	
encoding	
order	seq
postcoding	
public	false
top	true

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 15.

Figure 15 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.

Chapter 4

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 61
- **The Repository** on page 62
- **Using Export/Import Features** on page 65
- **Using Version Control Features** on page 79
- **Moving and Replicating Repository Objects** on page 100
- **Analyzing Component Dependencies** on page 102

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 16.

Figure 16 Repository Context Menu (Project Explorer)

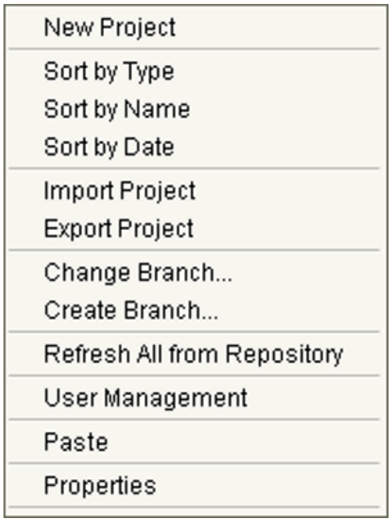


Table 12 Repository Context Menu Options (Project Explorer)

Option	Function
New Project	Adds a new Project to the current repository branch (see *Note* 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 **Importing a Project or Environment** on page 70.
Export Project	Presents a dialog with which you can export a Project from the current repository branch to an archive file.**Exporting a Project or Environment** on page 65
Change Branch	Presents a dialog with which you can switch to a different repository branch (if one exists). See **Changing Branches** on page 80.
Create Branch	Presents a dialog with which you can create a new repository branch. See the *Sun SeeBeyond eGate™ Integrator System Administration Guide*.
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 *Sun SeeBeyond eGate™ Integrator System Administration Guide*.
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 *Sun SeeBeyond eGate™ Integrator System Administration Guide*.

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 17.

Figure 17 Repository Context Menu (Environment Explorer)

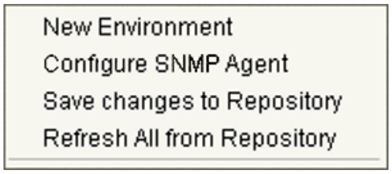


Table 13 Repository Context Menu Options (Environment Explorer)

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:

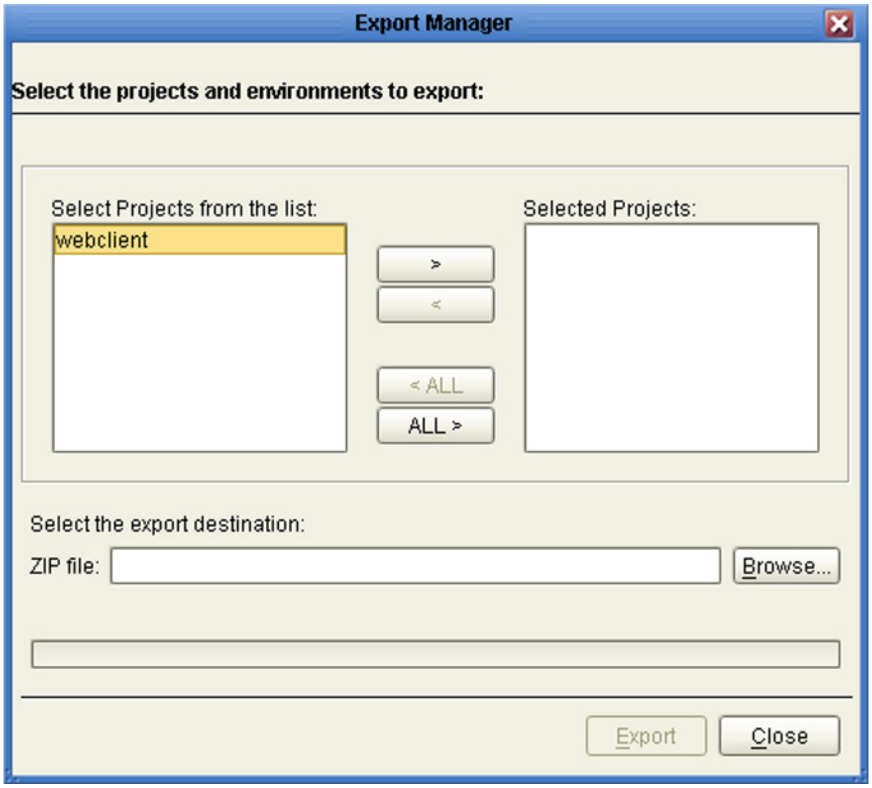
- The exported Project may have references to elements that are in other Projects. A list of such references is generated during the export process.
- Project deployment objects are *not* exported, since they have references to both Project and Environment elements that are not required at the Project 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 18. If you do, you will see the dialog shown in Figure 19.

Figure 18 Export Manager Dialog (1a)


Export Manager

Select the projects and environments to export:

Select Projects from the list:

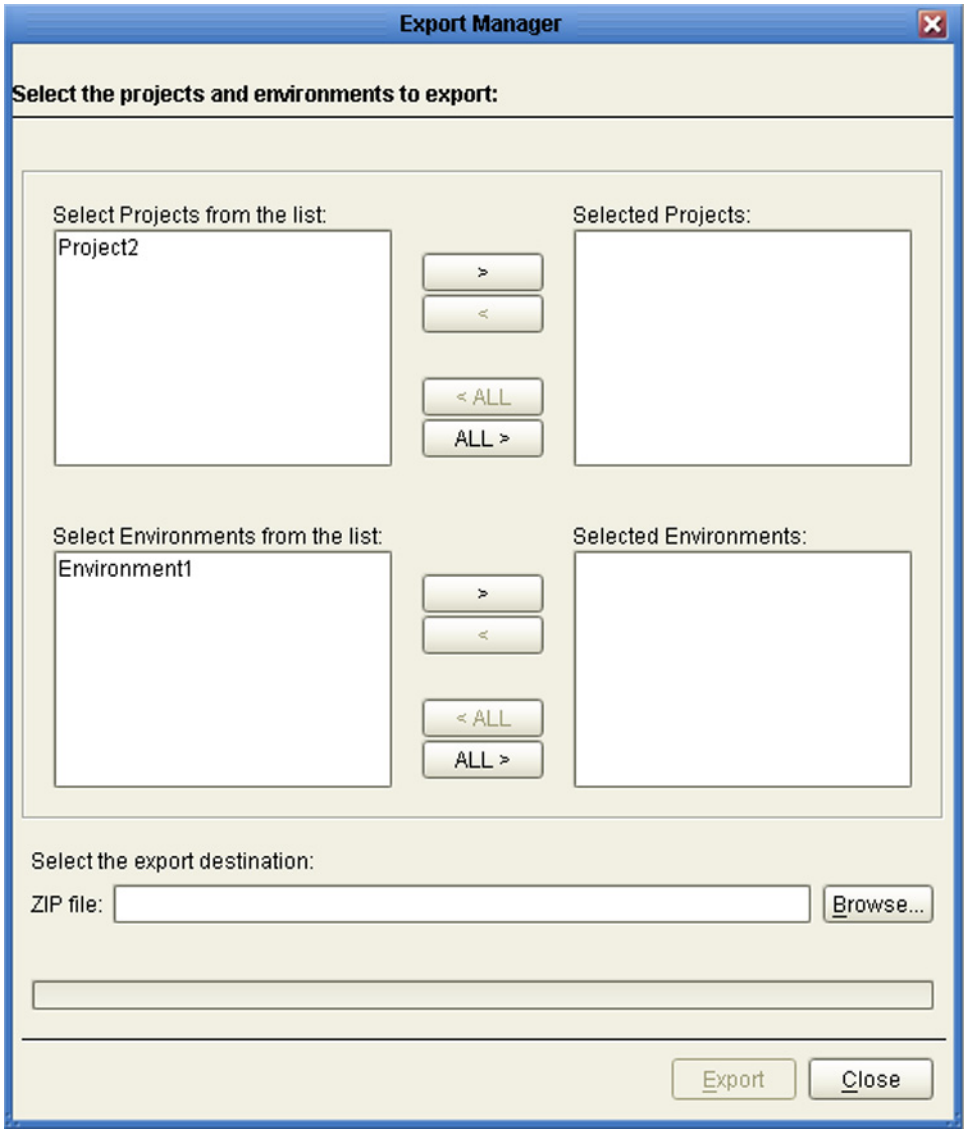
webclient

Selected Projects:

Select the export destination:

ZIP file:

Figure 19 Export Manager Dialog (1b)



Export Manager

Select the projects and environments to export:

Select Projects from the list: Project2

Selected Projects:

Select Environments from the list: Environment1

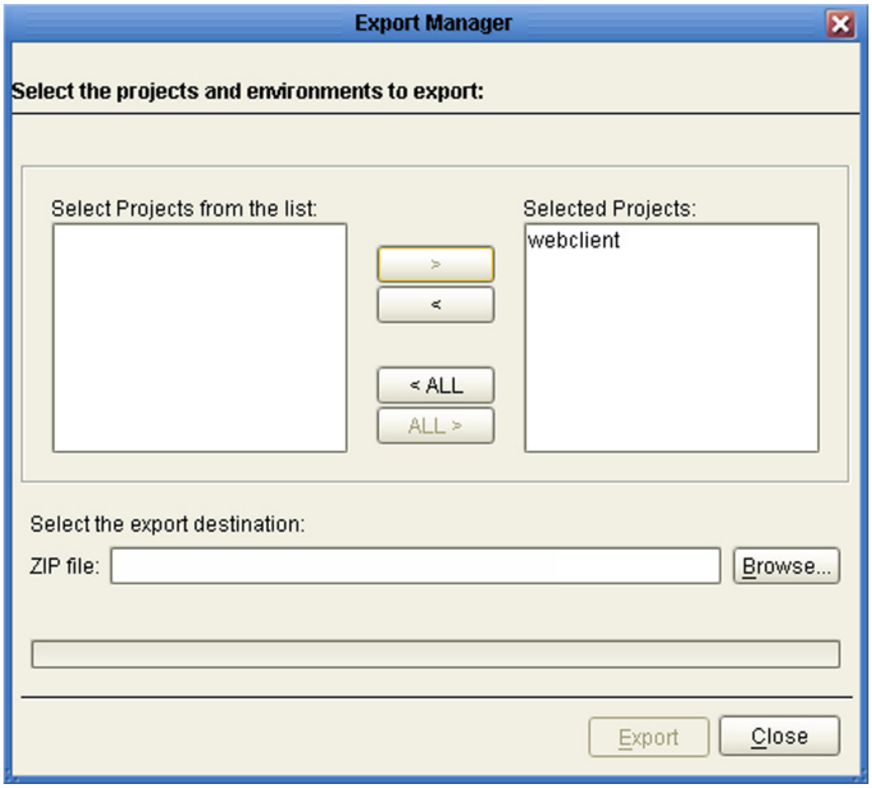
Selected Environments:

Select the export destination:

ZIP file:

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 20).

Figure 20 Export Manager Dialog (2)



Export Manager

Select the projects and environments to export:

Select Projects from the list:

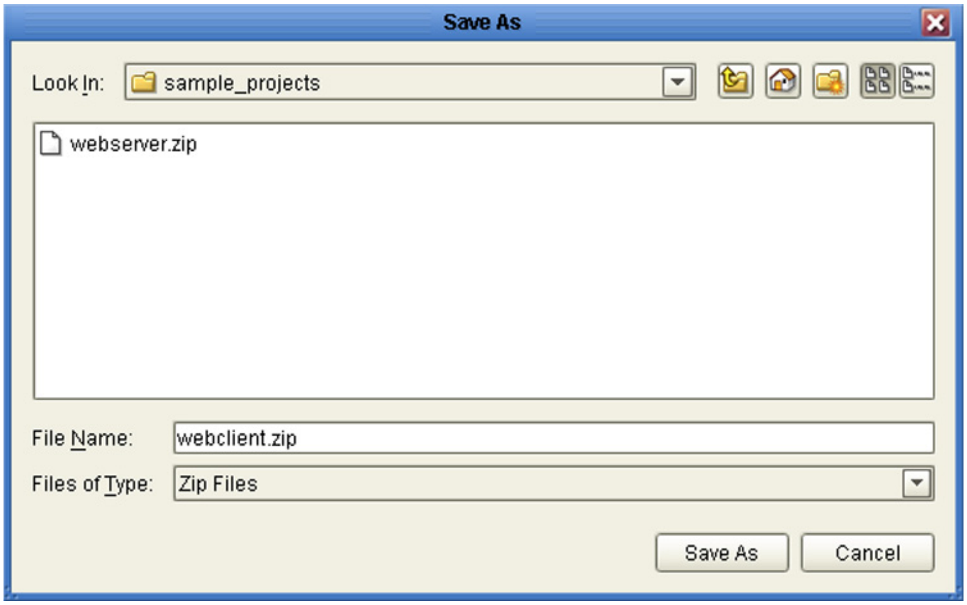
Selected Projects: webclient

Select the export destination:

ZIP file:

3 Click the **Browse** button to display the *Save As* dialog, as shown in Figure 21.

Figure 21 Save As Dialog



Save As

Look In: sample_projects

webserver.zip

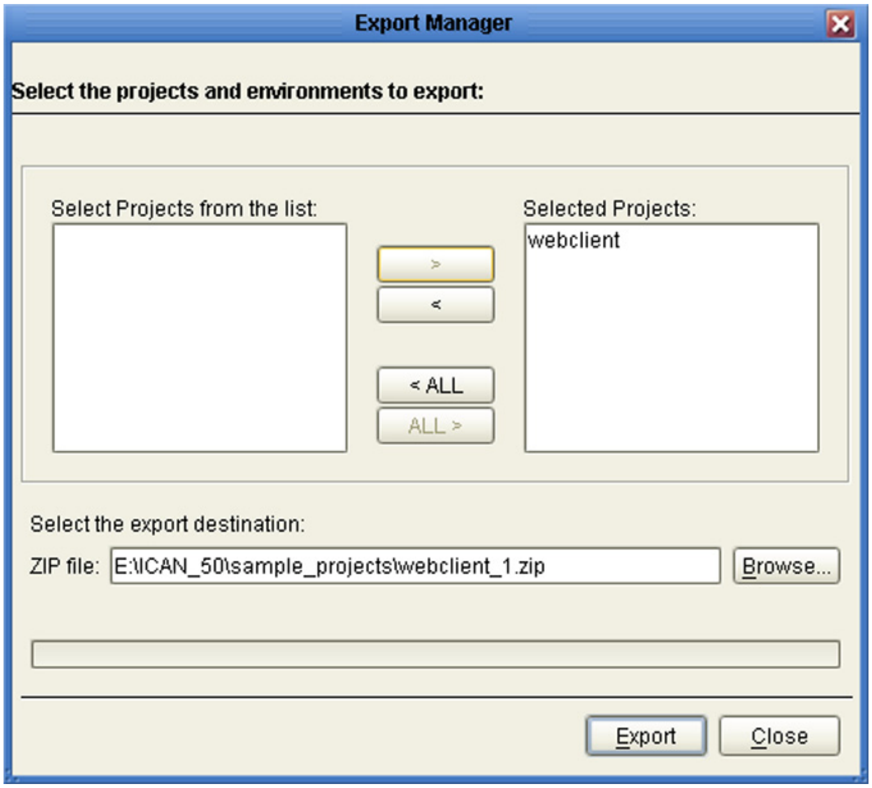
File Name: webclient.zip

Files of Type: Zip Files

4 Select the export destination and change the export file name, if desired.

5 Click **Save As** to enter the file name.

Figure 22 Enter File Name Dialog (2)



Export Manager

Select the projects and environments to export:

Select Projects from the list:

Selected Projects: webclient

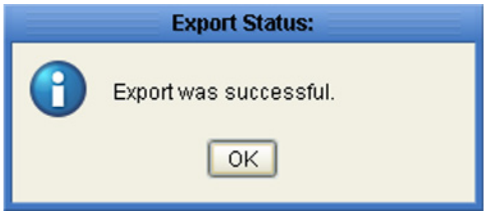
Select the export destination:

ZIP file: E:\ICAN_50\sample_projects\webclient_1.zip

6 Click **Export** to export the Project file (this process may take a few minutes).

7 The Export Status message box shown in Figure 23 appears after the file has been exported successfully.

Figure 23 Export Status Message Box



Export Status:

Export was successful.

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 65).

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.x.
- 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 24).

B For a Sub-Project, use the Project context menu (see Figure 25).

Figure 24 Repository Context Menu - Import Project Option

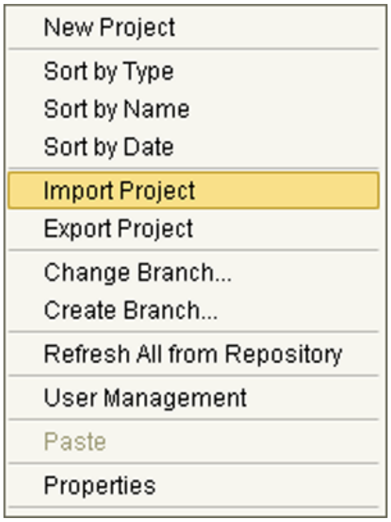
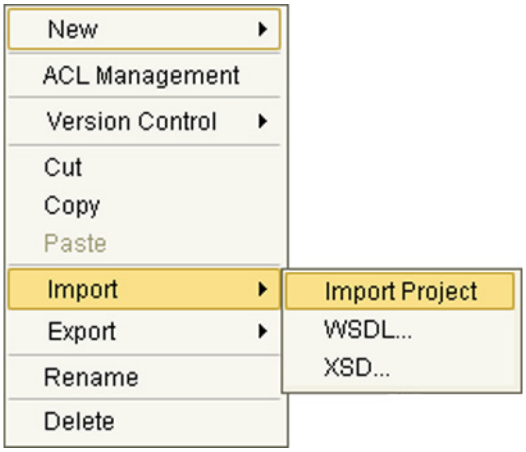
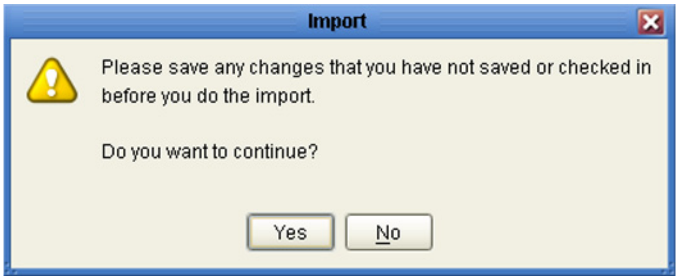


Figure 25 Project Context Menu - Import Options



2 The message box shown in Figure 26 appears, prompting you to save your changes.

Figure 26 Import Message Box



Import

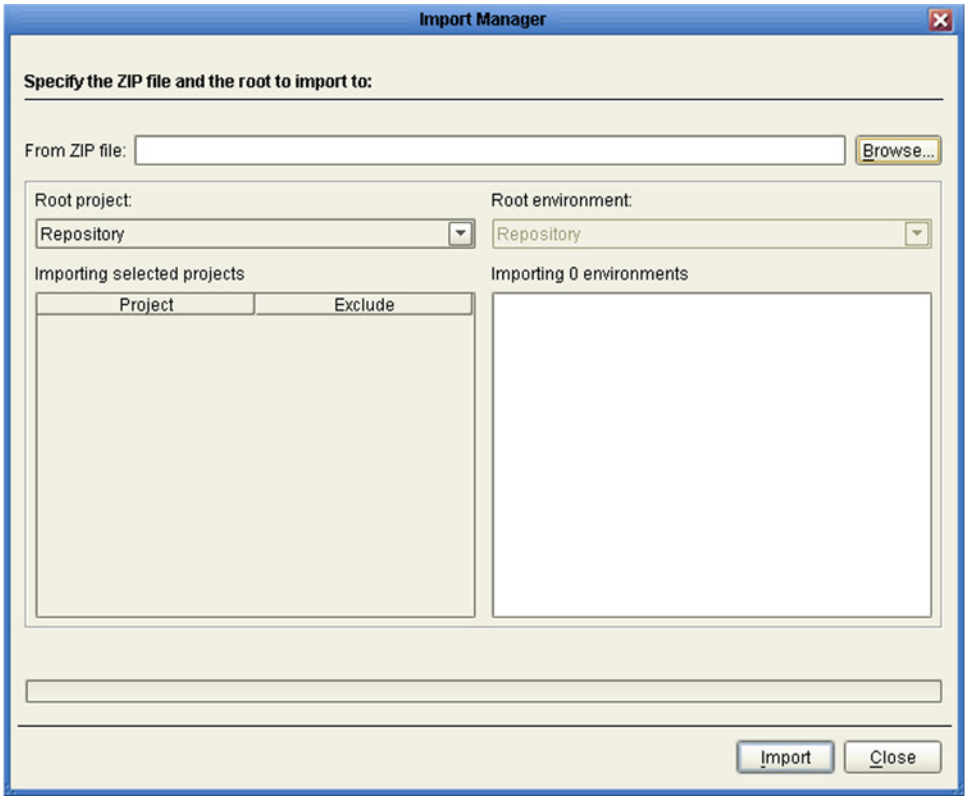
Please save any changes that you have not saved or checked in before you do the import.

Do you want to continue?

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 27.

Figure 27 Import Manager Dialog (1)



Import Manager

Specify the ZIP file and the root to import to:

From ZIP file:

Root project: Repository

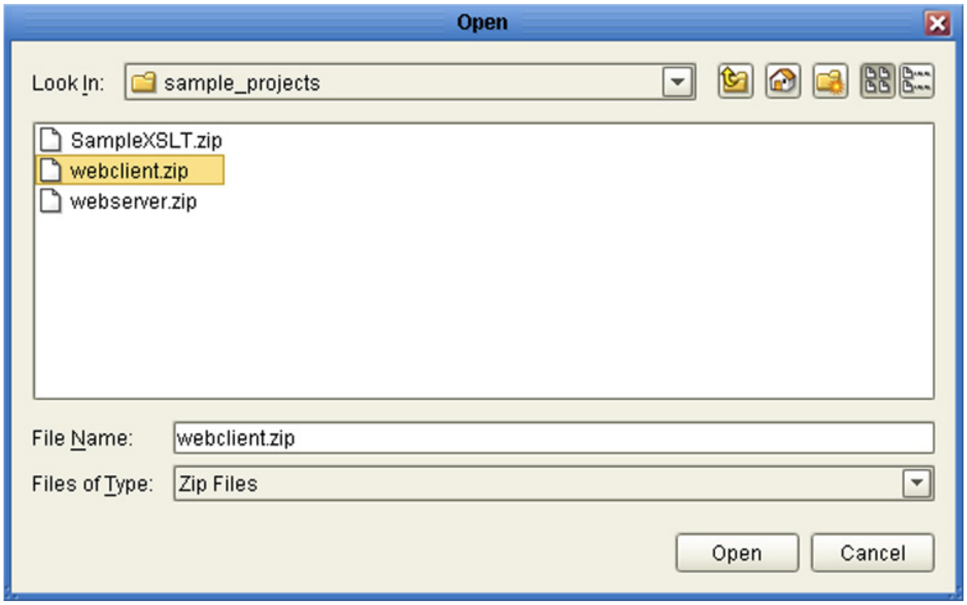
Root environment: Repository

Importing selected projects

Importing 0 environments

3 Click the **Browse** button to display the *Open File* dialog, as shown in Figure 28. If you browse to an Environment file, the *Root environment* field will be enabled.

Figure 28 Open File Dialog



File Name: webclient.zip

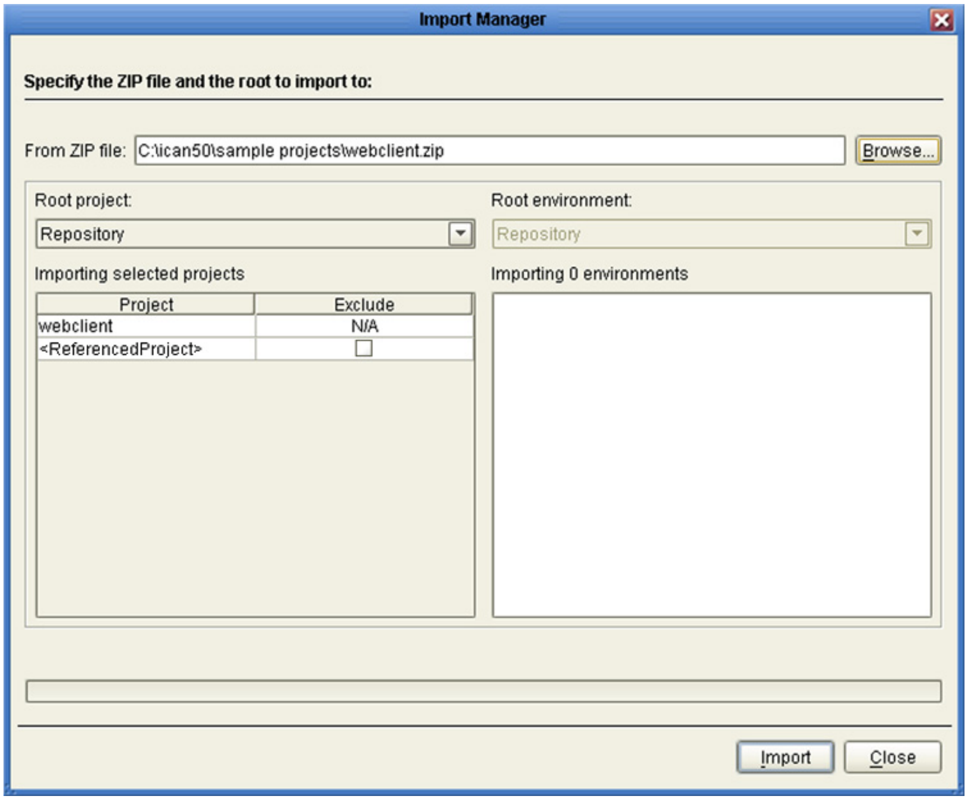
Files of Type: Zip Files

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 29.

Figure 29 Import Manager Dialog



Specify the ZIP file and the root to import to:

From ZIP file: C:\ican50\sample projects\webclient.zip

Root project: Repository

Root environment: Repository

Importing selected projects

Project	Exclude
webclient	N/A
<ReferencedProject>	☐

Importing 0 environments

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:

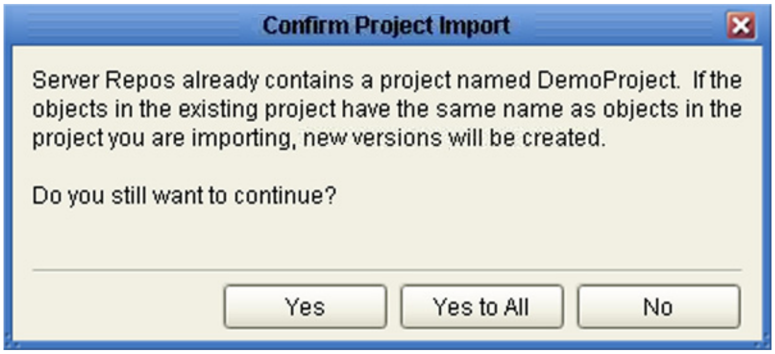
A There is no conflict with existing Project or Environment names, and the import process proceeds. Go to step 9.

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 30. 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.x.

Figure 30 Project Import Confirmation Dialog



Server Repos already contains a project named DemoProject. If the objects in the existing project have the same name as objects in the project you are importing, new versions will be created.

Do you still want to continue?

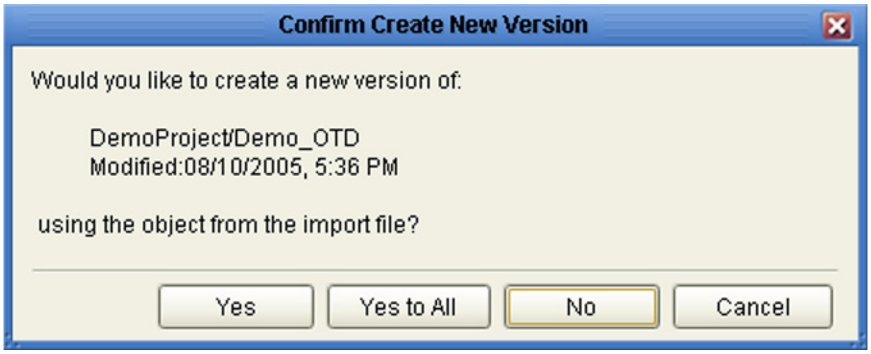
8 The dialog shown in Figure 30 offers three options:

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 31 for an example).

Figure 31 Create New Version Confirmation Dialog



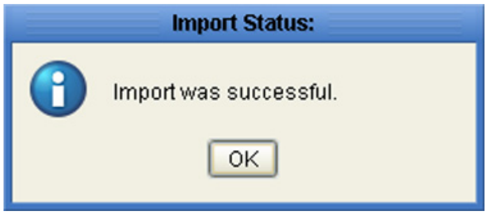
Would you like to create a new version of:

DemoProject/Demo_OTD
Modified:08/10/2005, 5:36 PM

using the object from the import file?

9 The Import Status message box shown in Figure 32 appears after the file has been imported successfully.

Figure 32 Import Status Message Box



Import Status:

Import was successful.

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.x.

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

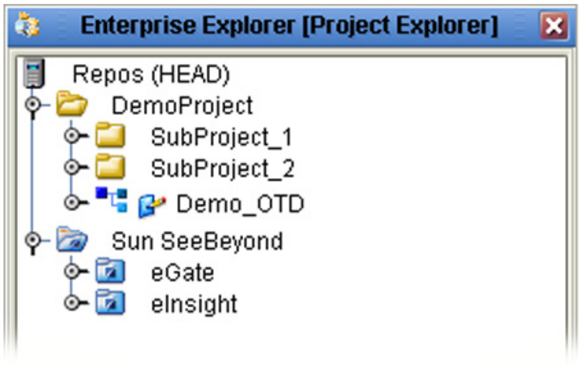
Extensive version control features have been built into Enterprise Designer for managing multiple versions of Projects and components. In addition to the global features described in this section, features specific to Deployment Profiles are described in **Version Control** on page 146.

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 33.

Figure 33 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 62). Refer to the *Sun SeeBeyond eGate™ Integrator System Administration Guide* for more information.

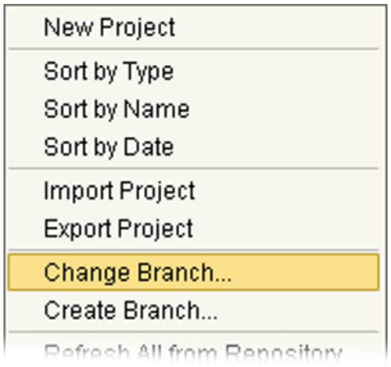
Changing Branches

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

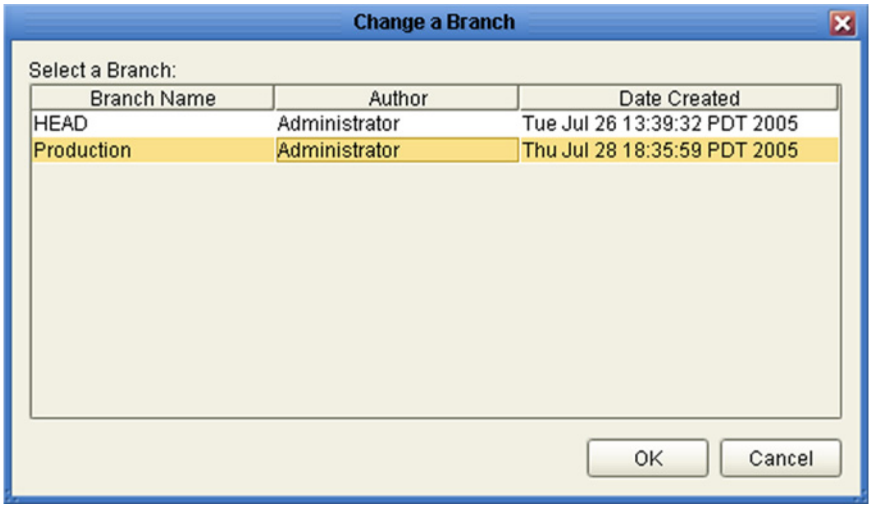
1 Display the Repository context menu (see Figure 34) by right-clicking the Repository in Project Explorer.

Figure 34 Repository Context Menu - Change Branch



2 Click the **Change Branch** option, which displays the *Change Branch* dialog (see Figure 35).

Figure 35 Change Branch Dialog



Select a Branch:

Branch Name	Author	Date Created
HEAD	Administrator	Tue Jul 26 13:39:32 PDT 2005
Production	Administrator	Thu Jul 28 18:35:59 PDT 2005

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

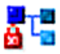
Version control allows you to maintain multiple versions of selected Project or Environment components within a particular branch of the Repository. The version history of each component is recorded to a log file, and can be viewed by means of a menu option (see **Viewing a Component's Version History** on page 87).

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 36 shows the lock, using the OTD icon as an example. See **Checking a Component In** on page 83, **Checking a Component In Without Revisions** on page 92, and **Checking In a Previous Version as the Latest Version** on page 93.

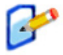
Figure 36 Checked In Icon (OTD Example)



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 37) 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 85.

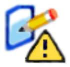
Figure 37 Checked Out Icon



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 38) 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 90 and **Checking In a Previous Version as the Latest Version** on page 93.

Figure 38 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 39.

Figure 39 Connectivity Map Context Menu

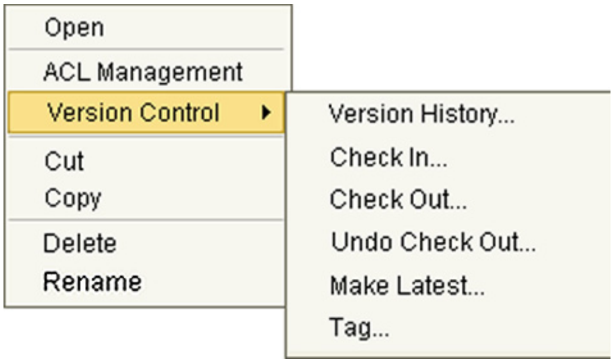


Table 14 Connectivity Map Menu Options - Version Control

Option	Function
Version History	Presents a dialog with which you can track the version history for the selected component. See **Viewing a Component's Version History** on page 87 for more information.
Check In	Presents a dialog, with which you can check in a new version of the selected component. See **Checking a Component In** on page 83 for more details.
Check Out	Presents a dialog with which you can check out the current version of the selected component. See **Checking a Component Out** on page 85 for more information.
Undo Check Out	Presents a dialog with which you can undo the check-out of the selected component. See **Checking a Component In Without Revisions** on page 92 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 **Checking In a Previous Version as the Latest Version** on page 93.
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.

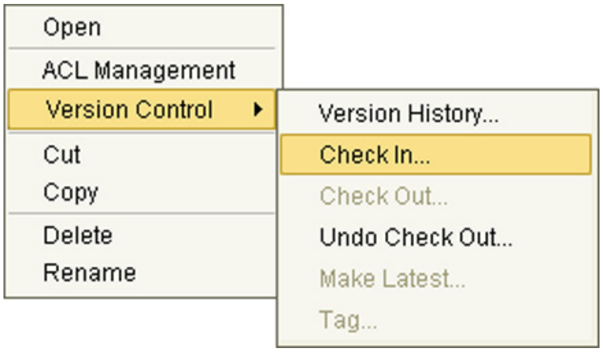
You can also check in multiple components by holding the **Ctrl** key while you select the components.

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.

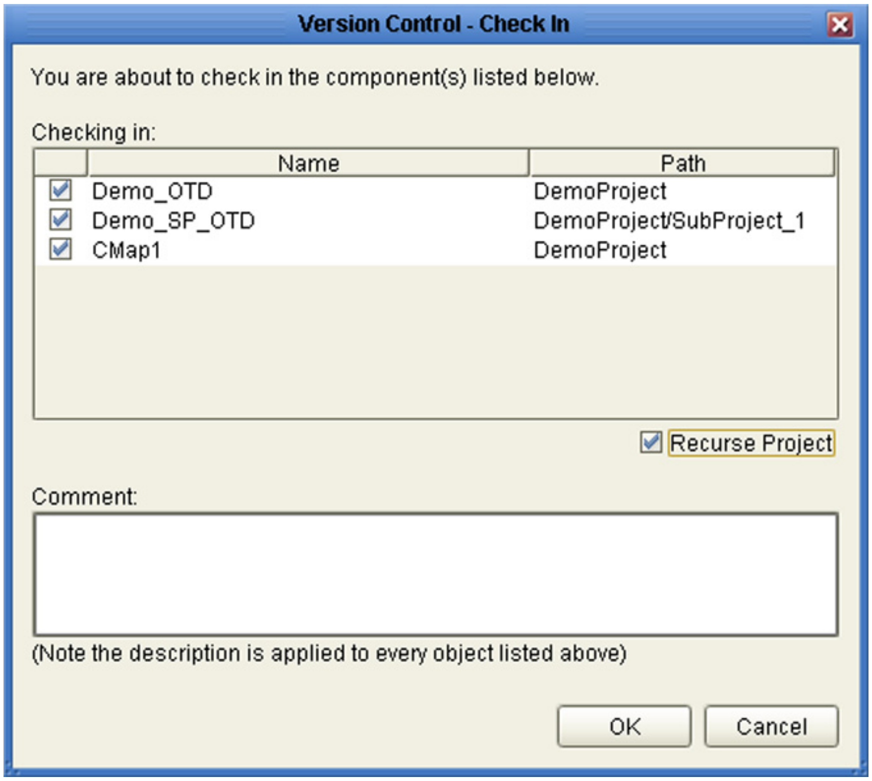
Figure 40 Example Context Menu - Check In Option



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



Version Control - Check In

You are about to check in the component(s) listed below.

Checking in:

	Name	Path
☑	Demo_OTD	DemoProject
☑	Demo_SP_OTD	DemoProject/SubProject_1
☑	CMap1	DemoProject

☑ Recurse Project

Comment:

(Note the description is applied to every object listed above)

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.

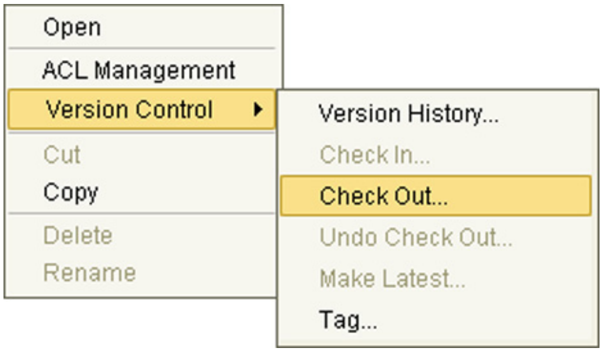
Note: *Although you can select multiple Projects (and they will appear highlighted in Project Explorer), only the components in the first Project you select will be shown in the dialog box. To check out components from multiple Projects at the same time, you must select the components themselves, not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

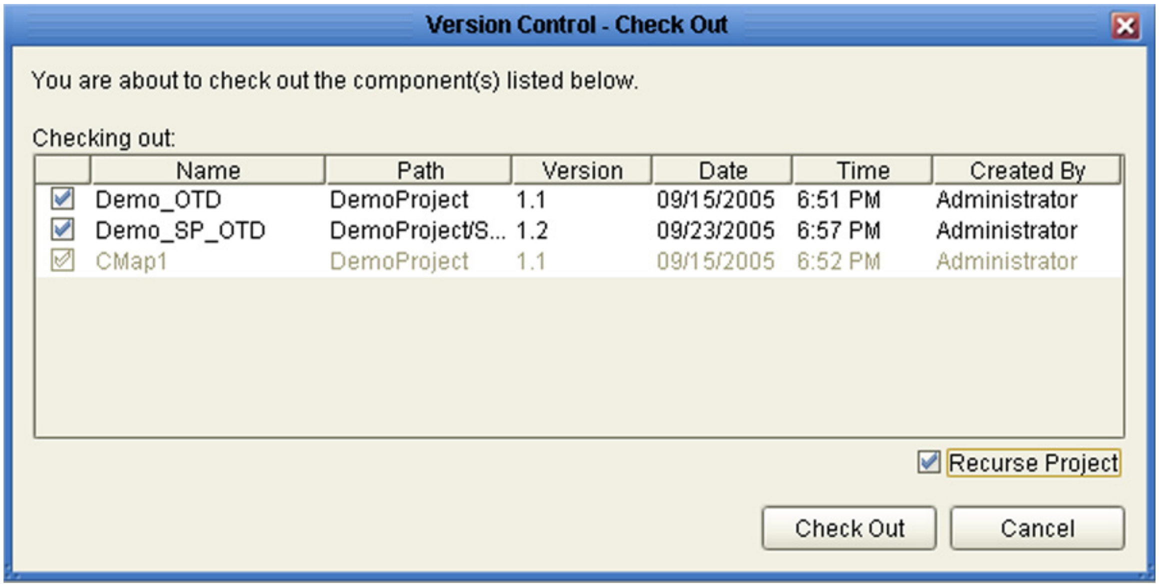
1 In the Enterprise Explorer, select the Project or component and right-click to display its context menu.

Figure 42 Example Context Menu - Check Out Option



2 Select **Version Control > Check Out**, as shown in Figure 42, to display the dialog shown in Figure 43.

Figure 43 Version Control - Check Out Dialog



You are about to check out the component(s) listed below.

Checking out:

	Name	Path	Version	Date	Time	Created By
☑	Demo_OTD	DemoProject	1.1	09/15/2005	6:51 PM	Administrator
☑	Demo_SP_OTD	DemoProject/S...	1.2	09/23/2005	6:57 PM	Administrator
☑	CMap1	DemoProject	1.1	09/15/2005	6:52 PM	Administrator

☑ Recurse Project

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.

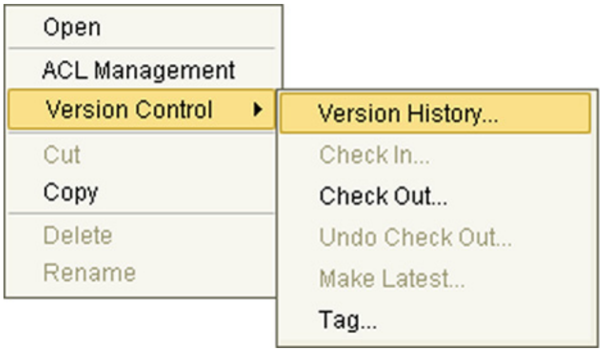
Viewing a Component's Version History

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

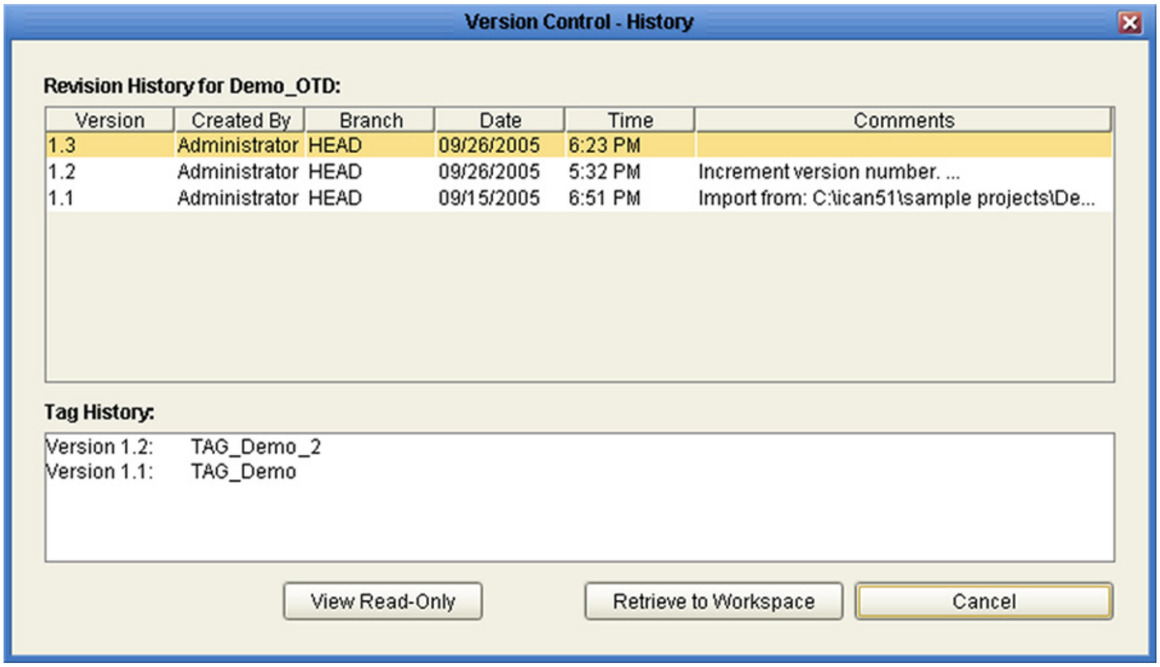
1. In the Enterprise Explorer, select the component and right-click to display its context menu.

Figure 44 Example Context Menu - Version History Option



2. Select **Version Control > Version History** to display the dialog shown in Figure 45. If the component has been tagged, the tags are shown in a separate *Tag History* pane.

Figure 45 Version Control - History Dialog



Revision History for Demo_OTD:

Version	Created By	Branch	Date	Time	Comments
1.3	Administrator	HEAD	09/26/2005	6:23 PM	
1.2	Administrator	HEAD	09/26/2005	5:32 PM	Increment version number. ...
1.1	Administrator	HEAD	09/15/2005	6:51 PM	Import from: C:\ican51\sample projects\De...

Tag History:

Version 1.2:	TAG_Demo_2
Version 1.1:	TAG_Demo

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

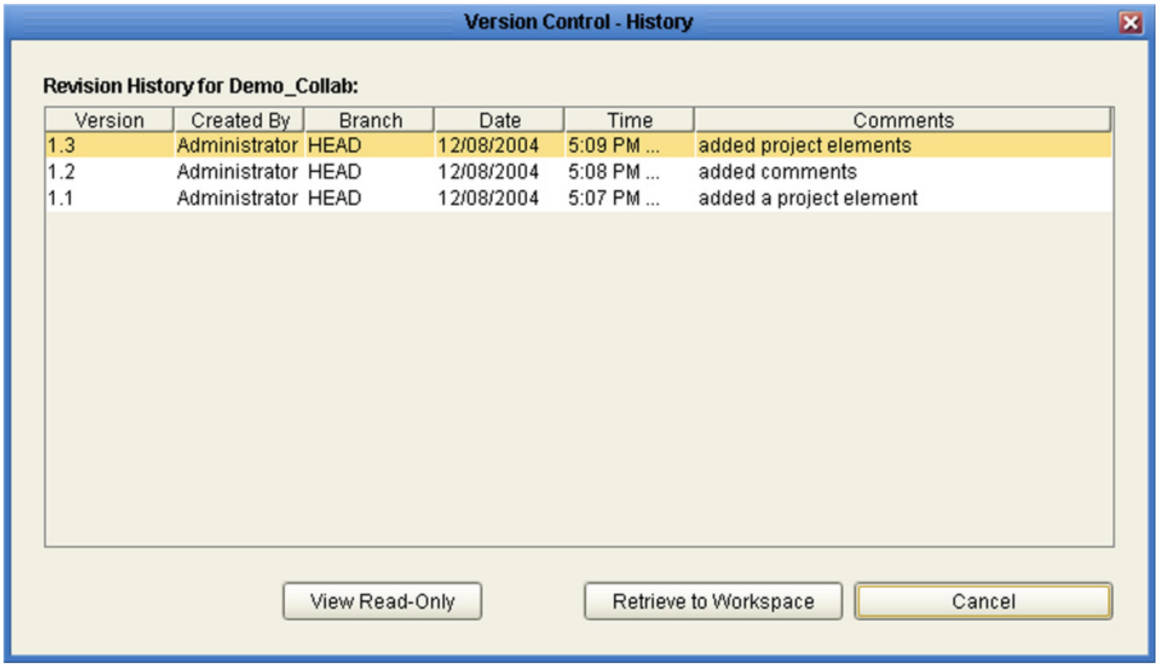
You can view a read-only instance of either the current or a previous version of a Java-based Collaboration or a User-Defined OTD by selecting the version from the Version History dialog box. The component must be checked in, and viewing as read-only does *not* lock the file from being checked out or 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 87) to display the dialog shown in Figure 47.

Figure 46 Version Control - History Dialog



Revision History for Demo_Collab:

Version	Created By	Branch	Date	Time	Comments
1.3	Administrator	HEAD	12/08/2004	5:09 PM ...	added project elements
1.2	Administrator	HEAD	12/08/2004	5:08 PM ...	added comments
1.1	Administrator	HEAD	12/08/2004	5:07 PM ...	added a project element

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

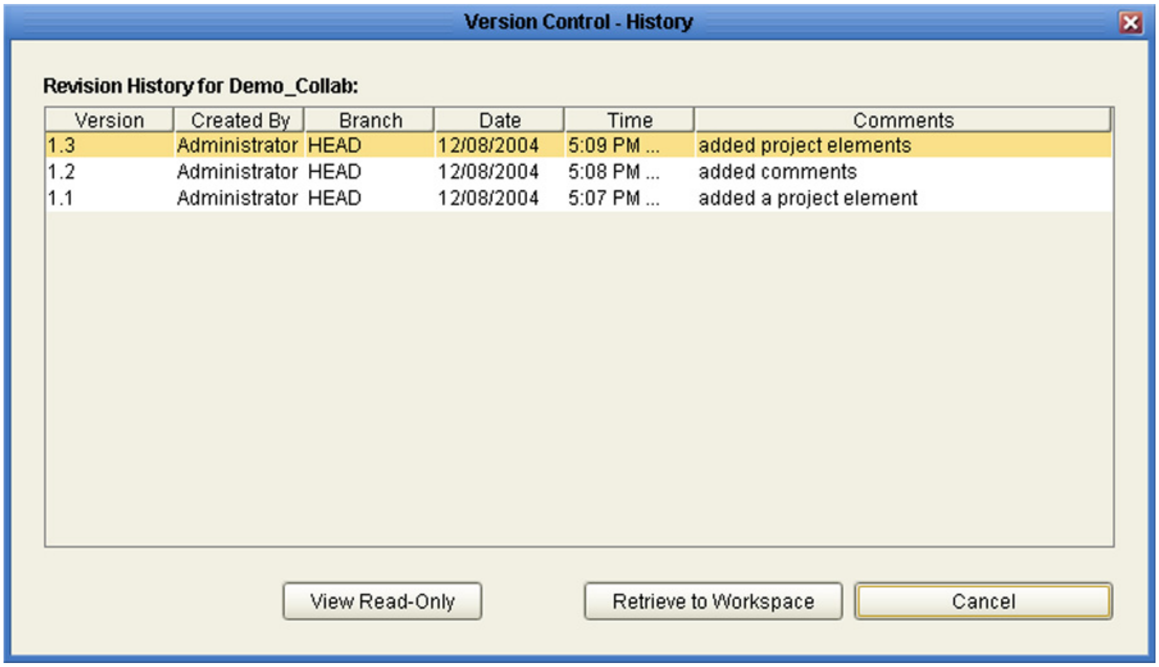
You can retrieve either the current or a previous version of any component by retrieving it from the Version History dialog box. Retrieving does *not* lock the file from being checked out or retrieved by other users for editing. To check a retrieved version back in as the latest version, you must use the **Make Latest** option described in **Checking In a Previous Version as the Latest Version** on page 93.

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 87) to display the dialog shown in Figure 47.

Figure 47 Version Control - History Dialog



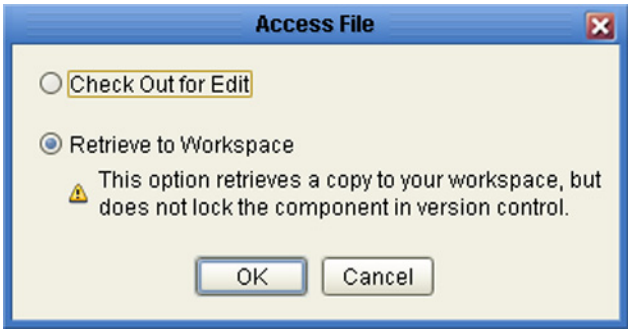
Revision History for Demo_Collab:

Version	Created By	Branch	Date	Time	Comments
1.3	Administrator	HEAD	12/08/2004	5:09 PM ...	added project elements
1.2	Administrator	HEAD	12/08/2004	5:08 PM ...	added comments
1.1	Administrator	HEAD	12/08/2004	5:07 PM ...	added a project element

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 48.

Figure 48 Access File Dialog

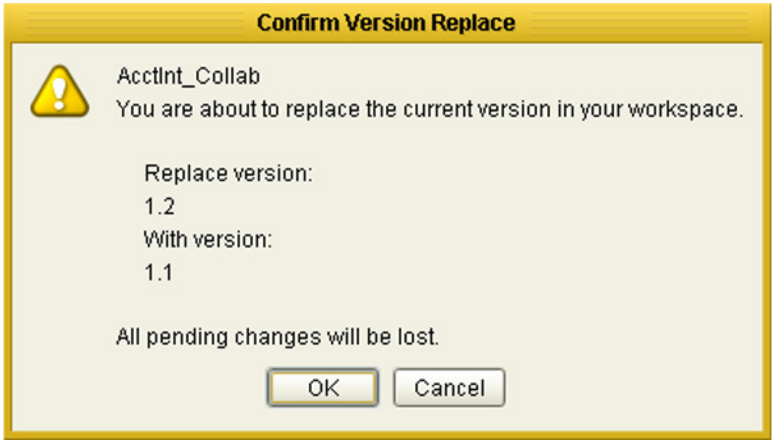


This option retrieves a copy to your workspace, but does not lock the component in version control.

- **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 85. 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 43).
- **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 49. 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 49 Confirm Version Replace Dialog



AcctInt_Collab
You are about to replace the current version in your workspace.

Replace version:
1.2
With version:
1.1

All pending changes will be lost.

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

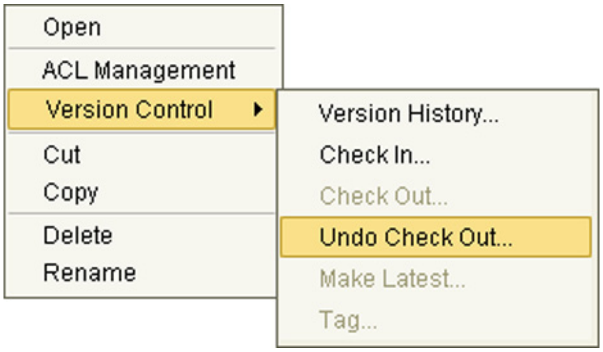
When you have checked out any version of a component and want to check it back in without revisions or incrementing the version number, you can simply cancel the check-out by using the following procedure. You can also check in multiple components by holding the **Ctrl** key while you select the components.

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

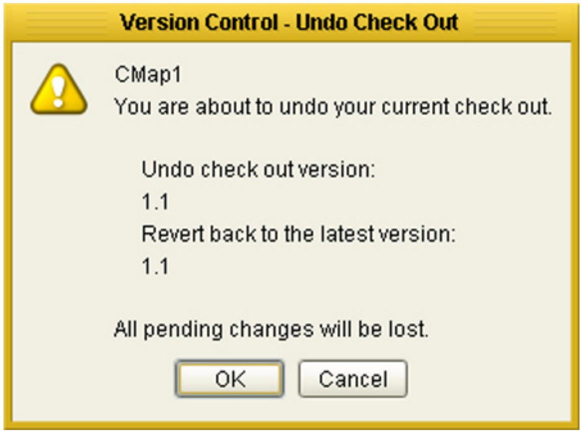
1 In the Enterprise Explorer, select the component and right-click to display its context menu.

Figure 50 Example Context Menu - Undo Check Out Option



2 Select **Version Control > Undo Check Out**, as shown in Figure 50, to display the *Version Control - Undo Check Out* dialog shown in Figure 51.

Figure 51 Version Control - Undo Check Out Dialog



Version Control - Undo Check Out

CMap1
You are about to undo your current check out.

Undo check out version:
1.1
Revert back to the latest version:
1.1

All pending changes will be lost.

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

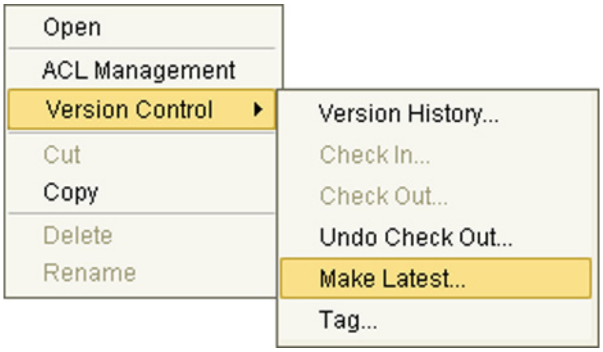
Checking In a Previous Version as the Latest Version

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.

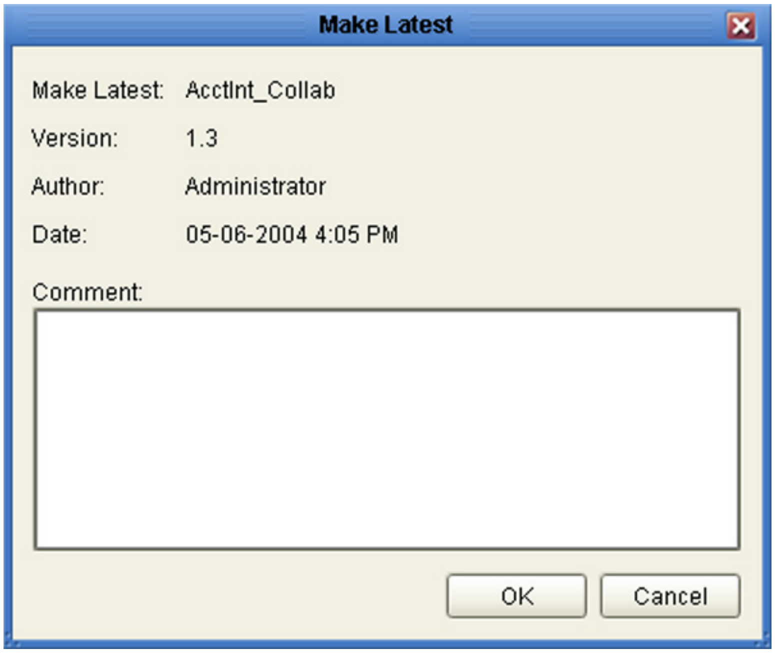
Figure 52 Example Context Menu - Make Latest Option



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 49.

Figure 53 Make Latest Dialog



Make Latest: AcctInt_Collab

Version: 1.3

Author: Administrator

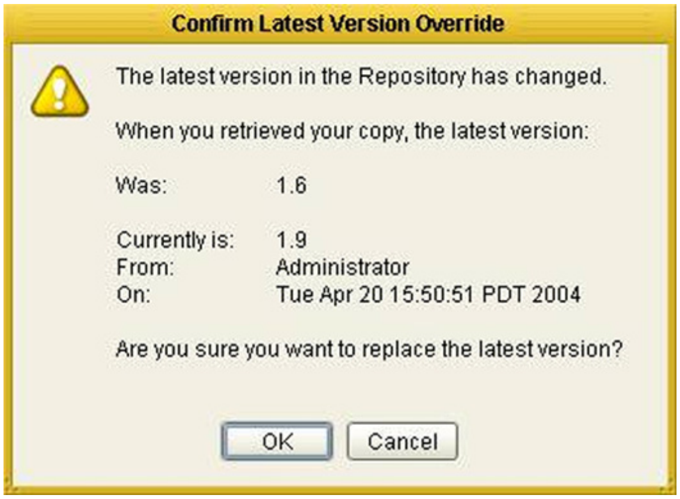
Date: 05-06-2004 4:05 PM

Comment:

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 54.

Figure 54 Confirm Latest Version Override Dialog



Confirm Latest Version Override

The latest version in the Repository has changed.

When you retrieved your copy, the latest version:

Was: 1.6

Currently is: 1.9
From: Administrator
On: Tue Apr 20 15:50:51 PDT 2004

Are you sure you want to replace the latest version?

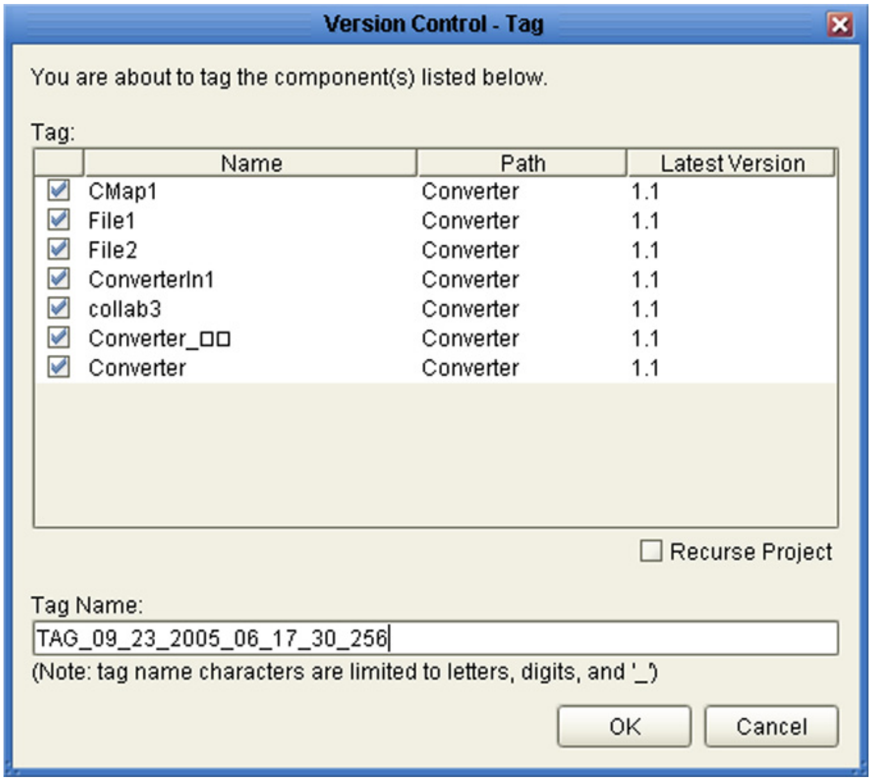
If you are sure you want to replace the current latest version, click **OK** to display the dialog shown previously (Figure 53). Type in a description of the changes in this version and click **OK**. The version in your workspace will be checked in as the latest version of the component.

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 55). You can then deselect individual components from the list.

Figure 55 Version Control Tag Dialog



You are about to tag the component(s) listed below.

Tag:

	Name	Path	Latest Version
☑	CMap1	Converter	1.1
☑	File1	Converter	1.1
☑	File2	Converter	1.1
☑	ConverterIn1	Converter	1.1
☑	collab3	Converter	1.1
☑	Converter_□□	Converter	1.1
☑	Converter	Converter	1.1

□ Recurse Project

Tag Name: TAG_09_23_2005_06_17_30_256

(Note: tag name characters are limited to letters, digits, and '_')

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.

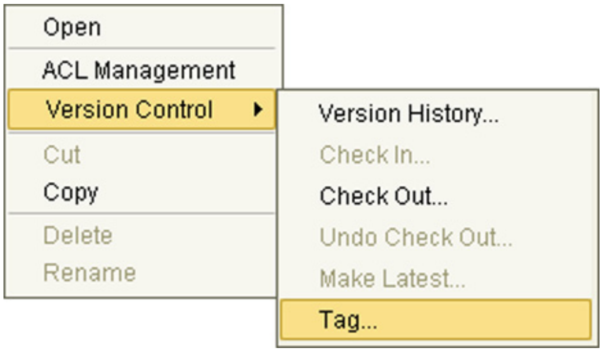
Note: *Previous versions of components can be tagged by retrieving the previous version to your workspace (see* **Retrieving a Component to Your Workspace** *on page 90).*

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.

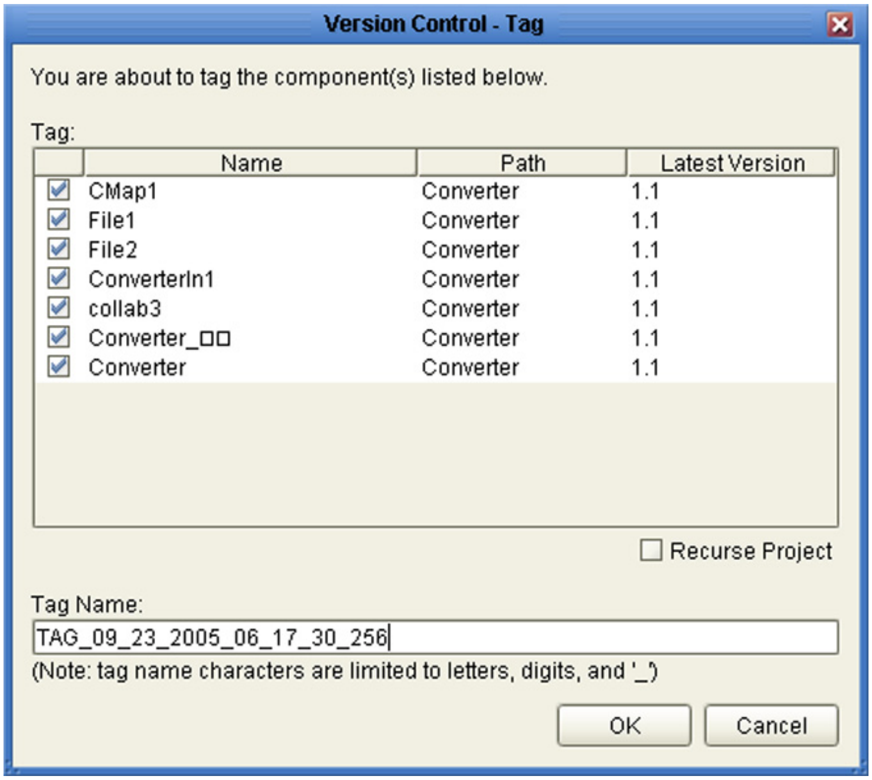
Figure 56 Example Context Menu - Tag Option



2 Click **Version Control** > **Tag** to display the dialog shown in Figure 57.

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 57 Version Control Tag Dialog


Version Control - Tag

You are about to tag the component(s) listed below.

Tag:

	Name	Path	Latest Version
☑	CMap1	Converter	1.1
☑	File1	Converter	1.1
☑	File2	Converter	1.1
☑	ConverterIn1	Converter	1.1
☑	collab3	Converter	1.1
☑	Converter_□□	Converter	1.1
☑	Converter	Converter	1.1

□ Recurse Project

Tag Name:

TAG_09_23_2005_06_17_30_256

(Note: tag name characters are limited to letters, digits, and '_')

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 140).

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 59).

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 58, to display the dialog shown in Figure 59.

Figure 58 Collaboration Definition Context Menu - Create Diff Option

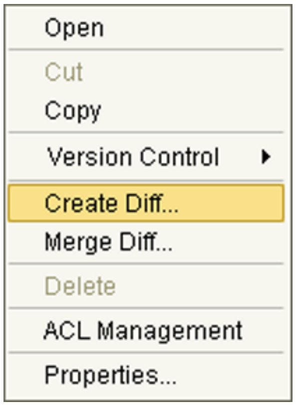
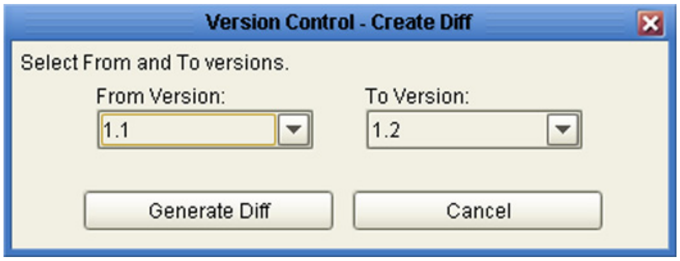


Figure 59 Version Control - Create Diff Dialog


Version Control - Create Diff

Select From and To versions.

From Version: 1.1

To Version: 1.2

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).

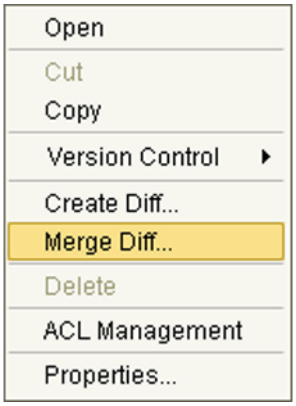
6 Click **Save** to save the file.

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

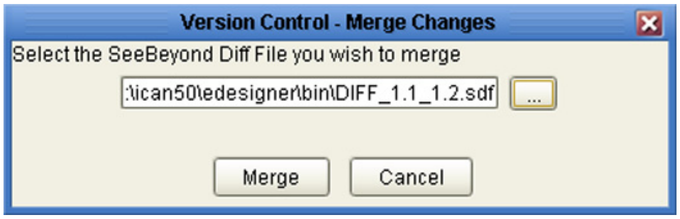
1 In Project Explorer, right-click the desired Collaboration Definition to display its context menu.

Figure 60 Collaboration Definition Context Menu - Merge Diff Option



2 Select **Merge Diff** (see Figure 60), to display the dialog shown in Figure 61.

Figure 61 Version Control - Merge Changes Dialog



Select the SeeBeyond Diff File you wish to merge

:\ican50\edesigner\bin\DIFF_1.1_1.2.sdf

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 61.

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.

Note: *You cannot cut-and-paste or copy-and-paste between branches. You can export from one branch and import to another to accomplish the same functionality.*

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:

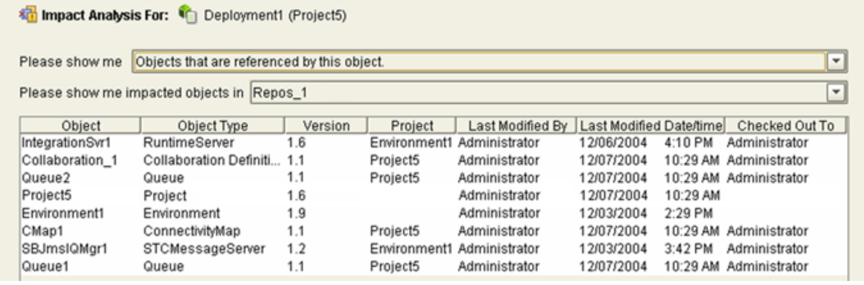
- Objects that have references to the selected component.
- Objects that are referenced by the selected component.

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 62 and Figure 63).

Figure 62 Objects Referenced by Deployment Profile



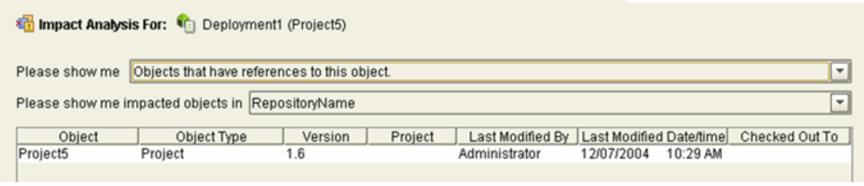
Impact Analysis For: Deployment1 (Project5)

Please show me: Objects that are referenced by this object.

Please show me impacted objects in: Repos_1

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
SBJmsIQMgr1	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 63 Objects Referencing Deployment Profile



Impact Analysis For: Deployment1 (Project5)

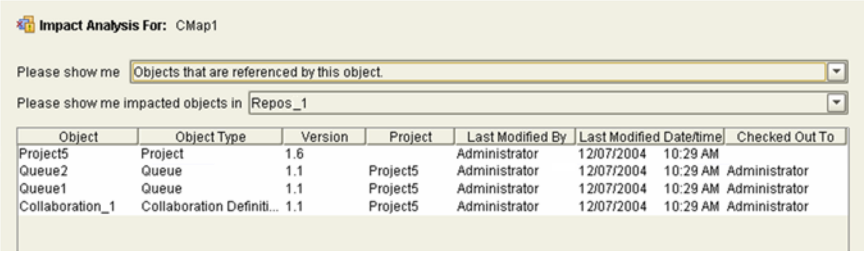
Please show me: Objects that have references to this object.

Please show me impacted objects in: RepositoryName

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 64 and Figure 65).

Figure 64 Objects Referenced by Connectivity Map



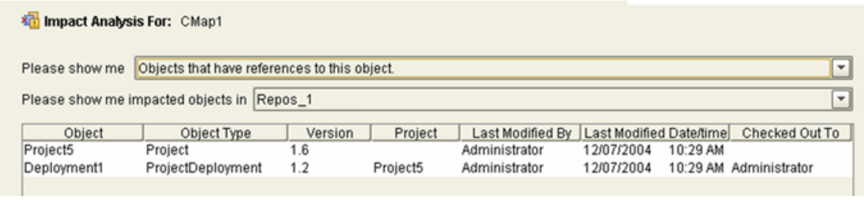
Impact Analysis For: CMap1

Please show me: Objects that are referenced by this object.

Please show me impacted objects in: Repos_1

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 65 Objects Referencing Connectivity Map



Impact Analysis For: CMap1

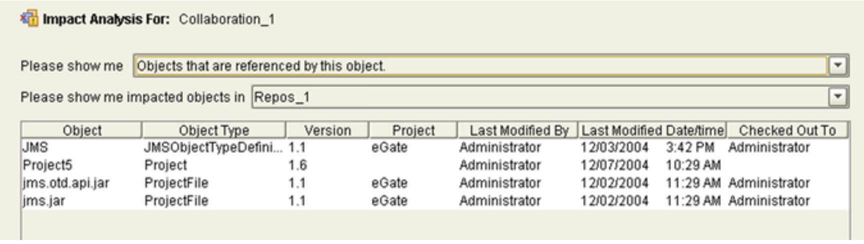
Please show me: Objects that have references to this object.

Please show me impacted objects in: Repos_1

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 66 and Figure 67).

Figure 66 Objects Referenced by Collaboration Definition



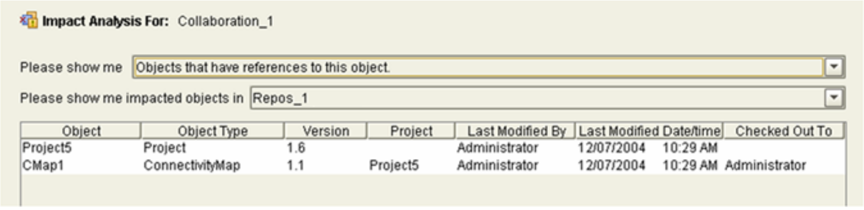
Impact Analysis For: Collaboration_1

Please show me: Objects that are referenced by this object.

Please show me impacted objects in: Repos_1

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 67 Objects Referencing Collaboration Definition



Impact Analysis For: Collaboration_1

Please show me: Objects that have references to this object.

Please show me impacted objects in: Repos_1

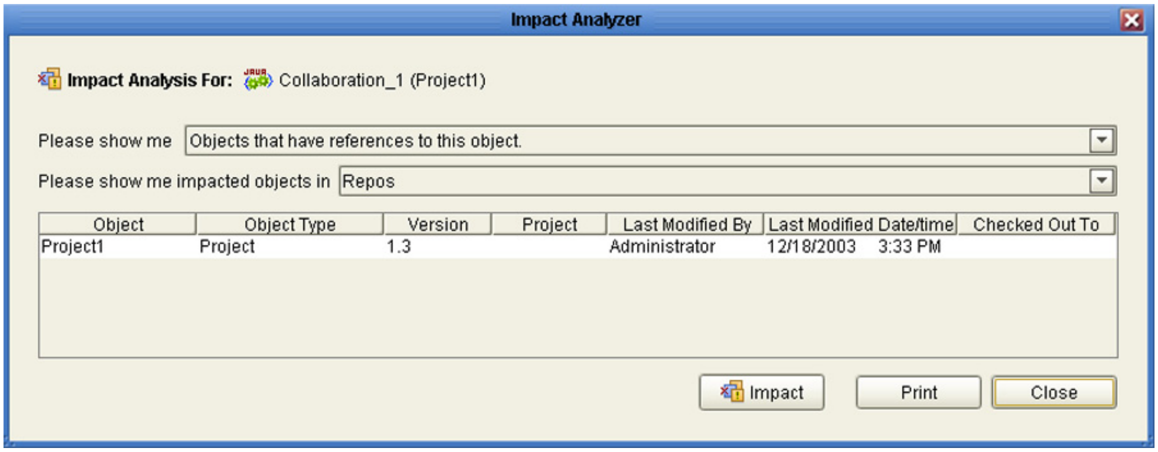
Object	Object Type	Version	Project	Last Modified By	Last Modified Date/time	Checked Out To
Project5	Project	1.6		Administrator	12/07/2004 10:29 AM	
CMap1	ConnectivityMap	1.1	Project5	Administrator	12/07/2004 10:29 AM	Administrator

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 68.
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 68 Impact Analyzer Dialog



Impact Analyzer

Impact Analysis For: Collaboration_1 (Project1)

Please show me: Objects that have references to this object.

Please show me impacted objects in: Repos

Object	Object Type	Version	Project	Last Modified By	Last Modified Date/time	Checked Out To
Project1	Project	1.3		Administrator	12/18/2003 3:33 PM	

Table 15 Impact Analyzer Command Buttons

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

Chapter 5

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 105
- **About Project Explorer** on page 107
- **Project Components** on page 109
- **Creating a Project** on page 115
- **Services** on page 117
- **Message Destinations** on page 119
- **External Applications** on page 121
- **Web Service External Applications** on page 122
- **Schedulers** on page 123
- **Constants and Variables** on page 125
- **Connectivity Maps** on page 129
- **Deployment Profiles** on page 138

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.

In a Java CAPS Project, you specify the business logic for the eGate implementation by defining the following items:

- **External applications** — the beginning and end points of the Project.

- **Collaboration Definitions** — the logical transformations to be performed on the data.
- **Message destinations** (topics or queues) — internal storage and routing nodes.
- **JMS client properties** — message connection and propagation options.

For each of these components you specify logical properties — these properties are independent from the physical implementation. Projects are where you add and name message destinations, by dragging and dropping topics and queue icons onto the Connectivity Map canvas.

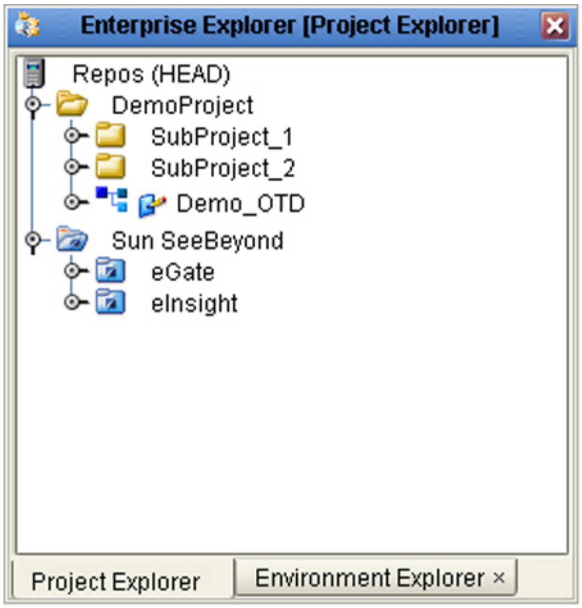
On links between message destinations and their subscribers and publishers, there is a JMS client properties icon. By double-clicking the JMS client properties icon in the Connectivity Map, you can configure the connection for such items as persistent or non-persistent delivery mode, XA, and concurrent processing.

After having added the components for message destinations, you create the Object Type Definitions (OTDs) and Collaboration Definitions. After defining the Collaboration Definitions, you then create the relationships between the components by associating Collaboration Definitions with services. This defines a service as a Collaboration, and creates a link between the Collaboration and it's Collaboration Definition.

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 69). It is used in conjunction with the various editors to create and configure the components of a Project.

Figure 69 Project Explorer



Each component in Project Explorer has an icon to identify the component type (see **Project Explorer Icons** on page 108). 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 109
- **Using Project Component Context Menus** on page 113

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 **Repository**, which is the central database where all Project information is saved. Binary files required at run time are also stored here.
	Represents a **Project** or Subproject.
	Represents a set of templates you can import to your Project.
	Represents a **Connectivity Map**, which contains the business logic and information about the data transmission.
	Represents a **Project variable** or **constant**.
	Represents an **Object Type Definition** (OTD) file.
	Represents a **Collaboration Definition (Java)** file.
	Represents a **Collaboration Definition (XSLT)** file.
	Represents a **Deployment Profile**, 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 Managing Component Versions on page 81 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 Managing Component Versions on page 81 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 62). 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 70. Additional menu options may be shown, depending upon what additional products you have installed.

Figure 70 Project Context Menu

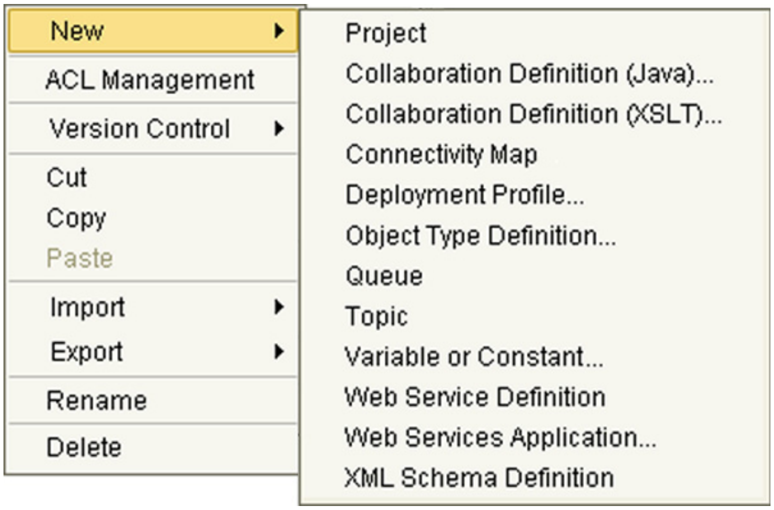


Table 17 Project Context Menu Options

Option		Function
New	Project	Adds a Subproject folder to the selected Project. **Note:** You should limit the total number of Project/ Subproject levels to **5**.
	Collaboration Definition (Java)	Presents the Collaboration Definition Wizard (Java), with which you can create a Java-based Collaboration Definition. See**Creating a Java-based Collaboration Definition** on page 326.
	Collaboration Definition (XSLT)	Presents the Collaboration Definition Wizard (XSLT), with which you can create an XSLT-based Collaboration Definition. See**Creating an XSLT-based Collaboration Definition** on page 390.
	Connectivity Map	Adds a Connectivity Map to the Project. See **Connectivity Maps** on page 129.

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 **Using the Deployment Editor** on page 138.
	Object Type Definition	Presents the OTD Wizard, with which you can create an Object Type Definition (OTD) file. See **Object Type Definitions II** on page 181 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 **Creating Web Service Definitions** on page 475)
	Web Services Application	Adds a Web Services Application to the selected Project. See **Web Service External Applications** on page 122.
	XML Schema Definition	Adds an XML Schema Definition (XSD) to your Project and opens the XSD Editor. (see **Creating XML Schema Definitions** on page 427)
ACL Management		Presents the ACL Properties dialog, with which an Administrator can assign read/write privileges to users for the selected Project. See the *Sun SeeBeyond eGate™ Integrator System Administration Guide*.
Version Control	Check In	Presents a dialog with which you can check in a new version of the selected Project. Refer to **Checking a Component In** on page 83 for more details.
	Check Out	Presents a dialog with which you can check out the current version of the selected Project. See **Checking a Component Out** on page 85 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 within the same branch (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 *Note* following this table, and **Moving and Replicating Repository Objects** on page 100 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 within the same branch (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 *Note* following this table, and **Moving and Replicating Repository Objects** on page 100 for additional information.
Paste		Pastes a cut or copied component into the selected Project within the same branch (Projects are pasted as Subprojects). Pasting is disabled for other users when you have the Project checked out. See the *Note* following this table, and **Moving and Replicating Repository Objects** on page 100 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 **Importing a Project or Environment** on page 70.
	Web Service Definition	Presents a dialog with which you can import existing Web Service Definitions for defining a web service. See **Importing Existing Web Service Definitions** on page 499.
	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 **Exporting a Project or Environment** on page 65.
Rename		Activates the field, allowing you to rename the selected Project.

Table 17 Project Context Menu Options

Option	Function
Delete	Deletes the selected Project, subject to the following conditions: ▪ You have *write* privileges for the Project (see *ACL Management*, above). ▪ The Project is not checked out by anyone other than yourself. If these conditions are true, a dialog is displayed in which you confirm that you want to delete the selected Project. Clicking **Yes** then deletes the Project. 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 **OK** button.

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 71. Those context menus that are different are described in the sections for the specific components.

Figure 71 Project Component Context Menu

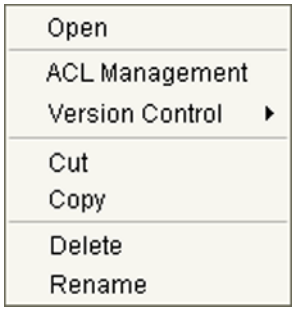


Table 18 Project Component Context Menu Options

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 *Sun SeeBeyond eGate™ Integrator System Administration Guide*.
Version Control	Accesses the version control features for the selected component. See **Accessing Component Version Control Features** on page 82 for additional information.
Cut	Copies the selected component and removes it from the current Project, after which you can paste it to another Project within the same branch (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 **Moving and Replicating Repository Objects** on page 100 for additional information.
Copy	Copies the selected component, after which you can paste it to other Projects within the same branch (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 **Moving and Replicating Repository Objects** on page 100 for additional information.

Table 18 Project Component Context Menu Options

Option	Function
Delete	Deletes the selected component, subject to the following conditions: ▪ You have *write* privileges for the component (see ACL Management, above). ▪ You have the component checked out to your workspace. ▪ The component is not checked out by anyone other than yourself. If these conditions are met, a dialog is presented in which you confirm that you want to delete the selected component. Clicking **Yes** then deletes the component.
Rename	Activates the field, allowing you to rename the selected component.

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.*

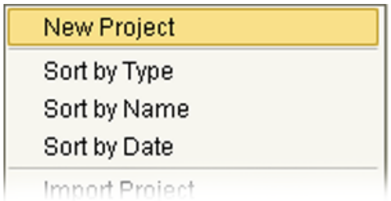
5.3.3 Creating a Project

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 72).

Figure 72 Repository Context Menu - New Project

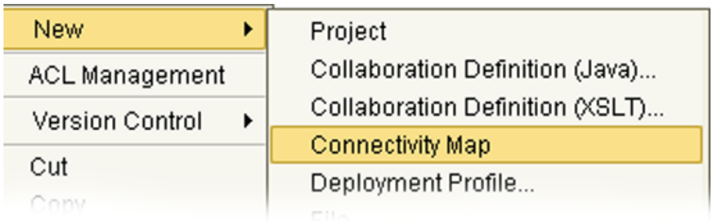


2 Select a **New Project**, which initiates a hierarchical structure under the Repository.

3 Rename the Project as desired.

4 Right-click the **Project** to display its context menu (see Figure 73).

Figure 73 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 129), 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 181.

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 252 or **Collaboration Definitions (XSLT)** on page 387.

10 Return to the Connectivity Map and connect the various components (see **Connectivity Maps** on page 129).

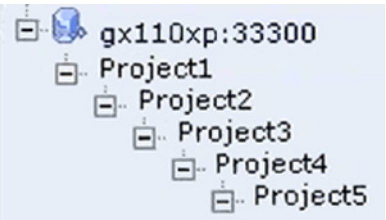
11 Save and test your Project.

Once you have created your Project, you need to create a run-time Environment (see **Run-time Environments** on page 516) to represent your physical system, and then map your Project to the Environment. See **Deployment Profiles** on page 138.

5.3.4 Project Nesting

Projects can contain Subprojects, which in turn can contain other Subprojects (see Figure 74). Primarily because of limitations in the user interface, the working limit for nesting Projects within other Projects is five levels (total), as shown in the figure.

Figure 74 Project Nesting



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

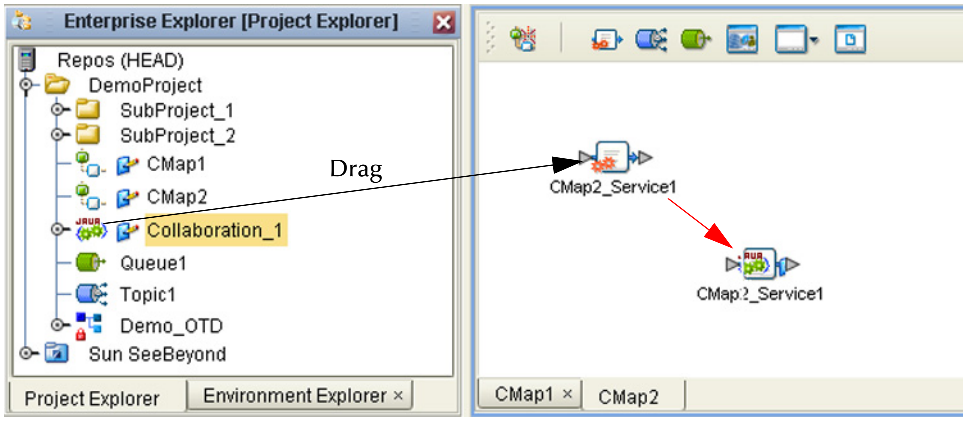
A business process is a collection of actions that take place in your company, revolving around a specific business practice. These business processes are modeled in eInsight Business Process Manager, and then embodied as a Project service in eGate Integrator. See the *Sun SeeBeyond eInsight™ Business 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 132. 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 75). The icon changes as shown in the figure to indicate its new identity.

Figure 75 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.

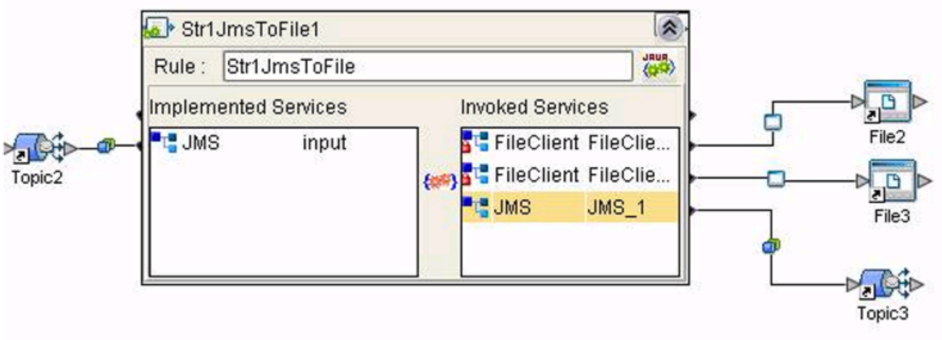
Linking Services and Destinations

After you have bound a Collaboration Definition to a service to create a Collaboration as described in **Services** on page 117, you must link the Collaboration's services to the inbound and outbound destinations.

To link services and destinations

1. Double-click the Collaboration to display its binding box (see Figure 76).
2. Drag the output port of the source component to the **Implemented Services** pane.
3. Drag the services from the **Invoked Services** pane to their appropriate destinations
4. Close the Collaboration binding box and **Save**.

Figure 76 Linking Multiple Destinations



When there are multiple destinations, as in the example, 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.

Connection-related properties for the Collaboration (or other service) are configured in the adjoining JMS Client (see **Configuring JMS Clients** on page 133).

5.5 Message Destinations

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

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

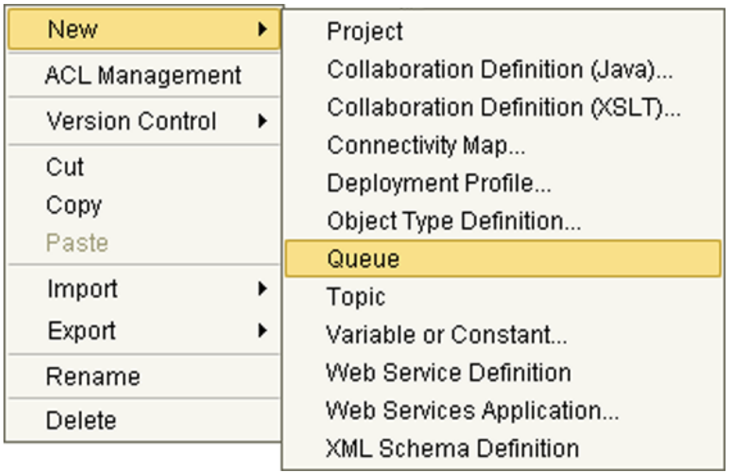
Figure 77 Topic Icon

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

Figure 78 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 79). You can also drag a Queue or Topic icon (see Figure 77 and Figure 78) 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 129).

Figure 79 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 113).

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:

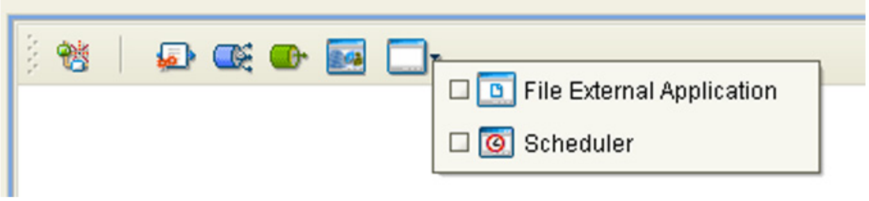
<   **>**   **&** $ %

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 80.

Figure 80 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 113). See the individual eWay user guides for additional information.

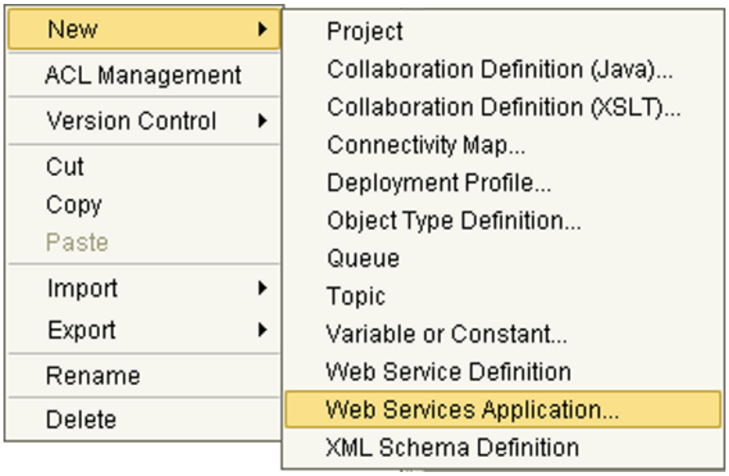
5.7 Web Service External Applications

A *Web Service External Application* is the logical representation of a web service that appears in a Project (see **Developing Web Services** on page 422).

Figure 81 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 82). You can also drag a Web Service External Application icon (see Figure 81) 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 129).

Figure 82 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 536.

Right-clicking the Web Service External Application in Project Explorer displays its context menu (see **Using Project Component Context Menus** on page 113).

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 83) 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 129).s.

Figure 83 Scheduler Icon

Right-clicking the Scheduler in Project Explorer displays its context menu (see **Using Project Component Context Menus** on page 113).

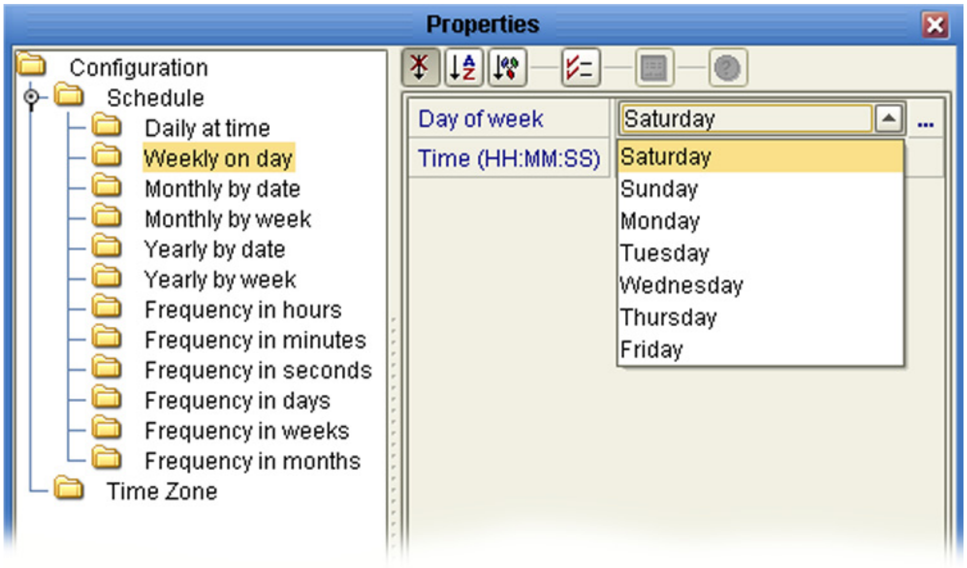
5.8.1 Configuring a Scheduler

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 84).

Figure 84 Scheduler Properties Dialog - Schedule



Properties

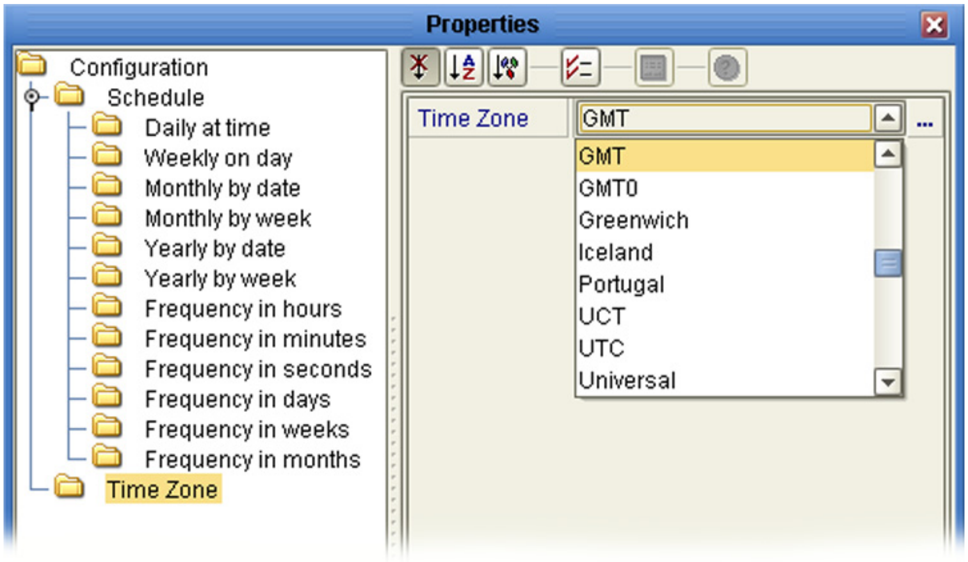
Day of week: Saturday

Time (HH:MM:SS)

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 85).

Figure 85 Scheduler Properties Dialog - Time Zone



Time Zone: GMT

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

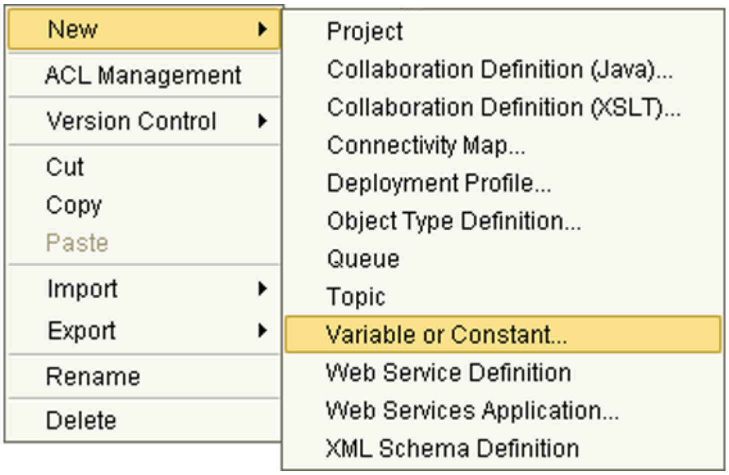
The following special characters can be used in scheduler names:

< **>** **&**

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 86).

Figure 86 Project Context Menu - New Variable or Constant

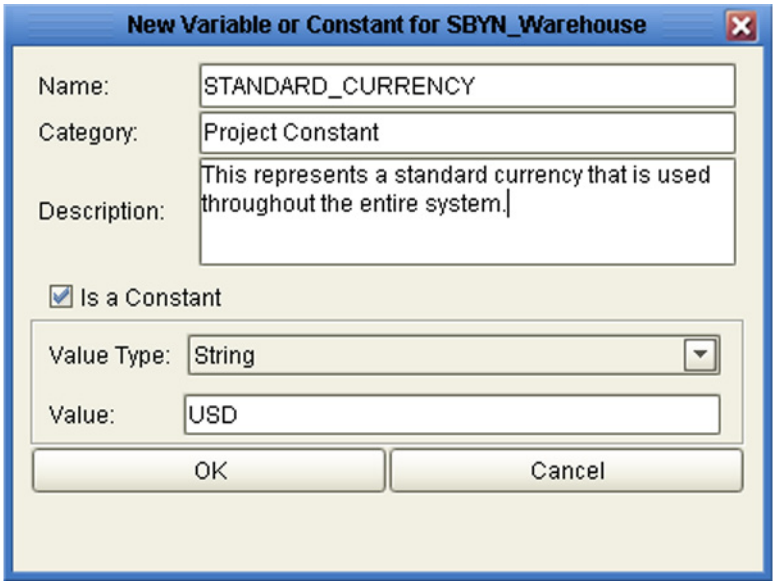


Right-clicking the constant or variable in Project Explorer displays its context menu (see **Using Project Component Context Menus** on page 113).

5.9.1 Project Constants

Constants are name-value pairs that are visible across the Project. For example, Figure 87 shows a standard currency defined to be used globally throughout the system.

Figure 87 Creating a Project Constant



New Variable or Constant for SBYN_Warehouse

Name: STANDARD_CURRENCY

Category: Project Constant

Description: This represents a standard currency that is used throughout the entire system.

☑ Is a Constant

Value Type: String

Value: USD

Table 19 Project Variable Options

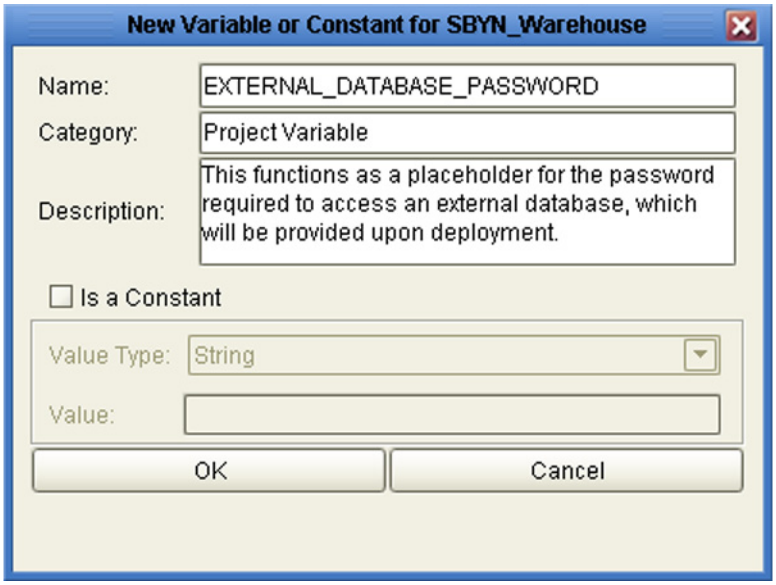
Property	Option	Description/Usage
Name		Your name for the Project constant.
Category		You may assign a category name, if desired.
Description		Your description for the constant.
Is a Constant		Select to enable the constant properties fields.
Value Type	String	Allows the string value to be displayed explicitly.
	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 145). Project variables can be literals or Environmental constants (see **Environmental Constants** on page 523).

As an example, Figure 88 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 88 Creating a Project Variable



New Variable or Constant for SBYN_Warehouse

Name: EXTERNAL_DATABASE_PASSWORD

Category: Project Variable

Description: This functions as a placeholder for the password required to access an external database, which will be provided upon deployment.

□ Is a Constant

Value Type: String

Value:

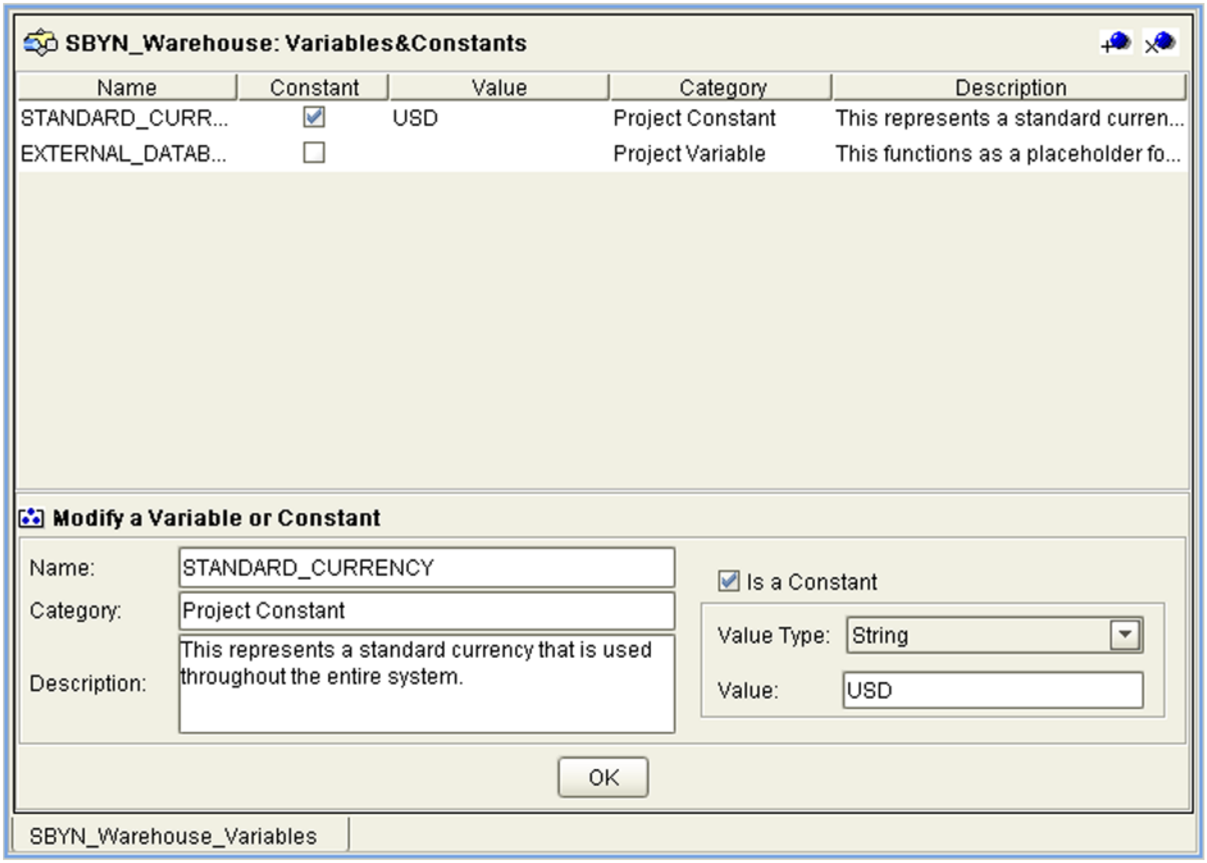
Table 20 Project Variable Options

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 89). 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 89 Variables and Constants Object Group



SBYN_Warehouse: Variables&Constants

Name	Constant	Value	Category	Description
STANDARD_CURR...	☑	USD	Project Constant	This represents a standard curren...
EXTERNAL_DATAB...	□		Project Variable	This functions as a placeholder fo...

Modify a Variable or Constant

Name: STANDARD_CURRENCY

Category: Project Constant

Description: This represents a standard currency that is used throughout the entire system.

☑ Is a Constant

Value Type: String

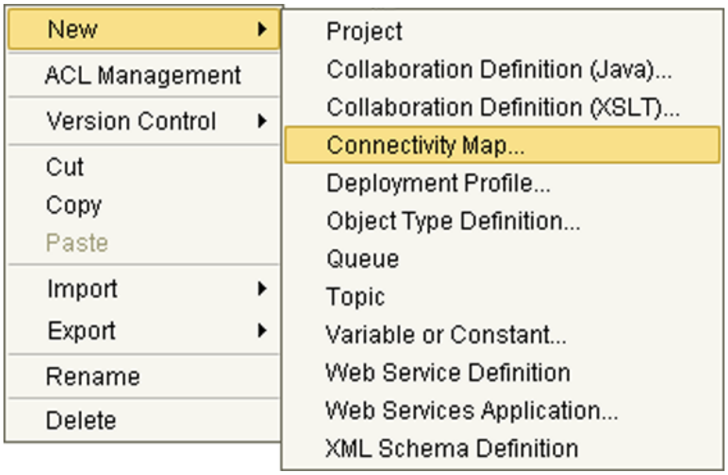
Value: USD

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 90).

Figure 90 Project Context Menu - New Connectivity Map



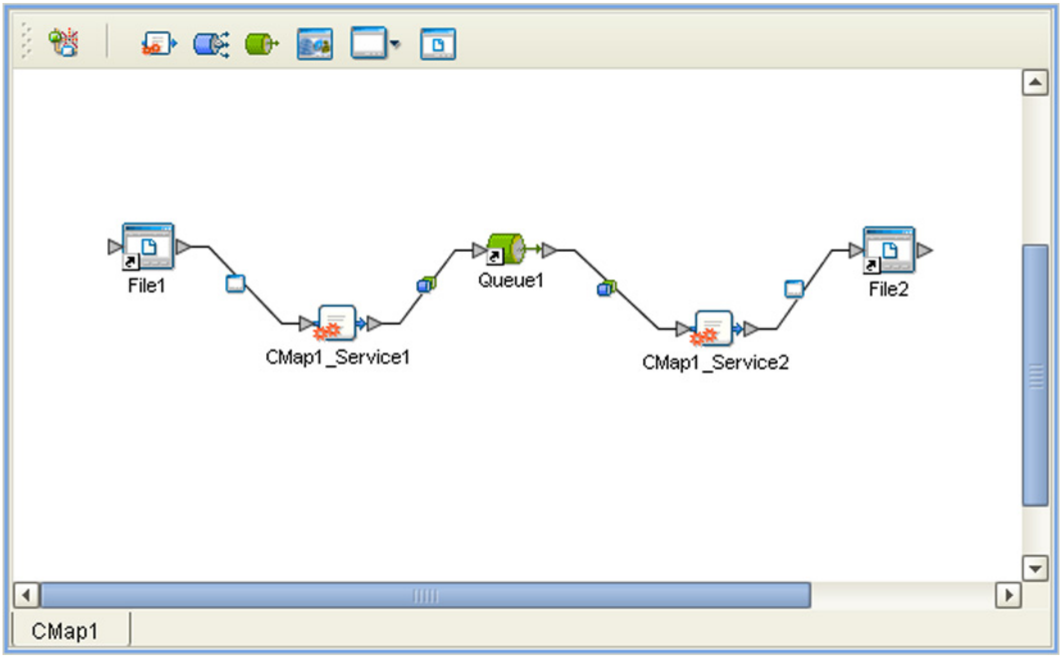
Right-clicking the connectivity map in Project Explorer displays its context menu (see **Using Project Component Context Menus** on page 113).

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 91). 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 91 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 132).

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 **Services** on page 117.
	Queue	A Message Destination that conforms to the point-to-point messaging paradigm, having one sender and one receiver. See **Message Destinations** on page 119.
	Topic	A Message Destination that conforms to the publish/subscribe messaging paradigm, having one sender (publisher) and multiple receivers (subscribers). See **Message Destinations** on page 119.
	Web Service External Application	Represents a Web Service External Application (see **Web Service External Applications** on page 122).
	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 **External Applications** on page 121.
	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 **Schedulers** on page 123.

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.

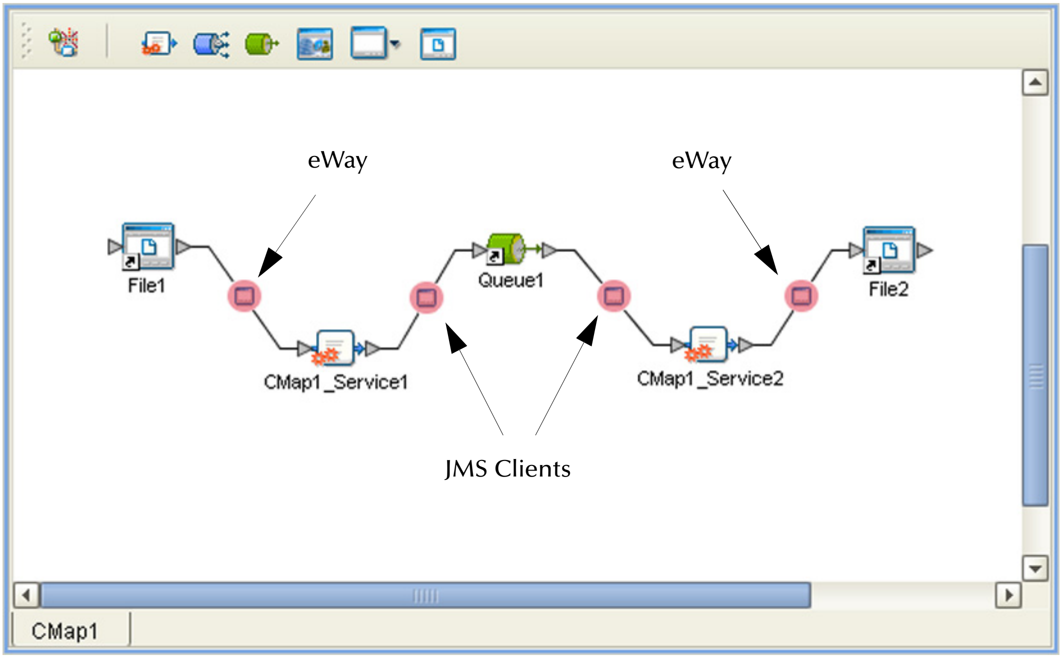
Also, renaming or deleting a queue or topic in the Repository will not affect the existence or name of the associated placeholder in the Connectivity Map. The change will, however, be reflected in the *tooltips* for the placeholder. This allows you to re-assign the placeholder without disrupting the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

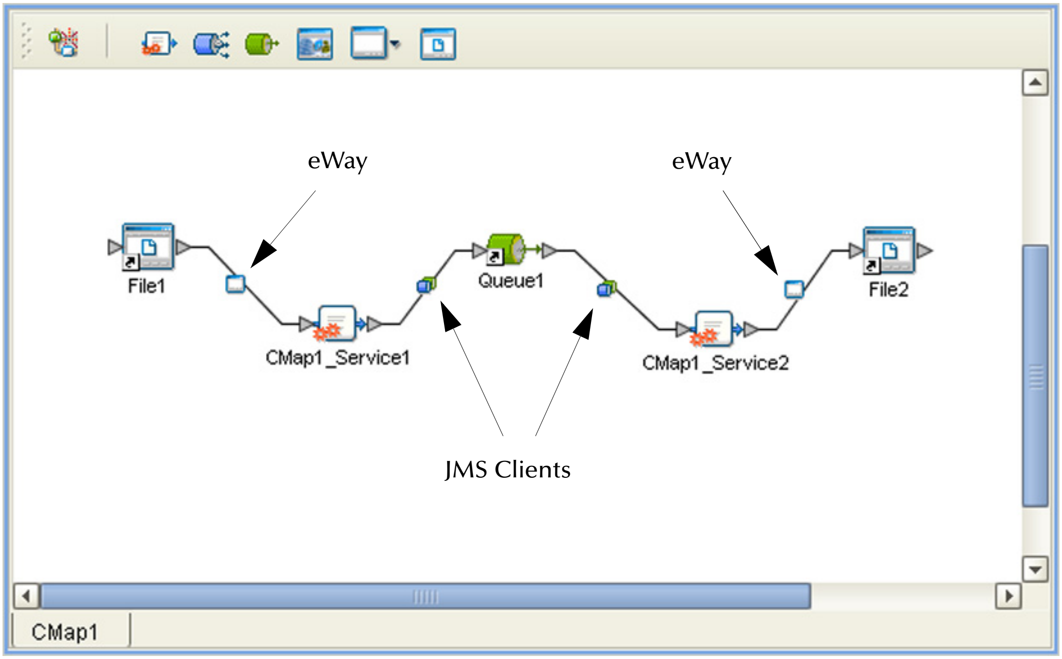
- When you link an external application with a Collaboration, Enterprise Designer automatically adds an eWay adapter to the link. The eWay enables communication and movement of data between the external application and the eGate Integrator system. The eWay configuration specifies the logical connection 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 504 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 **Configuring JMS Clients** on page 133).

Figure 92 Connection Icons in a Connectivity Map - Not Configured



Initially, these connectors are shown as red disks, as illustrated in Figure 92. 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 93.

Figure 93 Connection Icons in a Connectivity Map - Configured



5.10.4 Configuring JMS Clients

JMS clients are of two basic types: *producers* and *consumers* (or a combination of both). If associated with a queue, these become queue *senders* and *receivers*, respectively. If associated with a topic, they become topic *publishers* and *subscribers*, respectively. This section provides a summary of the JMS client configuration properties — see the *Sun SeeBeyond eGate™ Integrator JMS Reference Guide* for additional information.

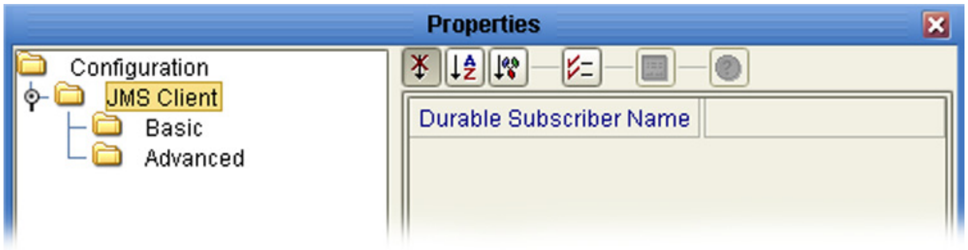
Double-clicking the JMS client icon (see Figure 92) displays the dialog shown in Figure 94.

Note: *To reconfigure an existing JMS client, redeploy the Deployment Profile.*

Root Properties

Properties appear in the root dialog only if the client is a topic subscriber.

Figure 94 JMS Client Configuration Properties Dialog (Subscriber)



Durable Subscriber Name

Table 22 JMS Client Configuration Properties

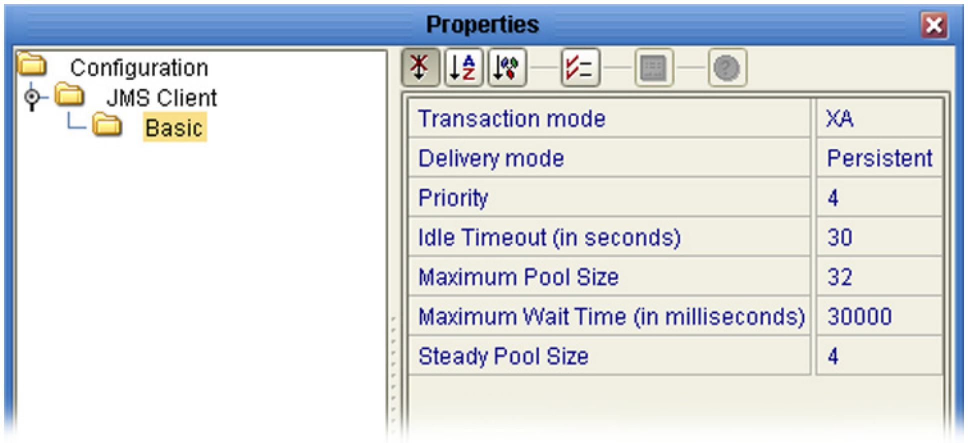
Property	Applies to	Description
Durable Subscriber Name	Topic subscribers only	Specifies the durable subscriber name.

Note: *Once set, the Durable Subscriber Name does not get autogenerated, and can only be changed manually. Copies of the Connectivity Map will also retain this name.*

Basic Properties

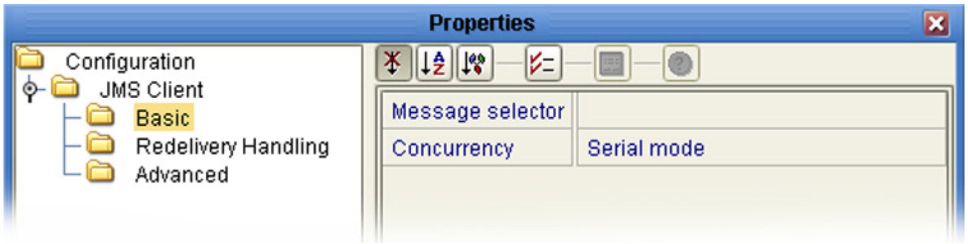
Which properties appear in the dialog depends upon whether the client is a producer or consumer.

Figure 95 JMS Client Basic Configuration Properties Dialog (Producer)



Transaction mode	XA
Delivery mode	Persistent
Priority	4
Idle Timeout (in seconds)	30
Maximum Pool Size	32
Maximum Wait Time (in milliseconds)	30000
Steady Pool Size	4

Figure 96 JMS Client Basic Configuration Properties Dialog (Consumer)



Message selector	
Concurrency	Serial mode

Table 23 JMS Client Basic Configuration Properties

Property	Applies to	Description
Transaction mode	All producers	Specifies whether messages for this session use **Transacted** or **XA** mode. For consumers, the default is always **XA**; otherwise, the default is **Transacted**.

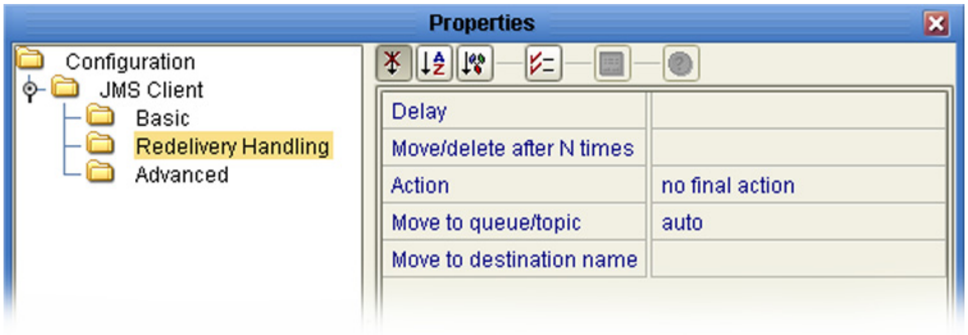
Table 23 JMS Client Basic Configuration Properties

Property	Applies to	Description
Delivery mode	All producers	Specifies whether the messages for this JMS connection are persistent or non-persistent. The default is **Persistent**.
Priority	All producers	Specifies the message priority. Allowed values are **0 - 9**; the default is **4**.
Idle timeout	All producers	Specifies the number of seconds to wait before returning a connection to the pool. The default is **30** seconds.
Maximum Pool Size	All producers	Specifies the maximum number of connections to be made to the message server. The default is **32**.
Maximum Wait Time	All producers	Specifies the maximum amount of milliseconds to wait for acquiring a connection before throwing an exception. The default is **30000** milliseconds.
Steady Pool Size	All producers	Specifies the minimum and initial number of connections maintained in the pool. The default is **4**.
Message selector	All consumers	Specifies a message selector.
Concurrency	All consumers	Specifies whether the message consumers uses connection consumer or serialized processing. The default is **Serial mode**.

Redelivery Handling

Redelivery has to do with the way messages are redelivered after previous attempts at delivery have failed. You can override the default behavior of the message redelivery process by configuring your own custom characteristics for the JMS client. Additional details are given in the *Sun SeeBeyond eGate™ Integrator JMS Reference Guide*, including an alternate method of configuring redelivery.

Figure 97 JMS Client Redelivery Handling Properties Dialog (Consumer)



Delay	
Move/delete after N times	
Action	no final action
Move to queue/topic	auto
Move to destination name	

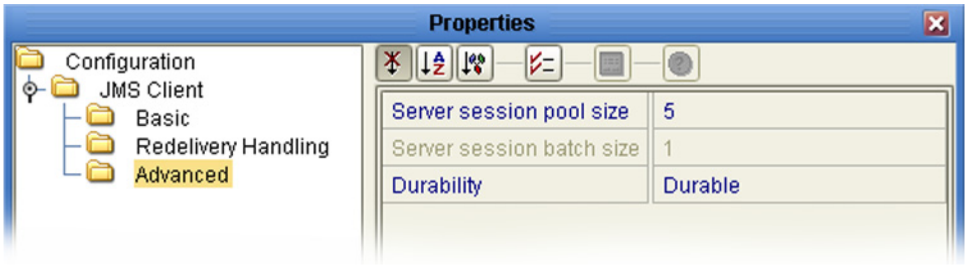
Table 24 JMS Client Redelivery Handling Configuration Properties

Property	Applies to	Description
Delay	All consumers	Specifies the delay(s) to apply following a number of retrys. The format is **<retrys:delay>**, where the number of retrys is counted from the original rollback and the delay time is in milliseconds. The maximum allowed delay is five seconds (5000 ms). Progressive delays can be specified by concatenating retry:delay pairs separated by a comma and a space.
Move/Delete after N times	All consumers	Specifies the number of retrys to allow before moving (redirecting) or deleting the message. The number of retrys is counted from the original rollback.
Action	All consumers	Specifies whether to **move** (redirect) or **delete** the message after the number of retrys specified by the previous property. The default is **no final action**, which specifies continued retrys until received.
Move to queue/topic	All consumers	Specifies whether to redirect to a **queue** or a **topic**, when move is selected as an action. The default is **auto**, which specifies redirection to a destination of the same type as the producer.
Move to destination name	All consumers	Specifies a name for the destination to which the message is to be redirected. The special character <**$**> specifies the original destination name.

Advanced Properties

Which properties appear in the dialog depends upon whether the destination is a topic or queue.

Figure 98 JMS Client Advanced Configuration Properties Dialog (Subscriber)



Server session pool size	5
Server session batch size	1
Durability	Durable

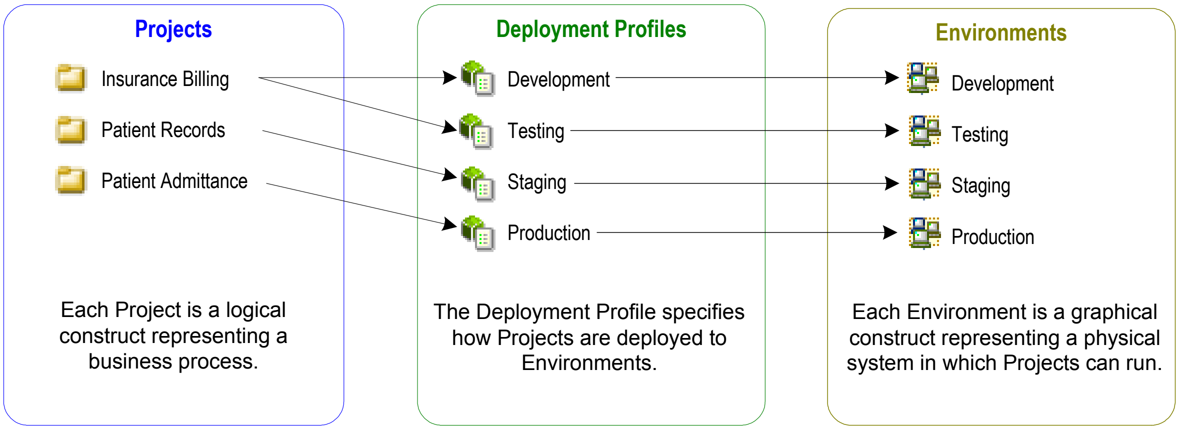
Table 25 JMS Client Advanced Configuration Properties

Property	Applies to	Description
Server session pool size	All consumers	Specifies the maximum number of threads per ServerSessionPool to be used for concurrent processing. The allowed values are **1 - 100**. The default is **5**.
Server session batch size	All consumers	Specifies the maximum number of messages that a connection consumer can load into a server session at one time. The default value is **1**, and cannot be changed.
Durability	Topic subscribers only	Specifies whether or not the subscriber to this JMS client connection is durable. The default is **Durable**.

5.11 Deployment Profiles

Deployment profiles define how specific instances of a Project are deployed to a particular Environment (see Figure 99). 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 99 eGate Integrator Implementation Model


Projects
Insurance Billing
Patient Records
Patient Admittance
Each Project is a logical construct representing a business process.
Deployment Profiles
Development
Testing
Staging
Production
The Deployment Profile specifies how Projects are deployed to Environments.
Environments
Development
Testing
Staging
Production
Each Environment is a graphical construct representing a physical system in which Projects can run.

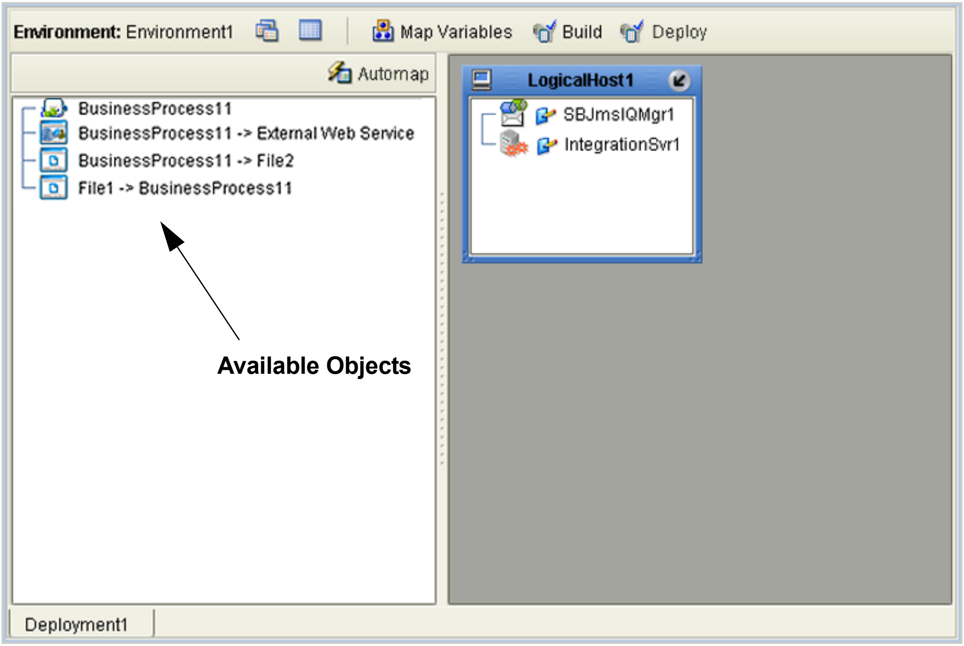
To create a new deployment profile, right-click on a Project in the Project Explorer to display its context menu. From the menu, select **New > Deployment Profile** (see **Creating a Deployment Profile** on page 140). To edit an existing deployment profile, select **Open** from its context menu (see **Using Project Component Context Menus** on page 113). 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 100) 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 100 Deployment Editor


Environment: Environment1
Map Variables
Build
Deploy
Automap
BusinessProcess11
BusinessProcess11 -> External Web Service
BusinessProcess11 -> File2
File1 -> BusinessProcess11
LogicalHost1
SBJmsIQMgr1
IntegrationSvr1
Deployment1

Available Objects

Table 26 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 **Figure 113 on page 146**).
⚡	Automatically deploys components to their matching containers, when there is a one-to-one correspondence between them. See **Automapping** on page 144.
Map Variables	Allows you to assign names and values to Project variables for the specific Deployment Profile. See **Mapping Variables** on page 145.
Build	Builds the enterprise archive (EAR) file, based on the Project and its Deployment Profile (see **Building an Application File** on page 148).
Deploy	Deploys the enterprise archive (EAR) file built in the previous step, following the Deployment Profile you have defined (see **Deploying the Project** on page 154).

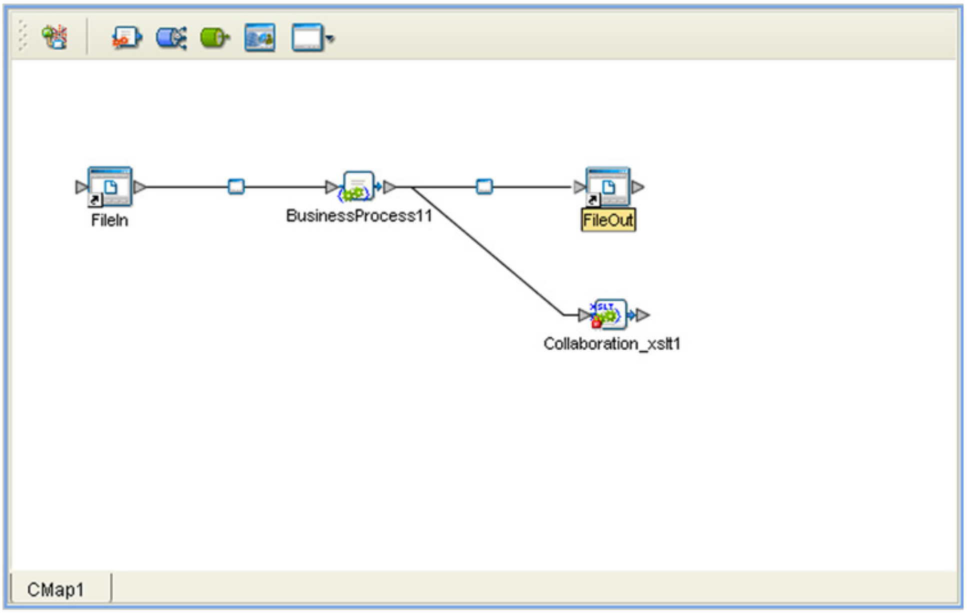
5.11.2 Creating a Deployment Profile

A Project having the Connectivity Map shown in Figure 101 will be used as a deployment example in the following procedure.

Note: *If the Project includes web services, then each of a Project's active Deployment Profiles must target a separate SOAP/HTTP External System; otherwise, the web containers can cause duplicate servlet names to appear in the domain.*

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

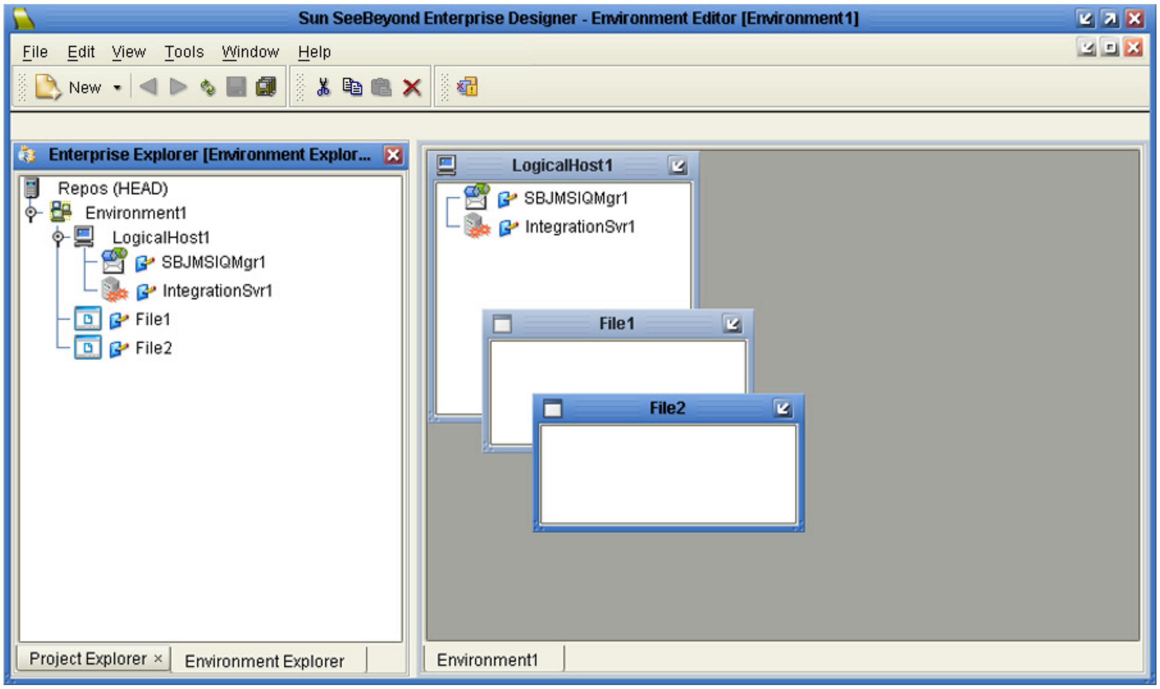
Figure 101 Example Project Connectivity Map



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 102.

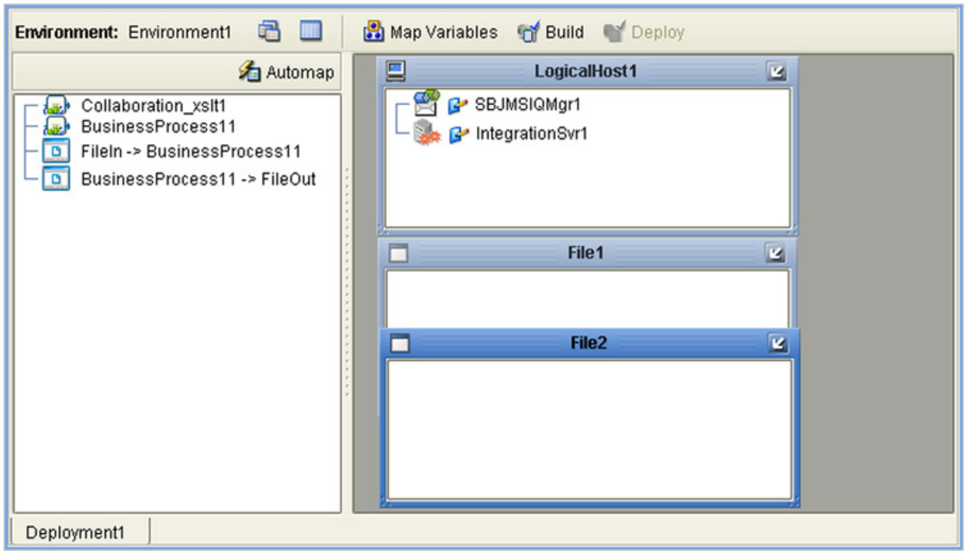
Figure 102 Web Client Example Environment



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 103).

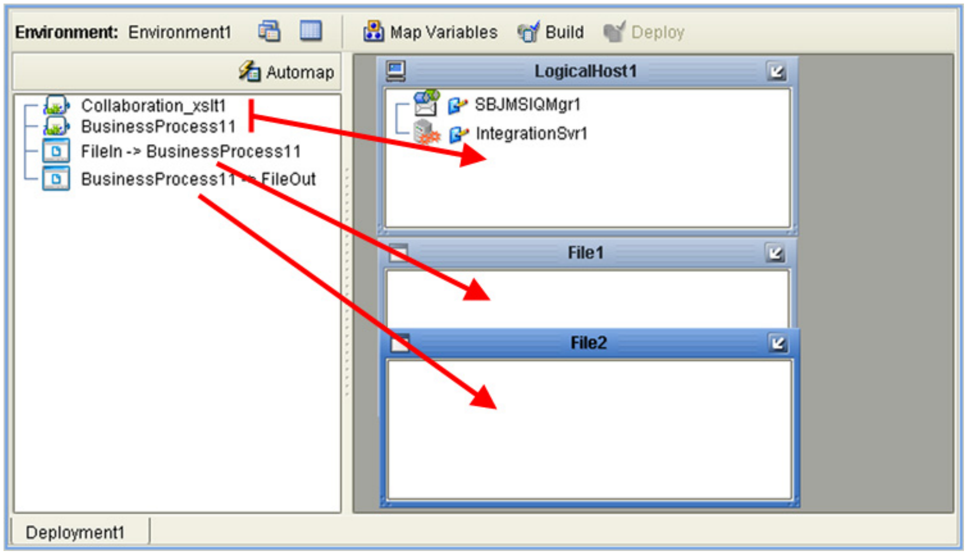
Figure 103 Example Deployment Profile (1)



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 104.

- Drag the topics and queues to the appropriate message server, being careful not to split inbound and outbound message destinations between separate servers.
- Drag the Collaborations to the appropriate integration (application) servers.
- Drag the eWay components to the appropriate external systems.

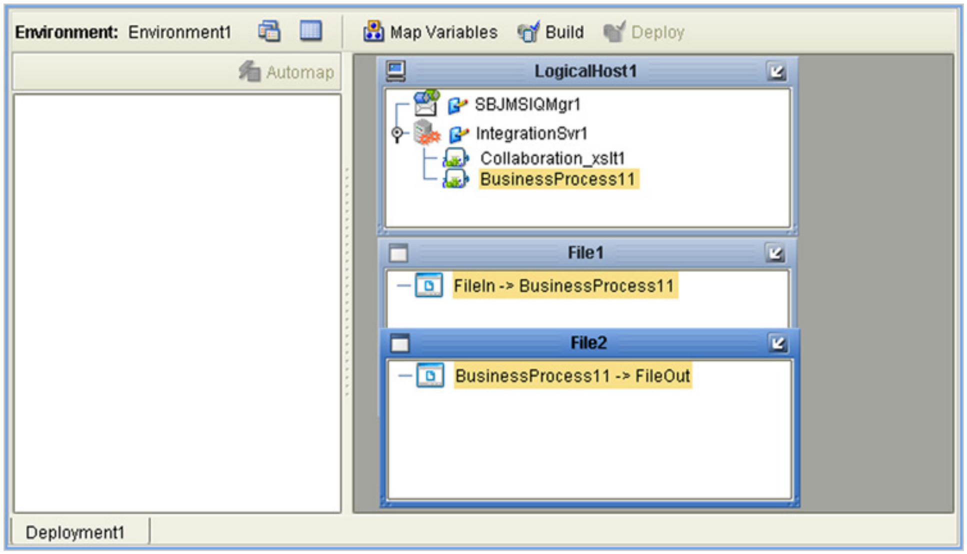
Figure 104 Example Deployment Profile (2)



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 105. You should now **Save** the profile.

Figure 105 Example Deployment Profile (3)

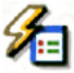


Note: *You can deploy two services that are not directly dependent upon each other (for example, two JCD services that send messages via JMS) across two Logical Hosts; however, two processes that need to have direct access to one another must be deployed to the same Logical Host.*

Automapping

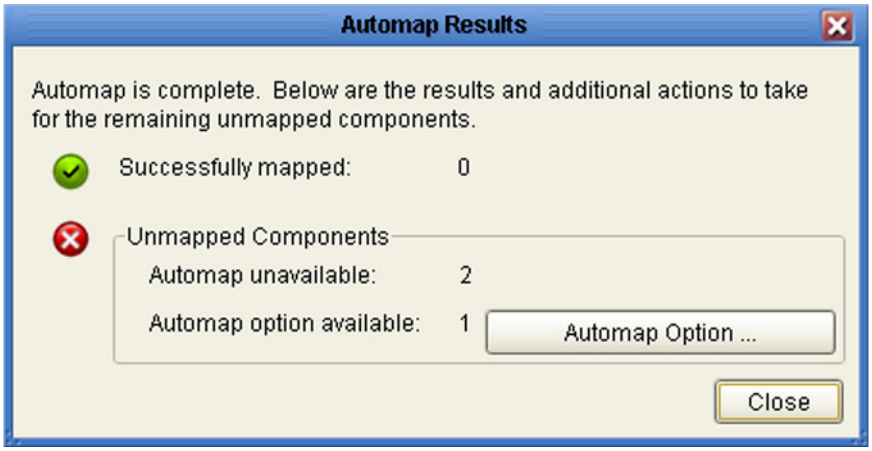
When a one-to-one correspondence exists between the available objects and the containers in the Environment to which you want to deploy them, clicking the **Automap** icon (see Figure 106) automatically deploys the components to their matching containers. (This feature only works with external systems for which it is enabled.)

Figure 106 Automap Icon



After the Automap feature executes, a dialog box is displayed showing the results (see Figure 107). The lack of an appropriate container, or ambiguity between potential containers, will cause a component to remain unmapped.

Figure 107 Automap Results Dialog Box



Automap Results

Automap is complete. Below are the results and additional actions to take for the remaining unmapped components.

Successfully mapped: 0

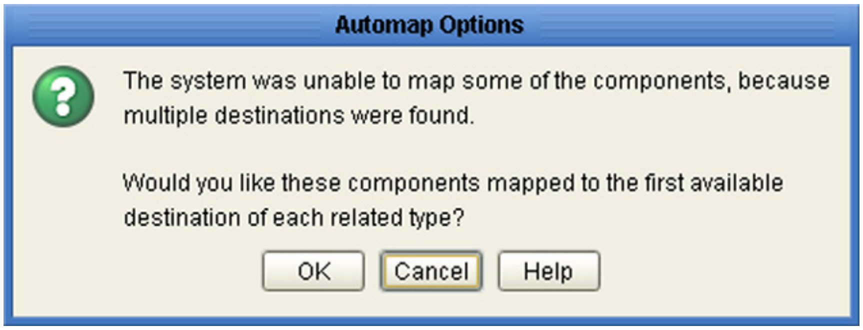
Unmapped Components

Automap unavailable: 2

Automap option available: 1

In the case of an ambiguity, for example, mapping options are presented in an Automap Options dialog box (see Figure 108) 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 108 Automap Options Dialog Box



Automap Options

The system was unable to map some of the components, because multiple destinations were found.

Would you like these components mapped to the first available destination of each related type?

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 523). Clicking the **Map Variables** button displays the Deployment Profile Mappings panel, where you can assign names (see Figure 110) and values (see Figure 111).

Figure 109 Map Variables Button

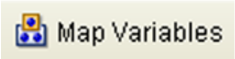
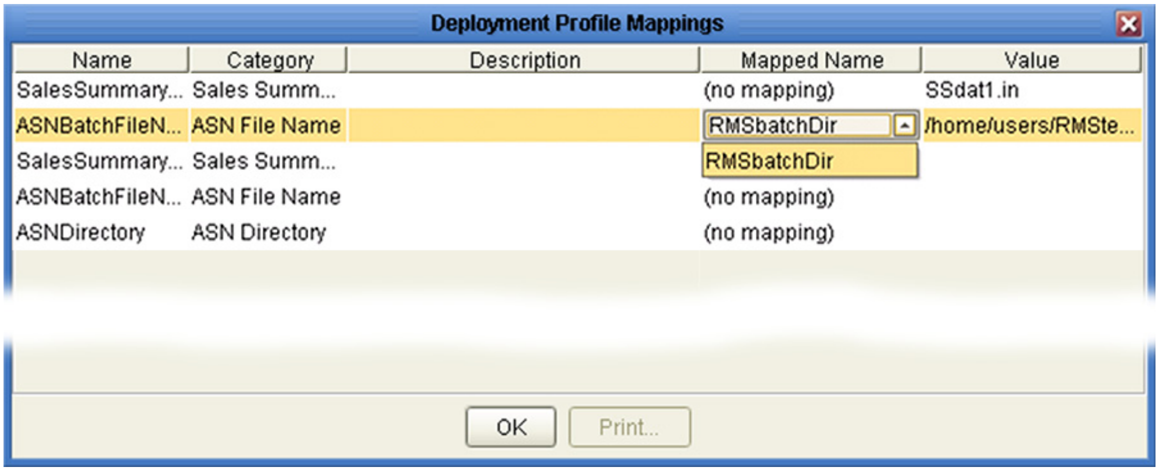
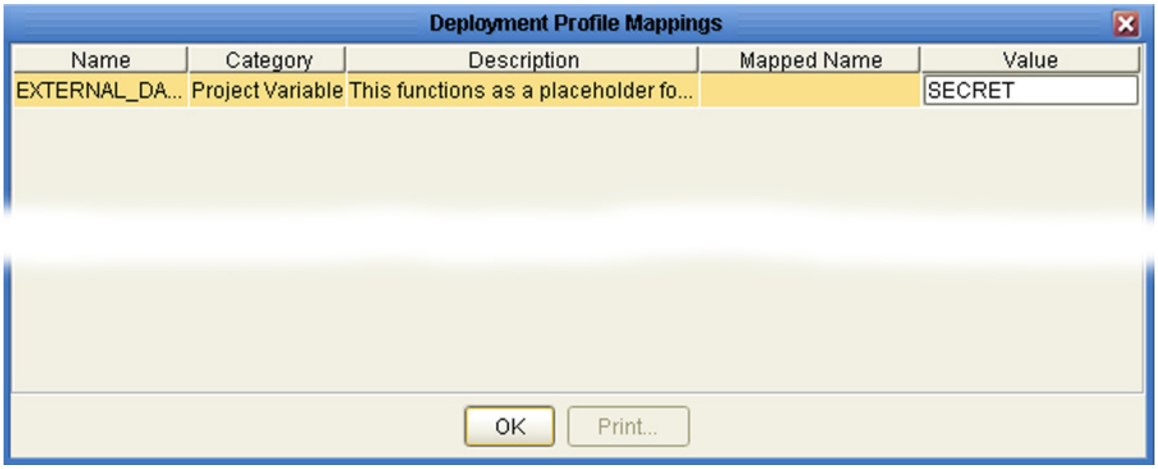


Figure 110 Deployment Profile Mappings



Name	Category	Description	Mapped Name	Value
SalesSummary...	Sales Summ...		(no mapping)	SSdat1.in
ASNBatchFileN...	ASN File Name		RMSbatchDir	/home/users/RMSte...
SalesSummary...	Sales Summ...		RMSbatchDir	
ASNBatchFileN...	ASN File Name		(no mapping)	
ASNDirectory	ASN Directory		(no mapping)	

Figure 111 Project Variable Value Entry



Name	Category	Description	Mapped Name	Value
EXTERNAL_DA...	Project Variable	This functions as a placeholder fo...		SECRET

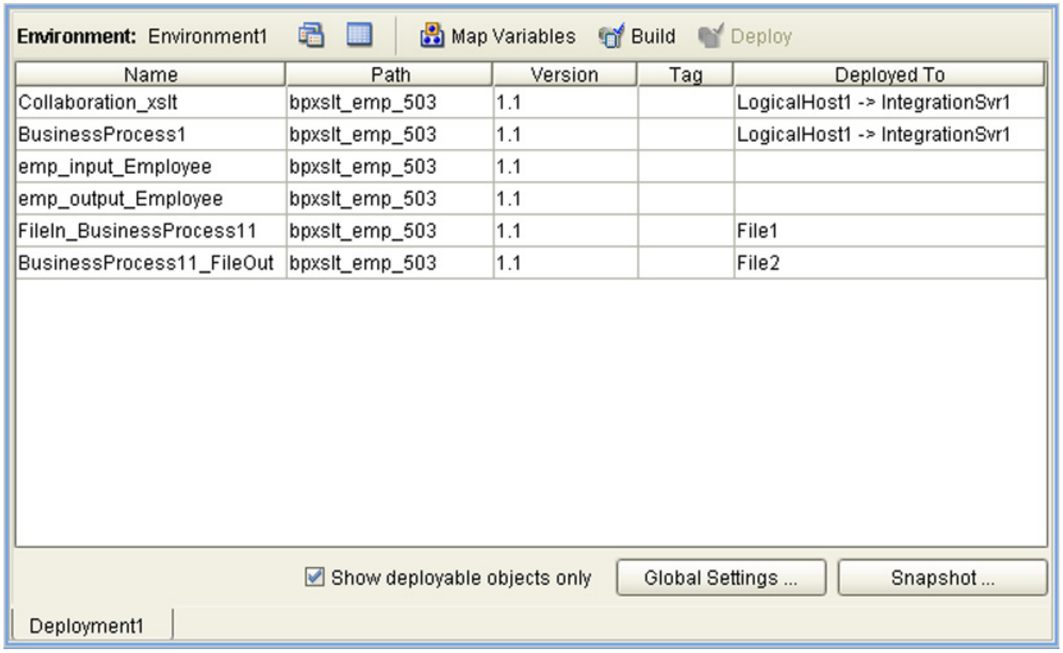
Version Control

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 112).

Figure 112 Spreadsheet View Icon

The spreadsheet view (see Figure 113) 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 113 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

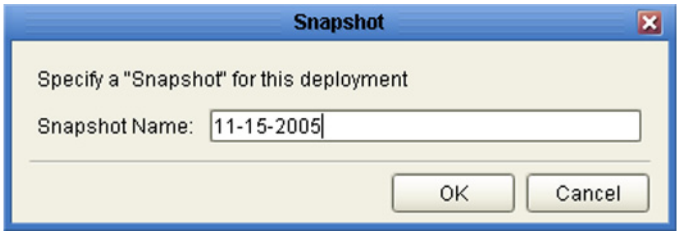
☑ Show deployable objects only

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 114). Clicking **OK** then saves the snapshot.

Figure 114 Snapshot Dialog



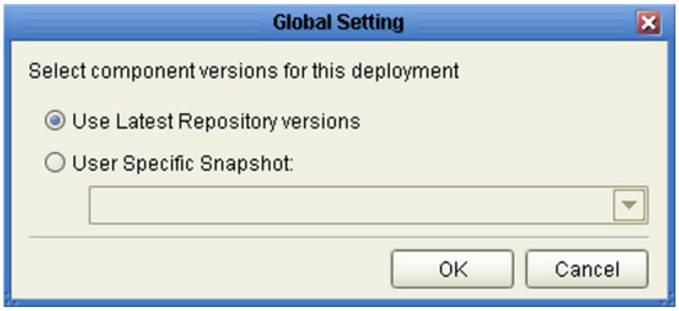
Specify a "Snapshot" for this deployment

Snapshot Name: 11-15-2005

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 115). Clicking **OK** then carries out the operation.

Figure 115 Global Settings Dialog



Select component versions for this deployment

● Use Latest Repository versions

○ User Specific Snapshot:

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 116). 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 149).

Figure 116 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 27 Commandline Codegen Properties

Property Name	Description
commandline.rep.url	Repository URL (required). The format is: *http://<host>:<port>/<repositoryname>*
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 *localrepository*.
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: *<snapshot name>, <tag1>, <tag2>, … , <tagN>*
commandline.rep.projectBranchName	Repository Branch for the Project (optional).
commandline.esr.select	Specific ESR numbers to be downloaded and applied (optional). If not specified, *all* ESRs will be downloaded. The comma-delimited format is: *<ESRnum1>, <ESRnum2>, … , <ESRnumN>*
commandline.esr.ignore	Specific ESR numbers to be ignored when downloading (optional). The comma-delimited format is: *<ESRnum1>, <ESRnum2>, … , <ESRnumN>*

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 27 (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 27):

```
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 27.
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 27 (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 27):

   ```
   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 27.
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 27:

```
<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 117). 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 117 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*.

Chapter 6

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.

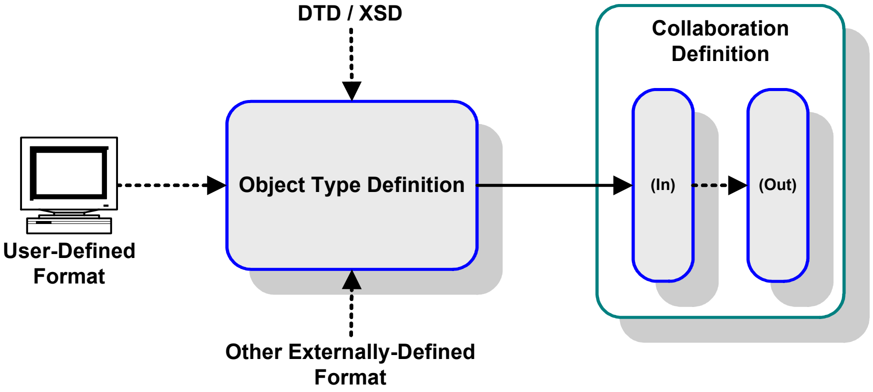
What's in This Chapter

- **Understanding OTDs** on page 155
- **Specifying Data Conversion and Encoding** on page 160
- **Using the OTD Editor** on page 168

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.

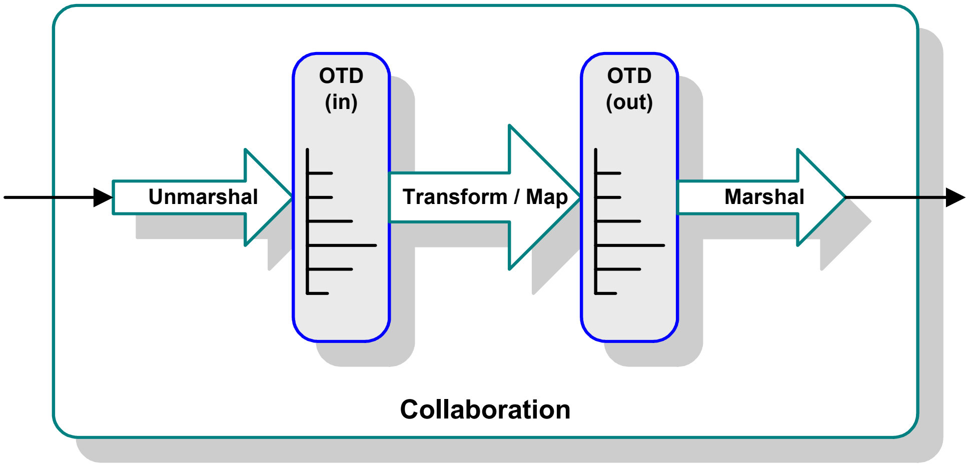
Figure 118 OTD Generation



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 119). 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 though 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 119 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.

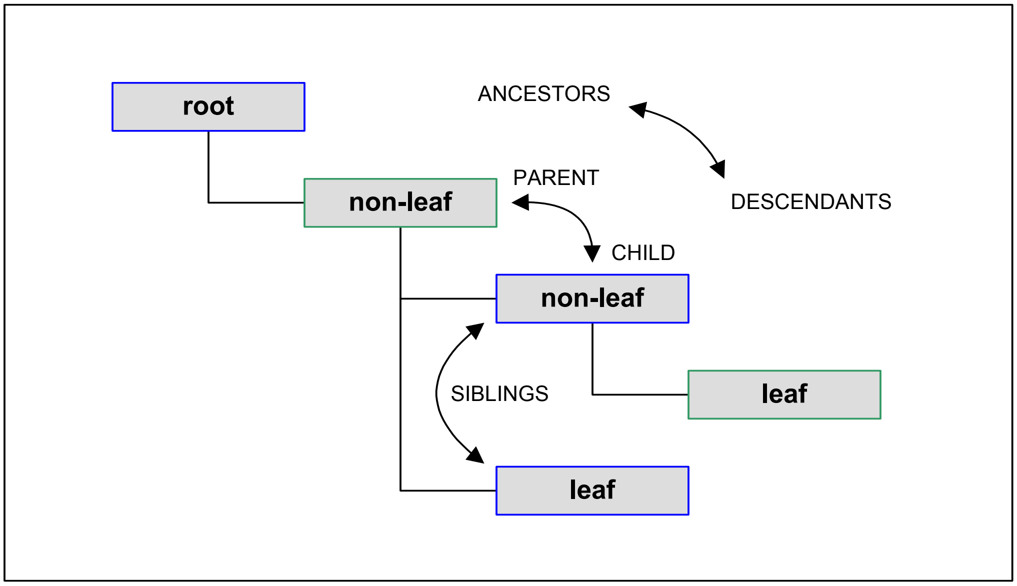
Data Structure

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 120 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 230).

- 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 235).

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 81 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 183.

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 200.

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 the JMS OTD** on page 206.

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 28.

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 28 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
EBCDIC	EUC-JP	EUC-KR		
UTF-16	SJIS	MS949		
	MS932			

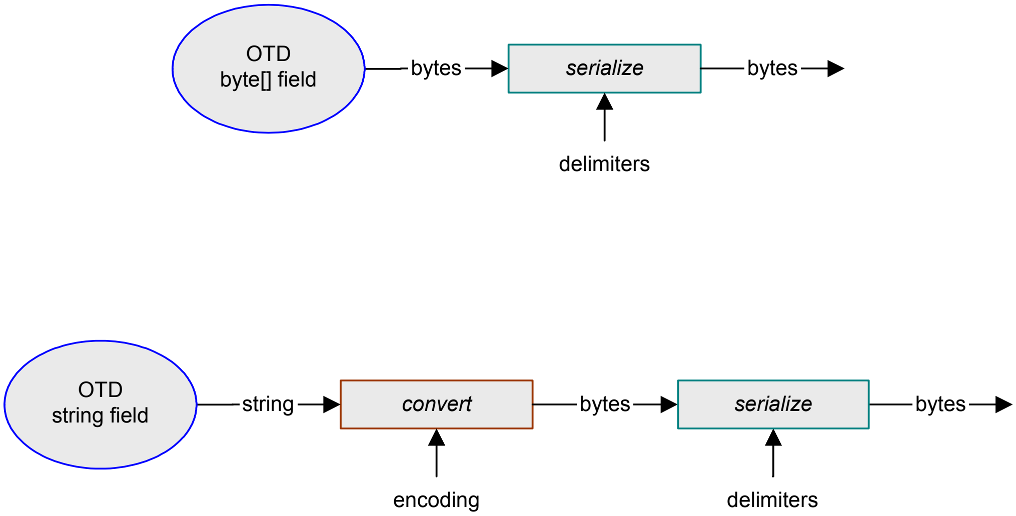
6.2.2 Marshaling and Unmarshaling Methods

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 121).

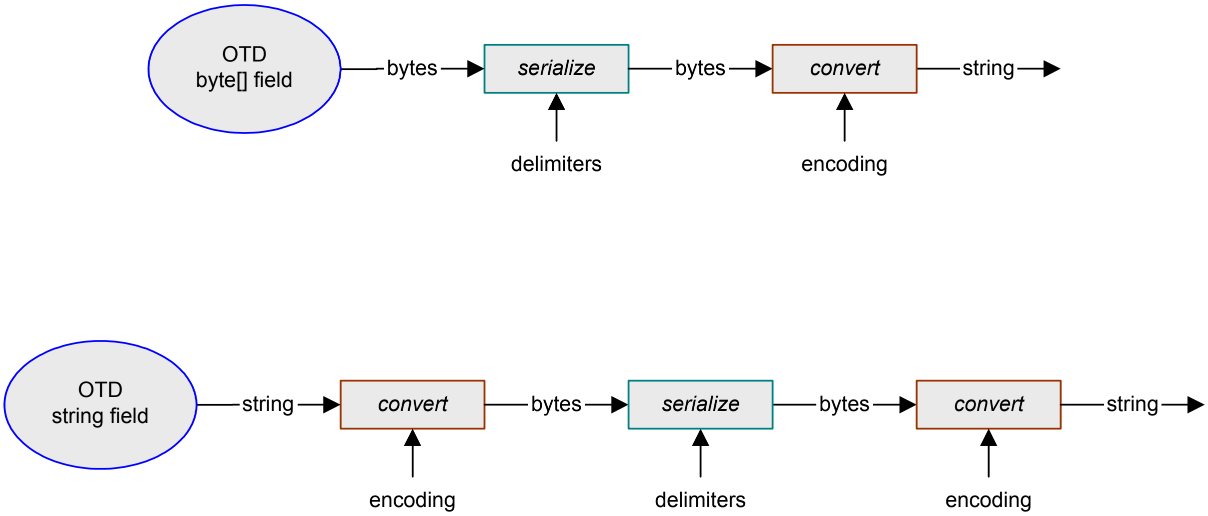
Figure 121 marshal()



marshalToString()

The **marshalToString()** method requires *encoding* to produce an output string after marshaling from a byte[] field (see Figure 122). 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.

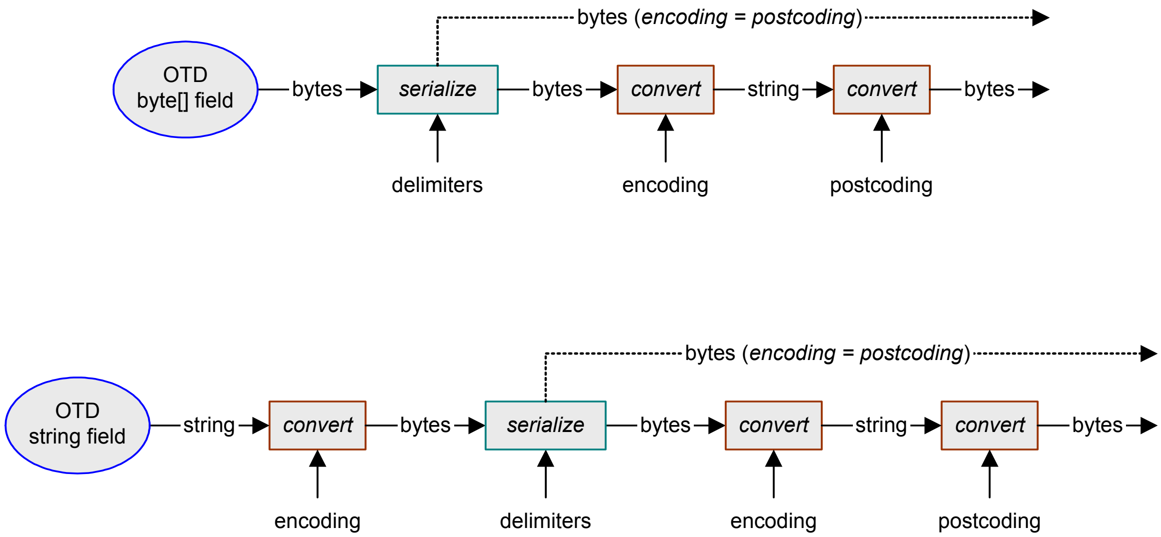
Figure 122 marshalToString()



marshalToBytes()

The **marshalToBytes()** method requires *encoding* to produce bytes after marshaling from a string field (see Figure 123). 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.

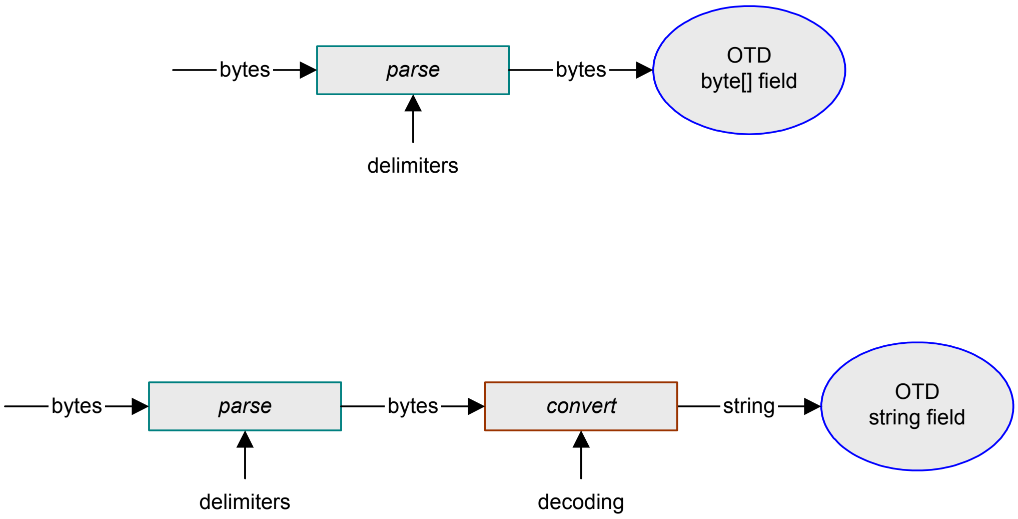
Figure 123 marshalToBytes()



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 124).

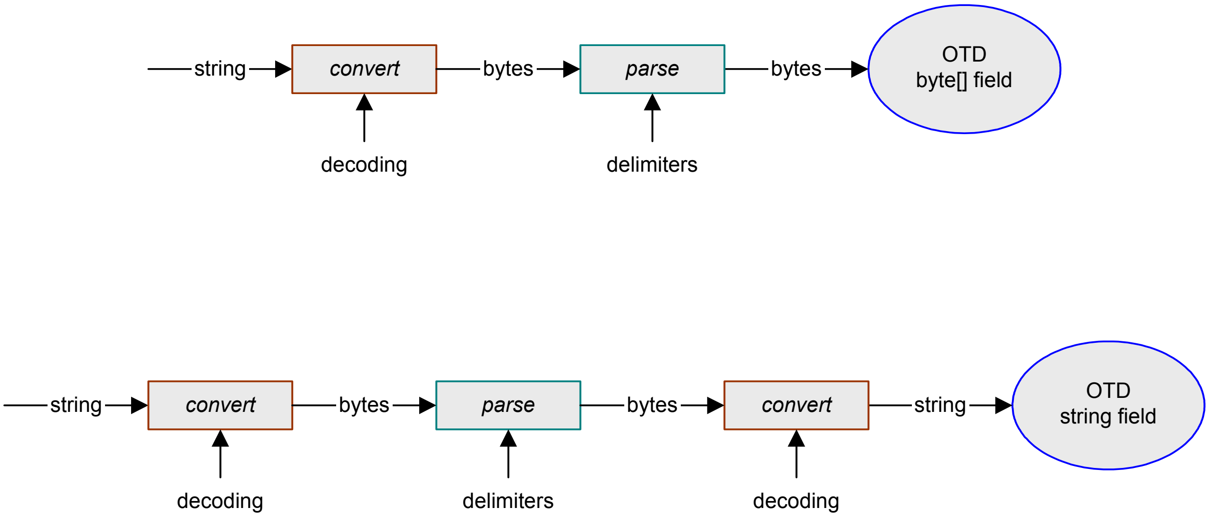
Figure 124 unmarshal()



unmarshalFromString()

The **unmarshalFromString()** method requires *decoding* of the input string, since the parser requires the data in bytes (see Figure 125). This method requires a second *decoding* when unmarshaling to a string field.

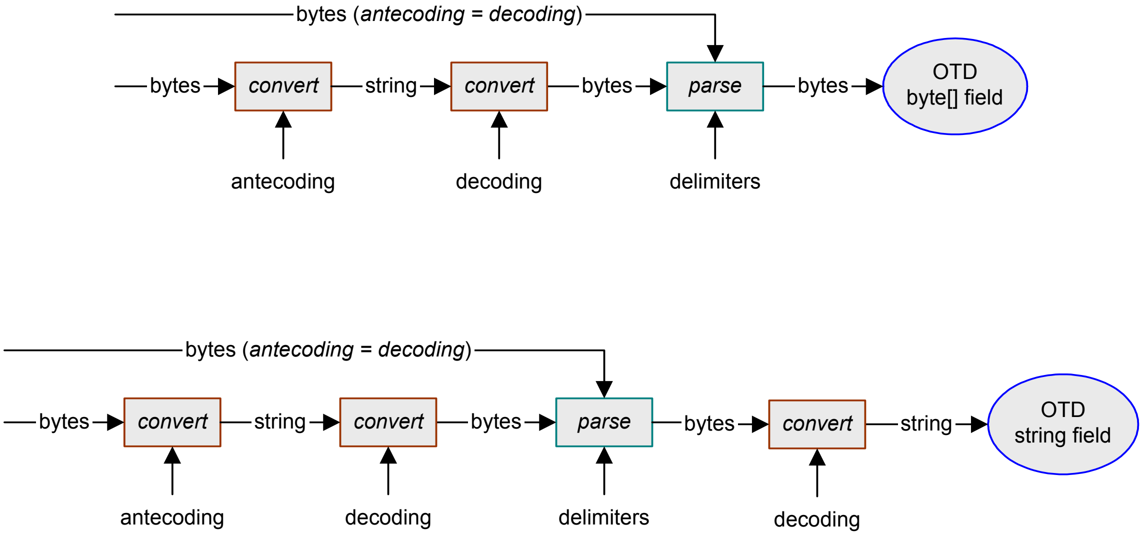
Figure 125 unmarshalFromString()



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 126). 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.

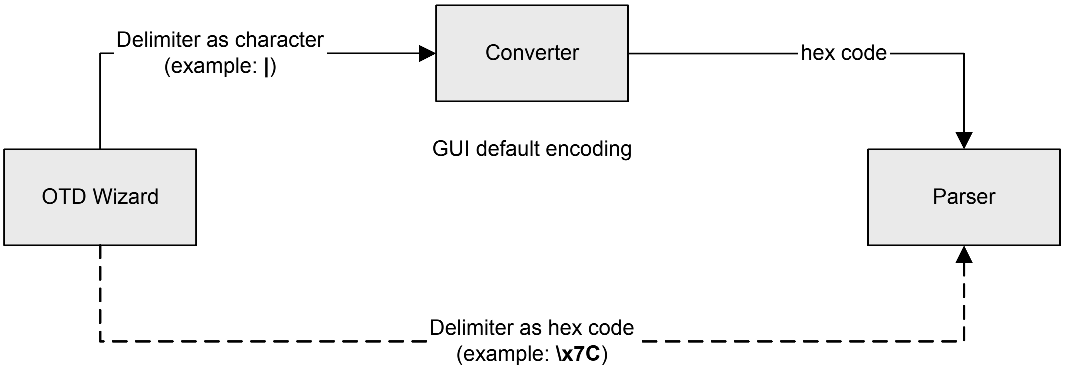
Figure 126 unmarshalFromBytes()



6.2.3 Setting Delimiters

Figure 127 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 127 Setting Delimiters



6.3 Using the OTD Editor

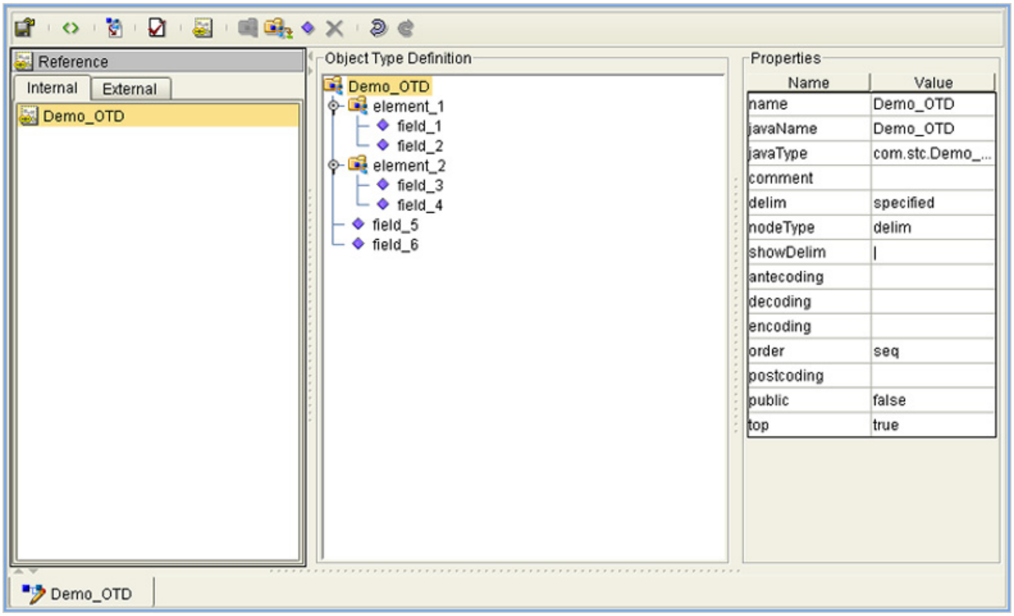
The OTD Editor displays the structure of the selected Object Type Definition (OTD) and allows you to verify its operation with a built-in tester. You can also use the editor to create and modify User-Defined OTDs.

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 128. 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 214.

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 128 OTD Editor



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

Table 29 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 30 OTD Editor Toolbar Icons

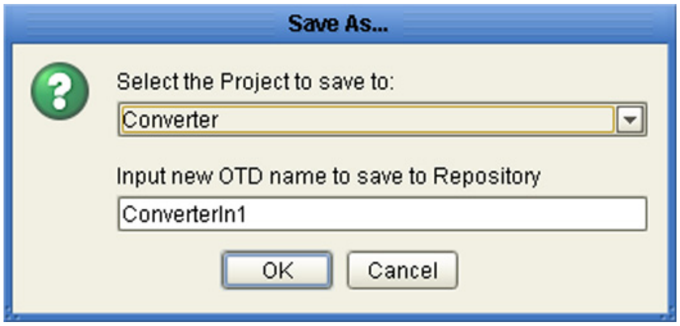
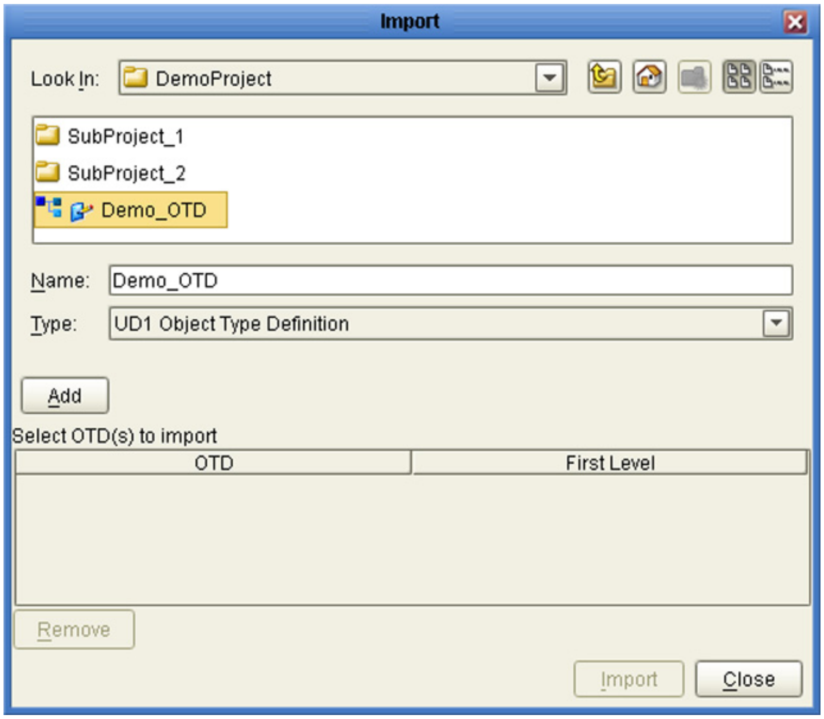
Icon	Command	Function
	Save as New Name to Repository	Presents a dialog with which you can save the current OTD under a new name to any Project or Subproject in the Repository.  Select the Project to save to: Converter Input new OTD name to save to Repository: ConverterIn1
<>	Generate Code	
	Import OTD to External Template	Presents a dialog with which you can import an OTD as an external template (it will appear in the *Reference: External* tab).  Look In: DemoProject SubProject_1 SubProject_2 Demo_OTD Name: Demo_OTD Type: UD1 Object Type Definition Select OTD(s) to import OTD / First Level
✓	Tester	Displays/hides the OTD Tester.

Table 30 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

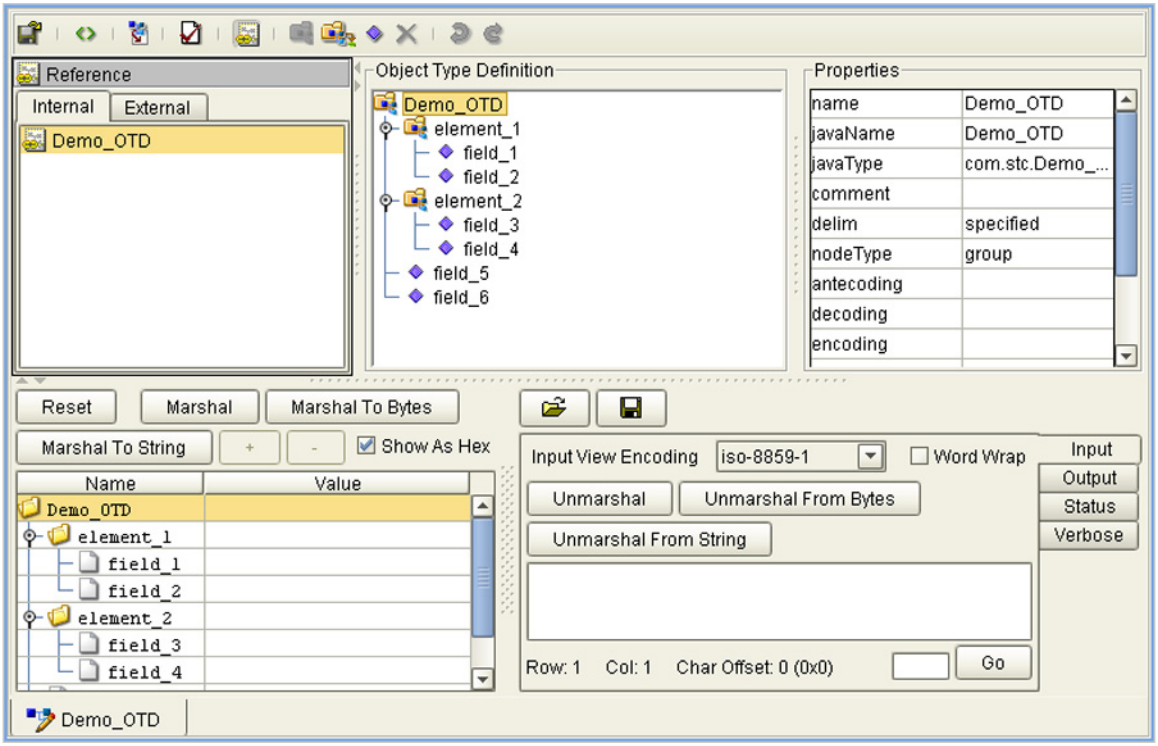
The OTD Tester facility allows you to simulate the operation of a Collaboration containing a specific OTD, thereby checking the correctness of the OTD during the design phase. You can enter input data values, perform the unmarshal and marshal operations, and also manipulate the OTD tree structure as 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 129) 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 130. Clicking the icon again hides the Tester. Use of the OTD Tester is described in **Using the OTD Tester** on page 178.

Figure 129 Tester Icon

Figure 130 OTD Tester Interface



name	Demo_OTD
javaName	Demo_OTD
javaType	com.stc.Demo_...
comment	
delim	specified
nodeType	group
antecoding	
decoding	
encoding	

☑ Show As Hex

Input View Encoding iso-8859-1

☐ Word Wrap

Row: 1 Col: 1 Char Offset: 0 (0x0)

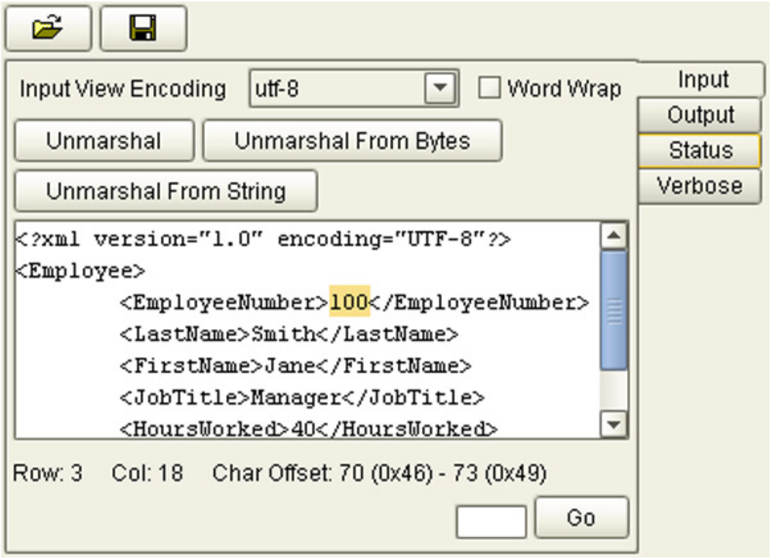
The data display panel on the right has four data display modes, selectable by tabs:

- 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 131).

Figure 131 Data Field Display



Input View Encoding utf-8

```
<?xml version="1.0" encoding="UTF-8"?>
<Employee>
        <EmployeeNumber>100</EmployeeNumber>
        <LastName>Smith</LastName>
        <FirstName>Jane</FirstName>
        <JobTitle>Manager</JobTitle>
        <HoursWorked>40</HoursWorked>
```

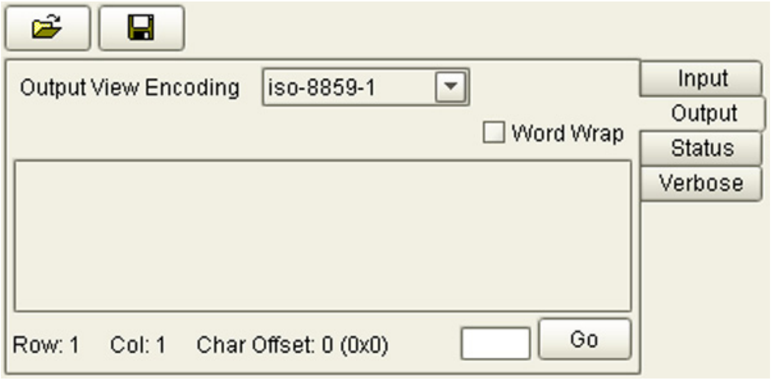
Row: 3 Col: 18 Char Offset: 70 (0x46) - 73 (0x49)

The *Input* panel also has buttons for running the tester in the three different unmarshaling modes (see **OTD Tester Controls** on page 175). Usage is described in **To test data unmarshaling functionality** on page 178.

Output Panel

The *Output* panel displays the results of a marshaling operation. Usage is described in **To test data marshaling functionality** on page 179.

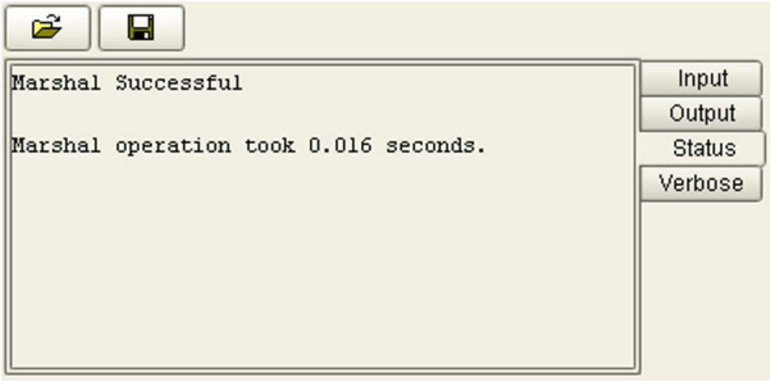
Figure 132 Output Panel



Status Panel

The *Status* panel shows the status of an attempted operation. Figure 133 shows the results of a successful operation; if the operation had failed, error information would be shown.

Figure 133 Status Panel

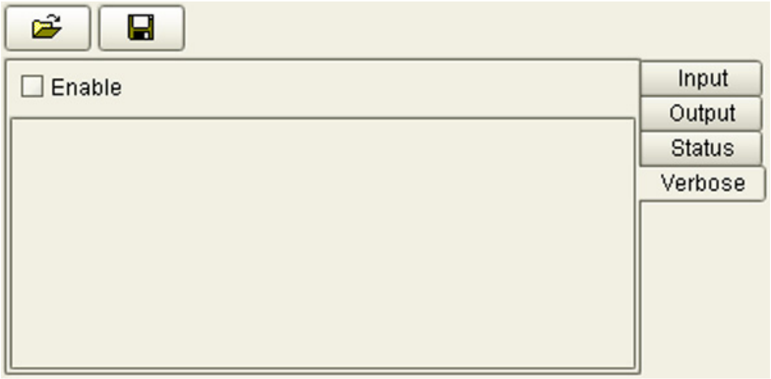


```
Marshal Successful

Marshal operation took 0.016 seconds.
```

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 134). The format and content of the data display are OTD-specific.

Figure 134 Verbose Panel



OTD Tester Controls

A large number of buttons, icons, and text appear in the OTD Tester interface. Table 31 contains a list of these artifacts, with brief descriptions of their respective functions.

Table 31 OTD Tester Controls

Button/Icon	Command	Function
Marshal		Runs the tester with the data values entered in the table; the data is marshaled from the OTD tree using the *marshal* method (see Table 32) and displayed in the *Output* panel.
Marshal To Bytes		Runs the tester with the data values entered in the table; the data is marshaled from the OTD tree using the *marshalToBytes* method (see Table 32) and displayed in the *Output* panel.
Marshal To String		Runs the tester with the data values entered in the table; the data is marshaled from the OTD tree using the *marshalToString* method (see Table 32) and displayed in the *Output* 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.
Reset		Resets the tester, repopulating the OTD with the current data from the *Input* 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 *Open File* 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 *Input* panel of the tester.
💾	Save File	Presents the *Save File* dialog, with which you can save the data from the selected OTD Tester panel, along with the encoding to use.

Table 31 OTD Tester Controls

Button/Icon	Command	Function
Unmarshal		Runs the tester, unmarshaling the data from the *Input* panel to the OTD tree using the *unmarshal* method (see Table 32).
Unmarshal From Bytes		Runs the tester, unmarshaling the data from the *Input* panel to the OTD tree using the *unmarshalFromBytes* method (see Table 32).
Unmarshal From String		Runs the tester, unmarshaling the data from the *Input* panel to the OTD tree using the *unmarshalFromString* method (see Table 32).
(None)	Word Wrap	Selecting this check box causes the content of the *Input* or *Output* 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 **Data Encoding** on page 177.
□ Go		Typing a character offset into the text box and clicking **Go** advances the cursor to that point in the currently selected line. (In the *Output* panel, the cursor position is indicated by a faint, yellow, vertical line that can be difficult to see.)

Marshal/Unmarshal Processes

Table 32 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

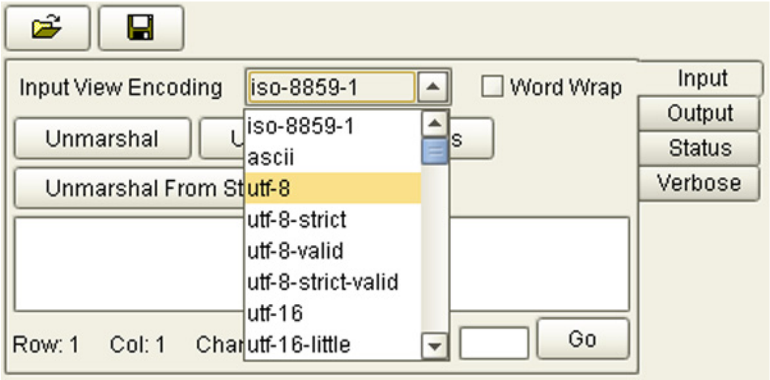
Note: *See* **Marshaling and Unmarshaling Methods** *on page 161 for expanded descriptions of the marshaling and unmarshaling processes and the associated default conditions.*

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 135.

Figure 135 Input View Encoding Option Menu



Input View Encoding: iso-8859-1

Marshaling

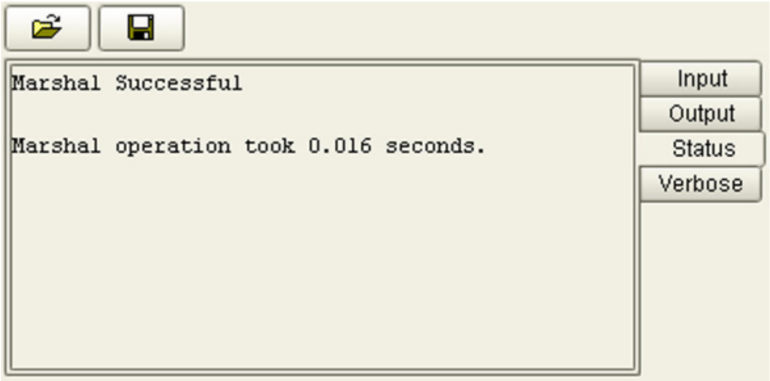
If you call **marshalToBytes**, the byte array returned by the method will be converted to a string using the character encoding selected in the *Output View Encoding* drop-down menu, which is identical to the *Input View Encoding* menu shown in Figure 135. That string will then be displayed in the *Output* 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 30 on page 169**).
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 31 on page 175**) 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 136.

Figure 136 Data Display - Status Panel



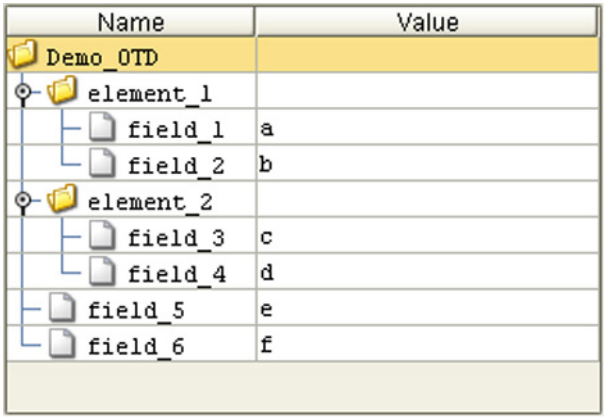
```
Marshal Successful

Marshal operation took 0.016 seconds.
```

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 137). Use the Add/Delete Instance (**+**/**-**) buttons to add or remove instances where appropriate.

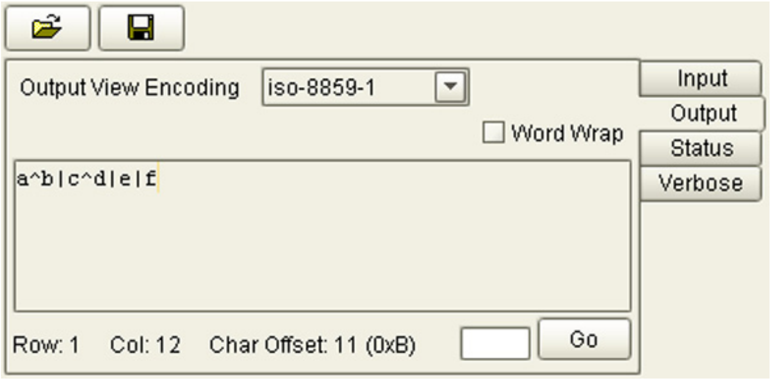
Figure 137 OTD Tester Node Table



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 177.
5. Click the desired marshal button to marshal (serialize) the data, as described in **Table 31 on page 175**.
6. The output is displayed in the *Output* panel (see Figure 138).

Figure 138 Serialized Data in Output Panel



Output View Encoding iso-8859-1

□ Word Wrap

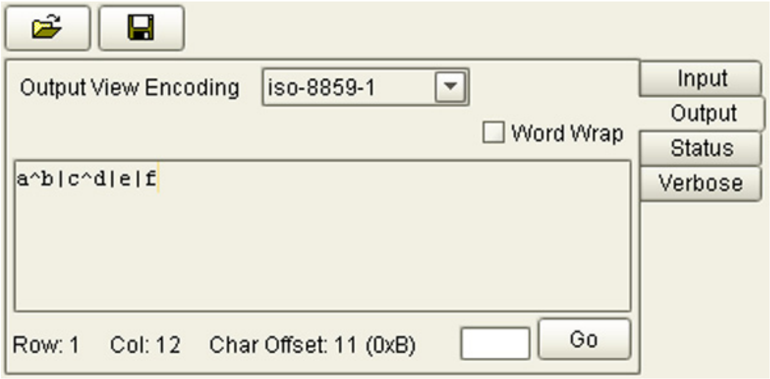
```
a^b|c^d|e|f
```

Row: 1 Col: 12 Char Offset: 11 (0xB)

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 138).

Figure 139 Saving Test Data



Output View Encoding iso-8859-1

□ Word Wrap

```
a^b|c^d|e|f
```

Row: 1 Col: 12 Char Offset: 11 (0xB)

2 Click the **Save File** button to display the *Save File* dialog.

3 Specify the desired location for the file.

4 Click **Save**.

Chapter 7

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 181
- **Using the New OTD Wizard** on page 182
- **Creating Externally-Defined OTDs** on page 183
- **Creating User-Defined OTDs** on page 200
- **Using Predefined OTDs** on page 205
- **Using the OTD Context Menu** on page 212

7.1 Overview

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

- Externally-defined
- User-defined
- Predefined

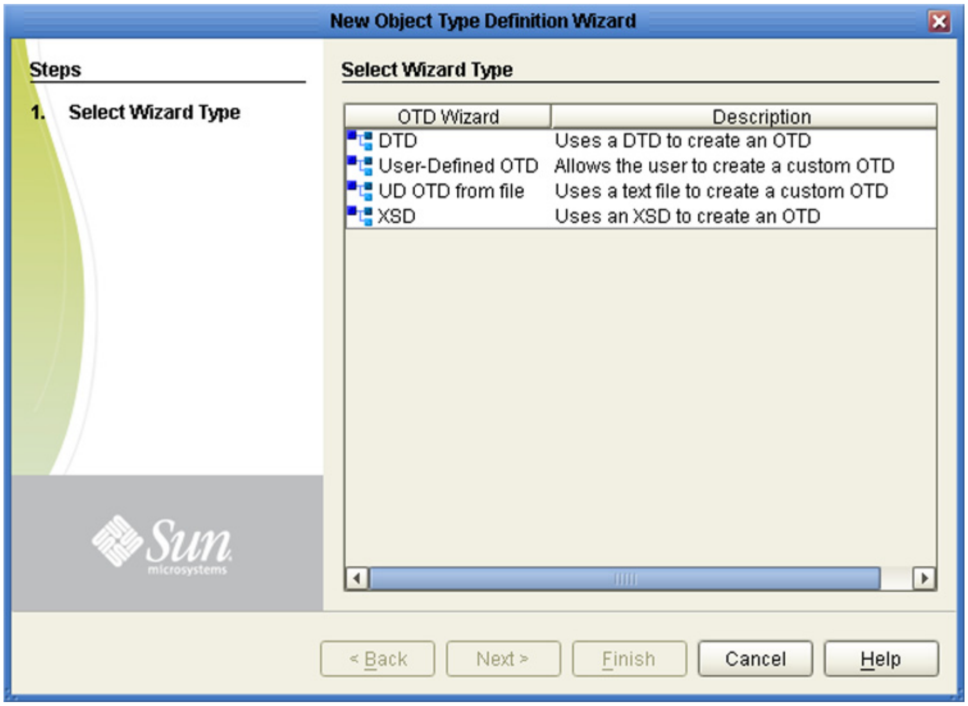
For externally-defined and user-defined OTDs, wizards are provided in Enterprise Designer to guide you through the OTD generation process. These wizards call back-end builders that actually implement the building of the code, based on the provided information. Note that some OTD Libraries fall into the 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 140. This initial dialog allows you to select the specific type of OTD Wizard needed for your application.

Figure 140 OTD Wizard Selection Dialog



OTD Wizard	Description
DTD	Uses a DTD to create an OTD
User-Defined OTD	Allows the user to create a custom OTD
UD OTD from file	Uses a text file to create a custom OTD
XSD	Uses an XSD to create an OTD

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

For externally-defined OTDs

- **Using the DTD Wizard** on page 183
- **Using the XSD Wizard** on page 190

For user-defined OTDs

- **Using the User-Defined OTD Wizard** on page 200
- **Using the UD OTD from File Wizard** on page 202

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 140.

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.

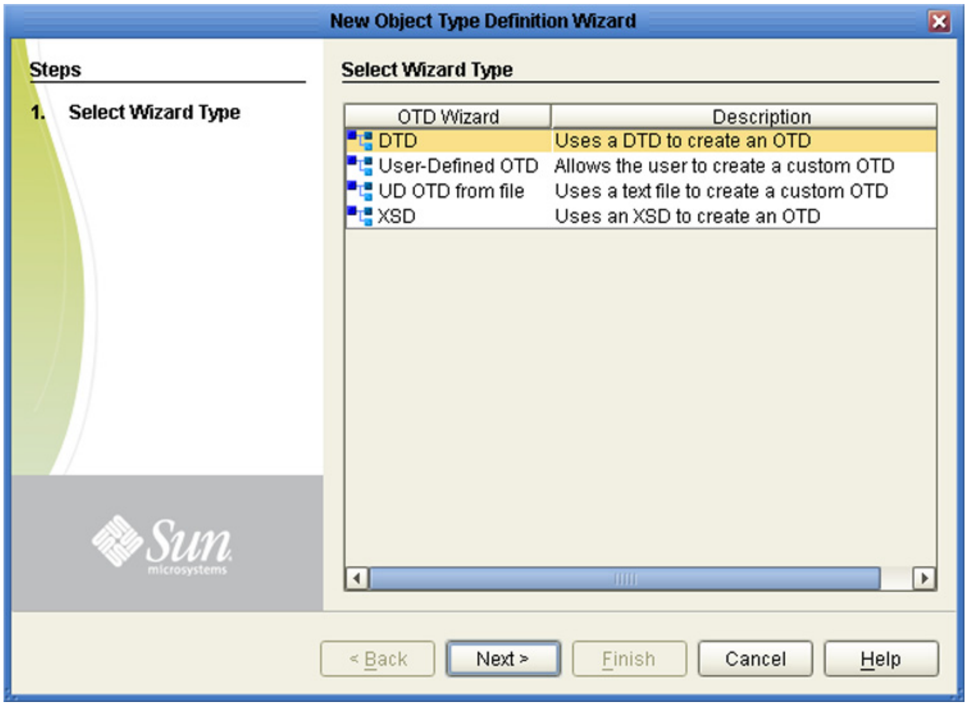
7.3.1 Using the DTD Wizard

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 141) to create an OTD from a Data Type Definition (DTD) file.

Figure 141 OTD Wizard Selection: DTD Wizard



New Object Type Definition Wizard

Steps

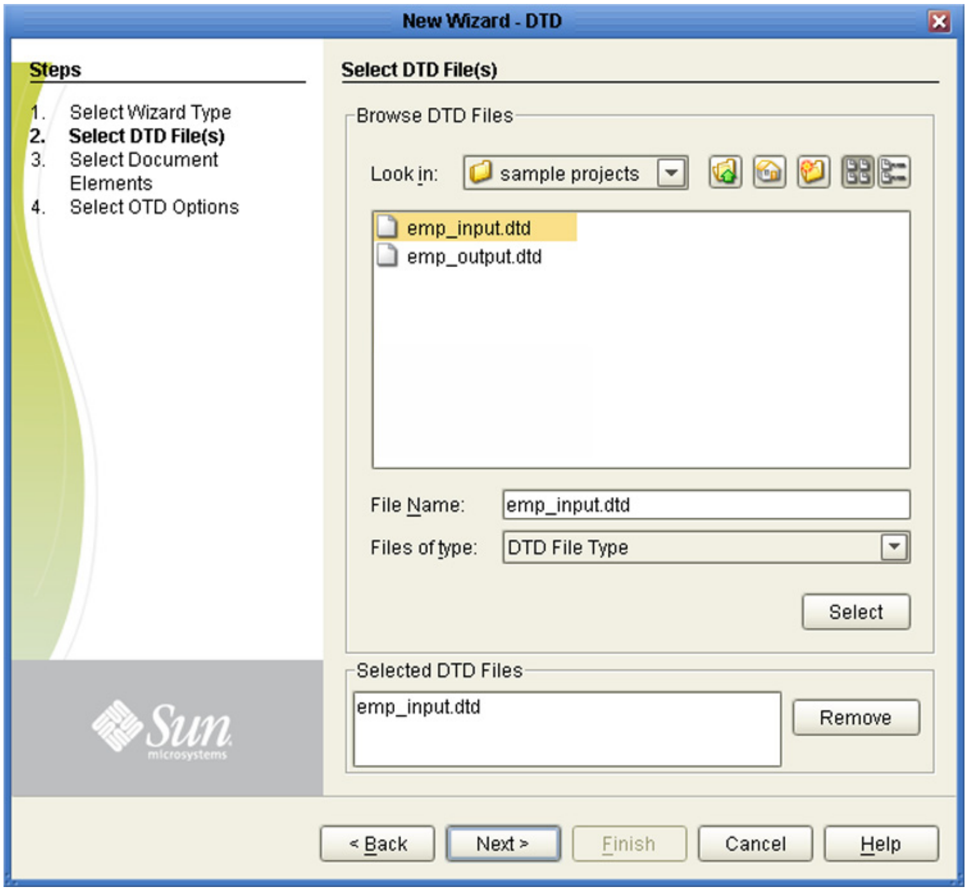
1. Select Wizard Type

Select Wizard Type

OTD Wizard	Description
DTD	Uses a DTD to create an OTD
User-Defined OTD	Allows the user to create a custom OTD
UD OTD from file	Uses a text file to create a custom OTD
XSD	Uses an XSD to create an OTD

2 Click **Next** to display the *Select DTD File(s)* dialog, shown in Figure 142.

Figure 142 Select DTD File(s) Dialog



New Wizard - DTD

Steps

1. Select Wizard Type
2. **Select DTD File(s)**
3. Select Document Elements
4. Select OTD Options

Select DTD File(s)

Browse DTD Files

Look in: sample projects

emp_input.dtd
emp_output.dtd

File Name: emp_input.dtd

Files of type: DTD File Type

Selected DTD Files

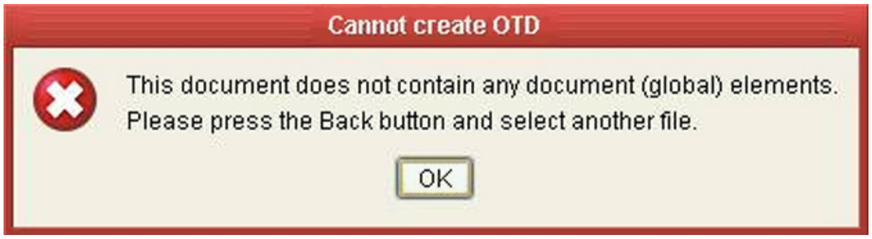
emp_input.dtd

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 143 will be displayed.

Figure 143 Cannot Create OTD Warning Box



Cannot create OTD

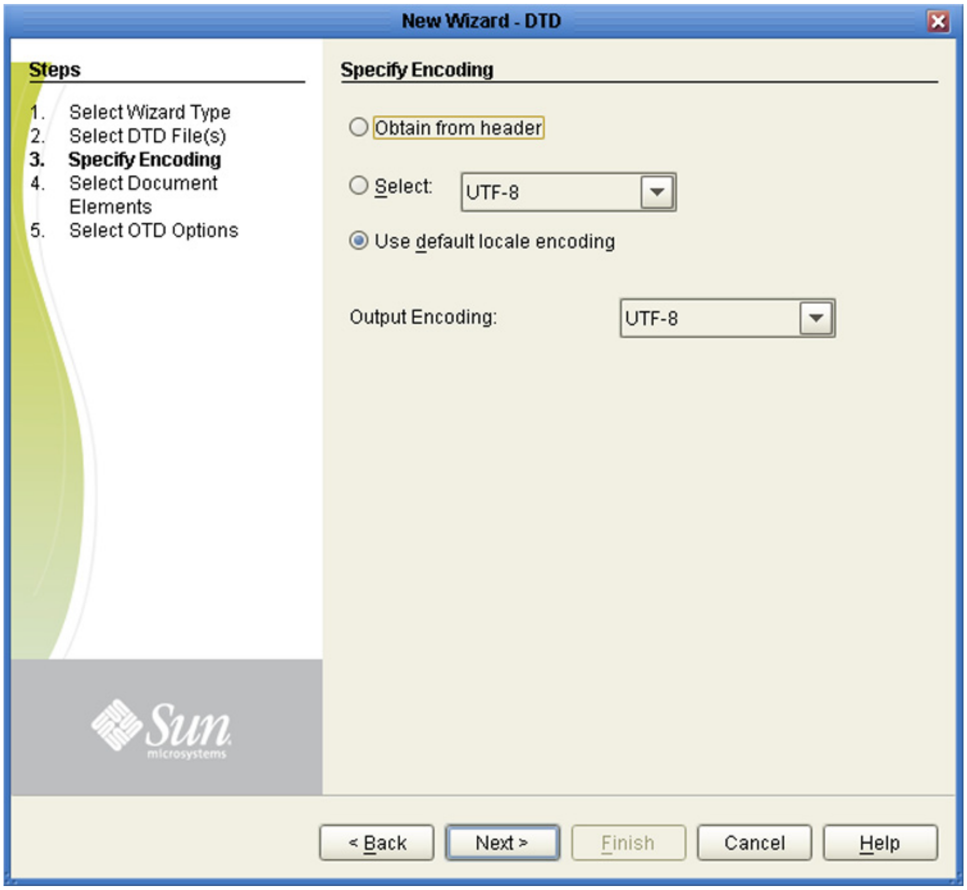
This document does not contain any document (global) elements.
Please press the Back button and select another file.

4 If you are using the extended language options (see **Setting Up Options** on page 55) click **Next** to display the *Specify Encoding* dialog, shown in Figure 144. 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 144 Select Encoding Dialog



New Wizard - DTD

Steps

1. Select Wizard Type
2. Select DTD File(s)
3. **Specify Encoding**
4. Select Document Elements
5. Select OTD Options

Specify Encoding

○ Obtain from header

○ Select: UTF-8

● Use default locale encoding

Output Encoding: UTF-8

5 Click the appropriate option button to specify the data encoding for the input XML file:

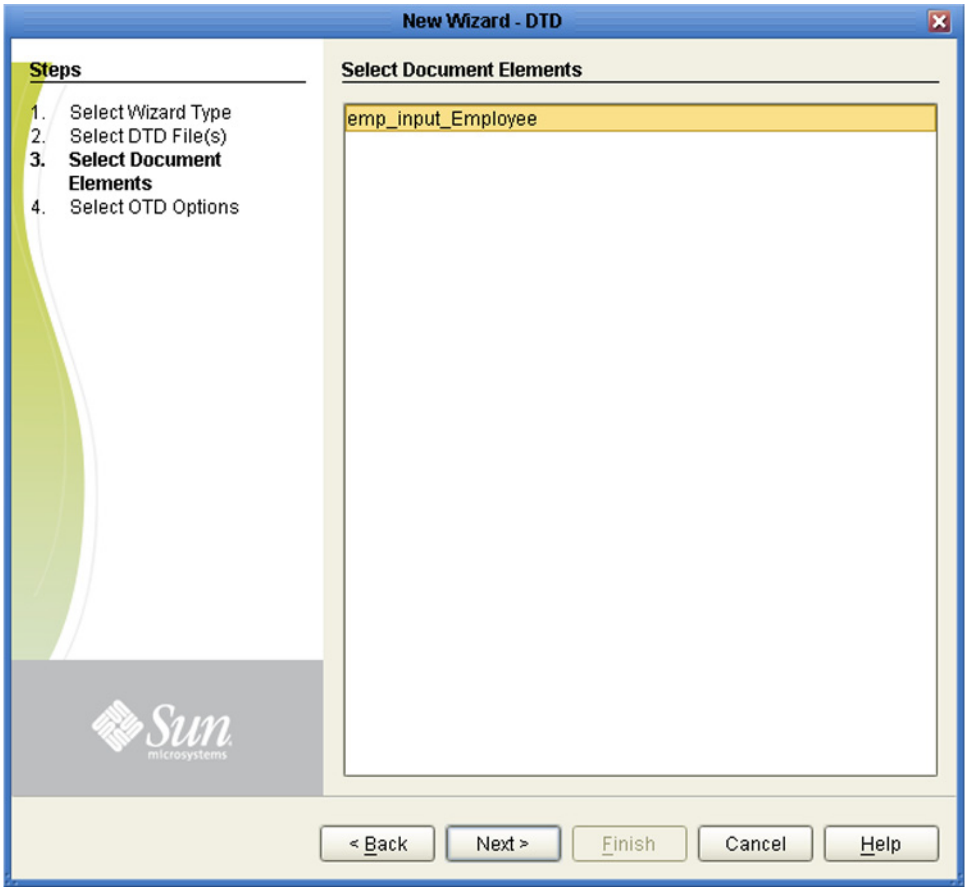
- **Obtain from header** does not apply to DTD OTDs.
- Click **Select** to explicitly specify the encoding from a drop-down list, the contents of which depends upon the locale (see **Specifying Data Conversion and Encoding** on page 160).
- Select **Use default locale encoding** to use the default encoding 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 160).

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 145.

Figure 145 Select Document Elements Dialog



New Wizard - DTD

Steps

1. Select Wizard Type
2. Select DTD File(s)
3. **Select Document Elements**
4. Select OTD Options

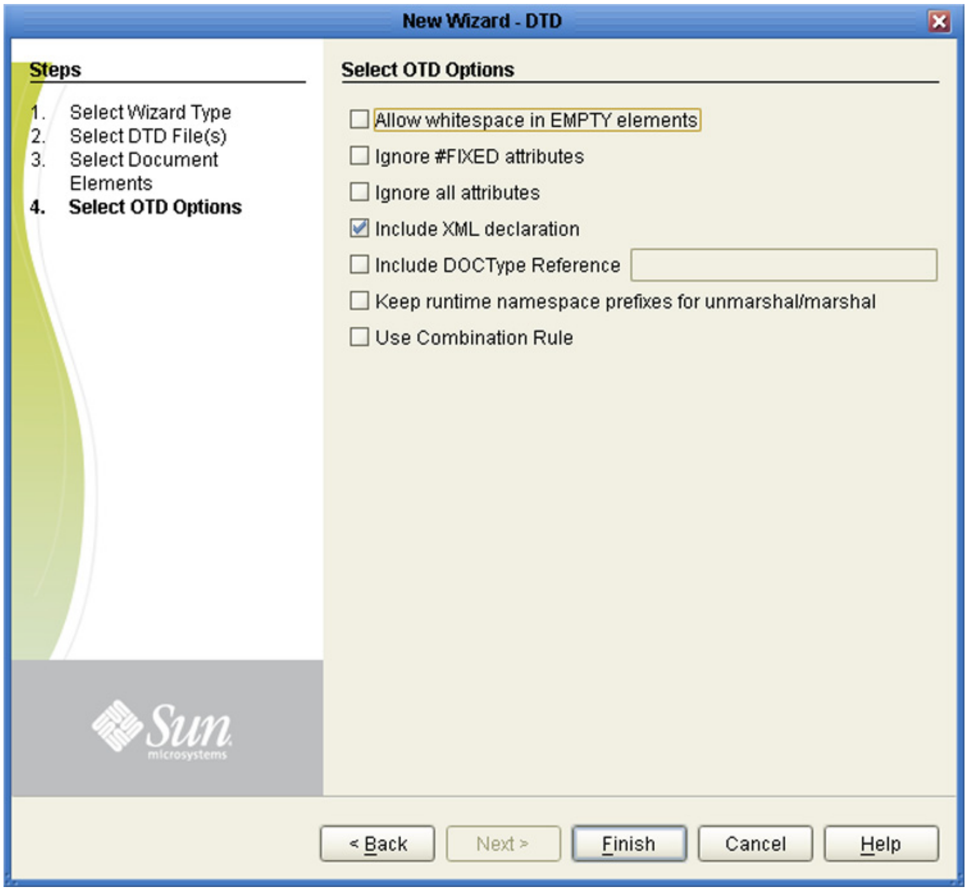
Select Document Elements

emp_input_Employee

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 146.

Figure 146 Select OTD Options Dialog



Select OTD Options

□ Allow whitespace in EMPTY elements
□ Ignore #FIXED attributes
□ Ignore all attributes
☑ Include XML declaration
□ Include DOCType Reference
□ Keep runtime namespace prefixes for unmarshal/marshal
□ Use Combination Rule

10 Select the check boxes next to the OTD options you want to enable (see Table 33).

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 168).

Table 33 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. ▪ If this option is *not* 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. ▪ If this option *is* selected, the attribute is ignored and neither of the above occurs.

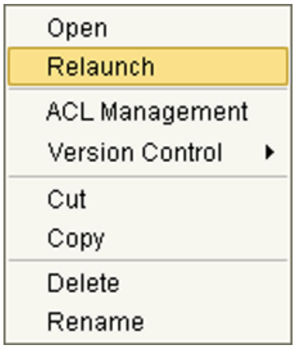
Option	Description
Ignore all attributes	Controls whether or not all attributes are ignored during the unmarshal and marshal processes. If both this option and the *Keep runtime namespace prefixes* option (below) are selected, only namespace attributes will be handled during the unmarshal process and consequently presented in the output during the marshal process. (The *namespace* attribute has the form **xmlns:XX**.)
Include XML declaration	Controls whether or not the XML declaration **<?xml version="1.0" encoding="......"?>** appears in the output during the marshal process. Option is checked by default.
Include DOC Type Reference	Controls whether or not the **"<!DOCTYPE ..."** string appears in the output during the marshal process.
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. - 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. - If this option is *not* 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). **Note:** 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).
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 154). 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 147 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 326 and **Developing Web Services** on page 422).

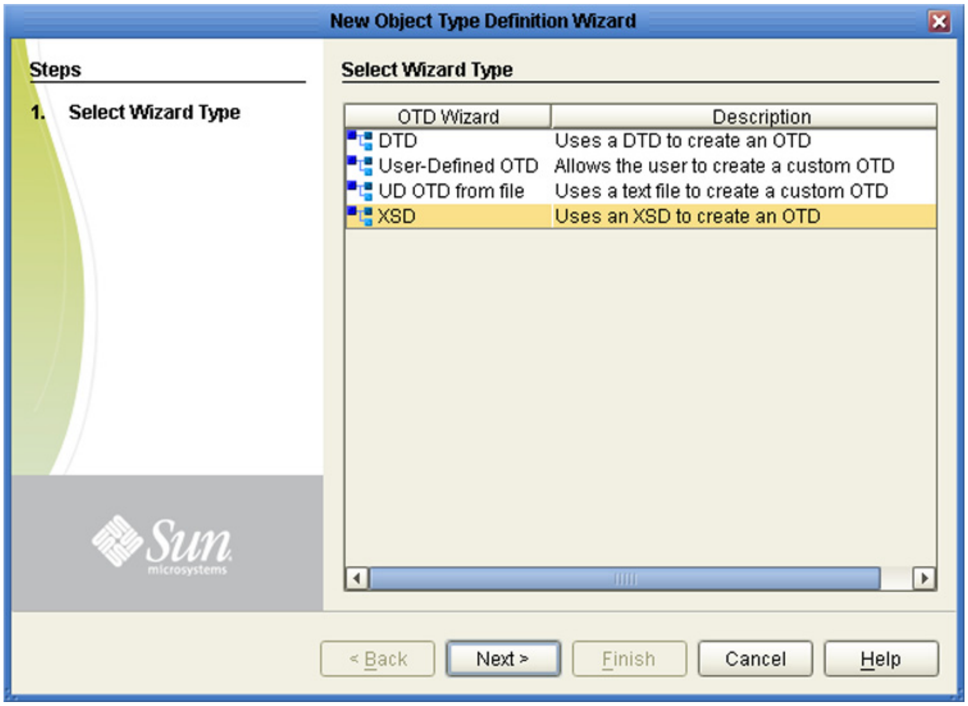
Note: *The* **reset()** *method resets the XSD OTD to its initial conditions, including any default values. It is highly recommended that this method be used in your Collaborations whenever marshaling in a loop, to conserve system resources.*

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 148) to create an OTD from an XSD file.

Figure 148 OTD Wizard Selection: XSD Wizard


New Object Type Definition Wizard

Steps

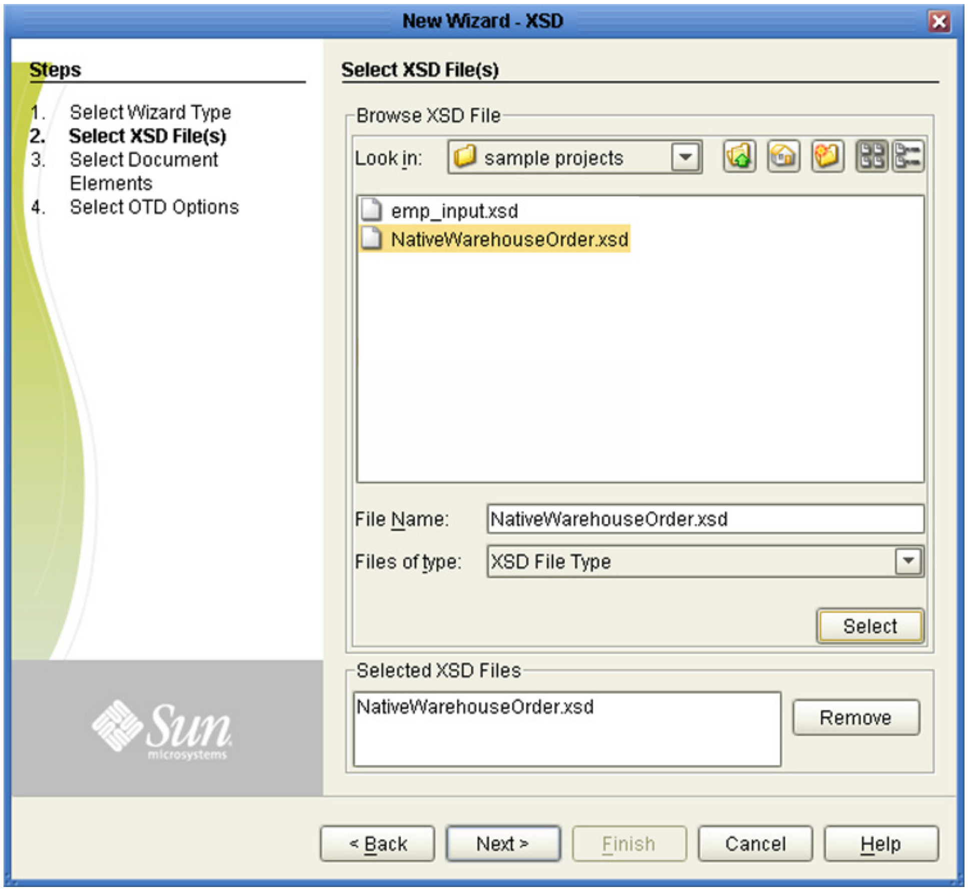
1. Select Wizard Type

Select Wizard Type

OTD Wizard	Description
DTD	Uses a DTD to create an OTD
User-Defined OTD	Allows the user to create a custom OTD
UD OTD from file	Uses a text file to create a custom OTD
XSD	Uses an XSD to create an OTD

2 Click **Next** to display the Select XSD File(s) dialog, shown in Figure 149.

Figure 149 XSD Wizard: Select XSD File(s)



New Wizard - XSD

Steps

1. Select Wizard Type
2. **Select XSD File(s)**
3. Select Document Elements
4. Select OTD Options

Select XSD File(s)

Browse XSD File

Look in: sample projects

emp_input.xsd
NativeWarehouseOrder.xsd

File Name: NativeWarehouseOrder.xsd

Files of type: XSD File Type

Selected XSD Files

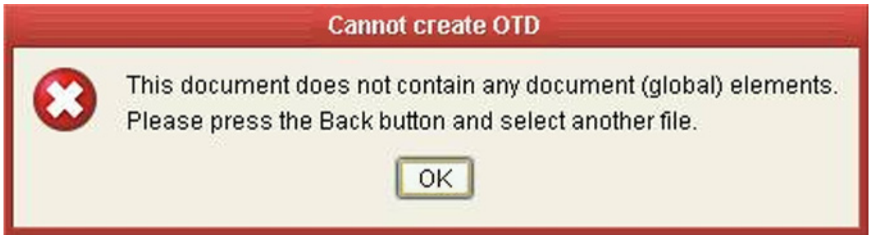
NativeWarehouseOrder.xsd

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 150 will be displayed.

Figure 150 Cannot Create OTD Warning Box



Cannot create OTD

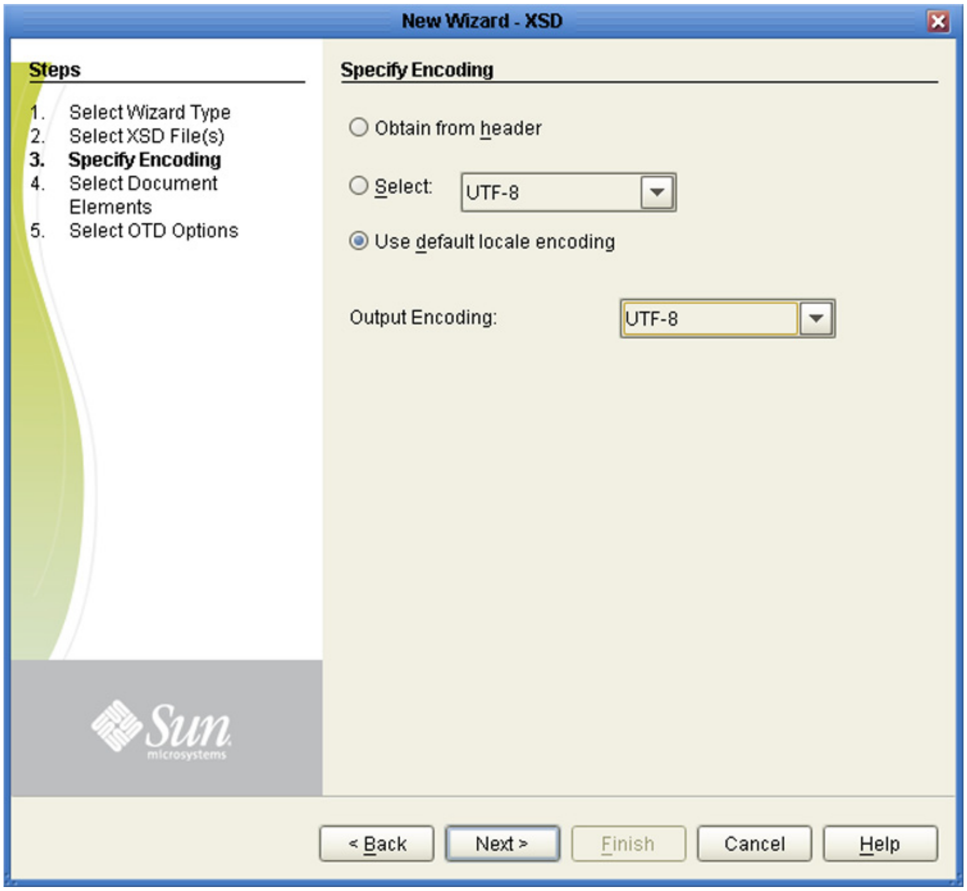
This document does not contain any document (global) elements.
Please press the Back button and select another file.

4 If you are using the extended language options (see **Setting Up Options** on page 55) click **Next** to display the *Specify Encoding* dialog, shown in Figure 151. 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 151 Select Encoding Dialog



New Wizard - XSD

Steps

1. Select Wizard Type
2. Select XSD File(s)
3. **Specify Encoding**
4. Select Document Elements
5. Select OTD Options

Specify Encoding

○ Obtain from header

○ Select: UTF-8

● Use default locale encoding

Output Encoding: UTF-8

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 160).
- Select **Use default locale encoding** to use the default encoding for the locale.

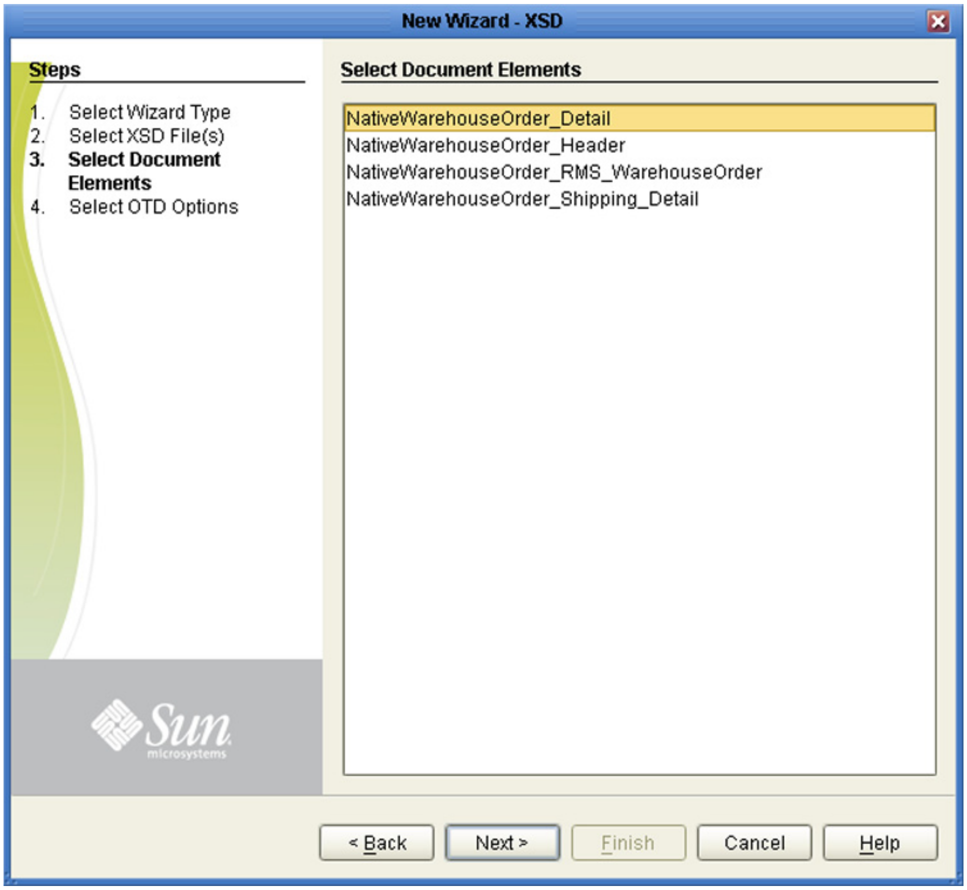
Note: *When opening a data file, if the file is a .xml file with an encoding in the header, the input view encoding will default to the encoding in the .xml file.*

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 160).

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 152.

Figure 152 Select Document Elements Dialog



New Wizard - XSD

Steps

1. Select Wizard Type
2. Select XSD File(s)
3. **Select Document Elements**
4. Select OTD Options

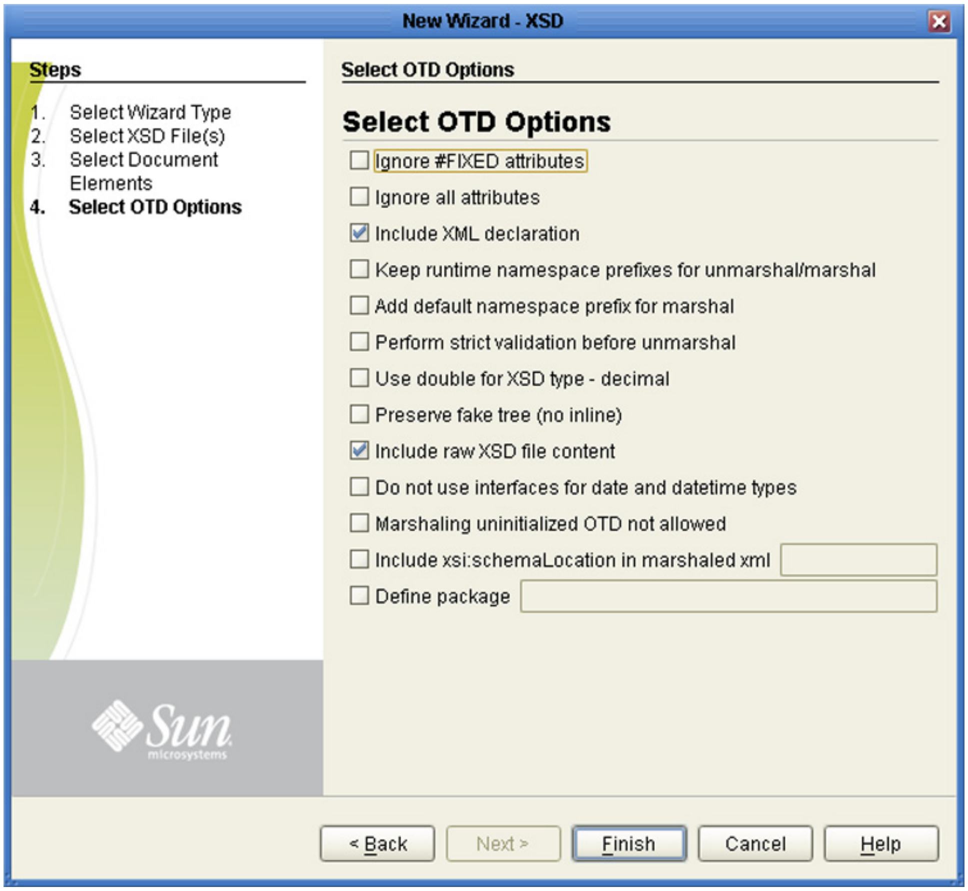
Select Document Elements

NativeWarehouseOrder_Detail
NativeWarehouseOrder_Header
NativeWarehouseOrder_RMS_WarehouseOrder
NativeWarehouseOrder_Shipping_Detail

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 153.

Figure 153 Select OTD Options Dialog



New Wizard - XSD

Steps

1. Select Wizard Type
2. Select XSD File(s)
3. Select Document Elements
4. **Select OTD Options**

Select OTD Options

Select OTD Options

☐ Ignore #FIXED attributes

☐ Ignore all attributes

☑ Include XML declaration

☐ Keep runtime namespace prefixes for unmarshal/marshal

☐ Add default namespace prefix for marshal

☐ Perform strict validation before unmarshal

☐ Use double for XSD type - decimal

☐ Preserve fake tree (no inline)

☑ Include raw XSD file content

☐ Do not use interfaces for date and datetime types

☐ Marshaling uninitialized OTD not allowed

☐ Include xsi:schemaLocation in marshaled xml

☐ Define package

10 Select the check boxes next to the OTD options you want to enable (see Table 34).

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 168).

Table 34 XSD OTD Options

Option	Description
Ignore #FIXED attributes	Controls whether or not attributes defined as FIXED are ignored during the unmarshal and marshal processes. ▪ If this option is *not* 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. ▪ If this option *is* selected, the attribute is ignored and neither of the above occurs.
Ignore all attributes	Controls whether or not all attributes should be ignored in the unmarshal and marshal processes. If both this option and the *Keep runtime namespace prefixes* 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 *namespace* attribute has the form **xmlns:XX**.)
Include XML declaration	Controls whether or not the XML declaration **<?xml version="1.0" encoding="......"?>** 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. ▪ 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. ▪ If this option is *not* 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). **Note:** 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).
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. ▪ If both this flag and the *Keep runtime namespace prefixes* option (above) are selected, then the default target namespace of an element will be applied to the element during the marshal process, *if it is a root element*. ▪ If the *Keep runtime namespace prefixes* option is *not* selected, then the elements are qualified based on the XSD definition and this flag has no effect.

Option	Description
Perform strict validation before unmarshal	Enables strict validation of the following (for elements only): ▪ Union type ▪ List type ▪ Restriction ▪ Fundamental facets (as implied by built-in data types) ▪ Constraining facets See Table 35 for additional information.
Use double for XSD type - decimal	Specifies the use of Java **double** type for the decimal type in the OTD. If not checked (default) Java **BigDecimal** 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 though programming). Checked by default. **Note:** 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: ▪ No default values are set to any fields. ▪ Marshaling will throw an exception if any mandatory fields are not initialized with some value.
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 35 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	-

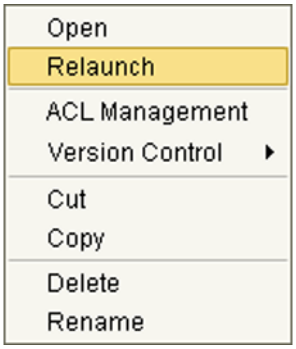
Category	Parameter	Comments
Primitive Datatypes	anyURI	-
	base64Binary	-
	boolean	-
	date	-
	dateTime	-
	decimal	-
	double	-
	duration	-
	float	-
	gDay	-
	gMonth	-
	gMonthDay	-
	gYear	-
	gYearMonth	-
	hexBinary	-
	NOTATION	-
	QName	-
	string	-
	time	-
Constraining Facets	enumeration	-
	fractionDigits	-
	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 154). 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 154 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.

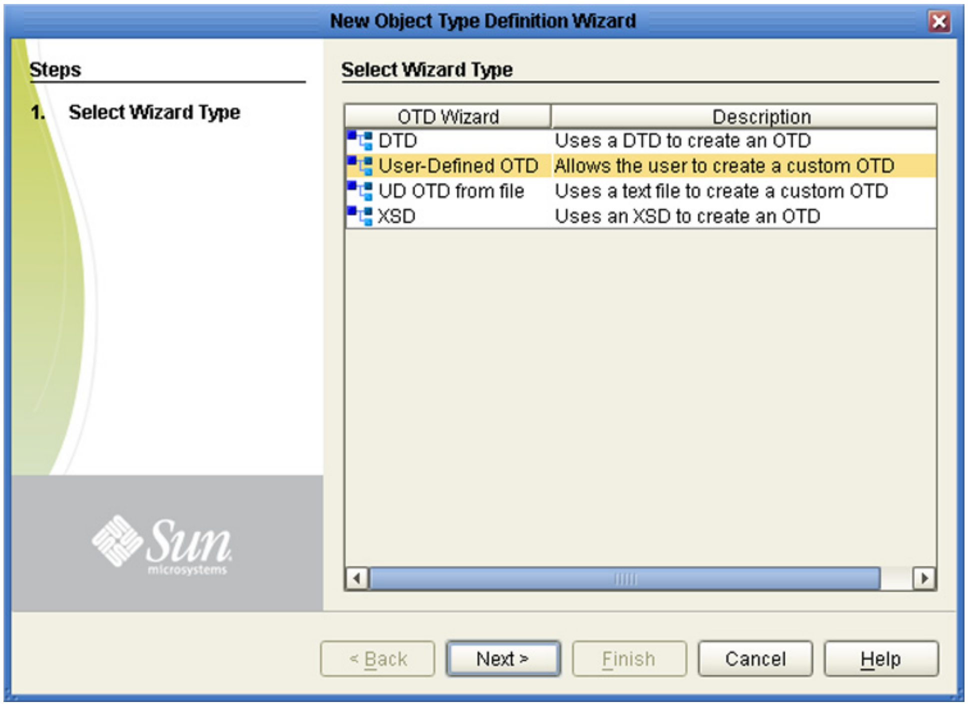
Note: *The **reset()** method resets the User-Defined OTD to its initial conditions, including any default values. It is highly recommended that this method be used in your Collaborations whenever marshaling in a loop, to conserve system resources.*

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 155) to create an OTD without using a source file.

Figure 155 OTD Wizard Selection: User-Defined OTD



New Object Type Definition Wizard

Steps

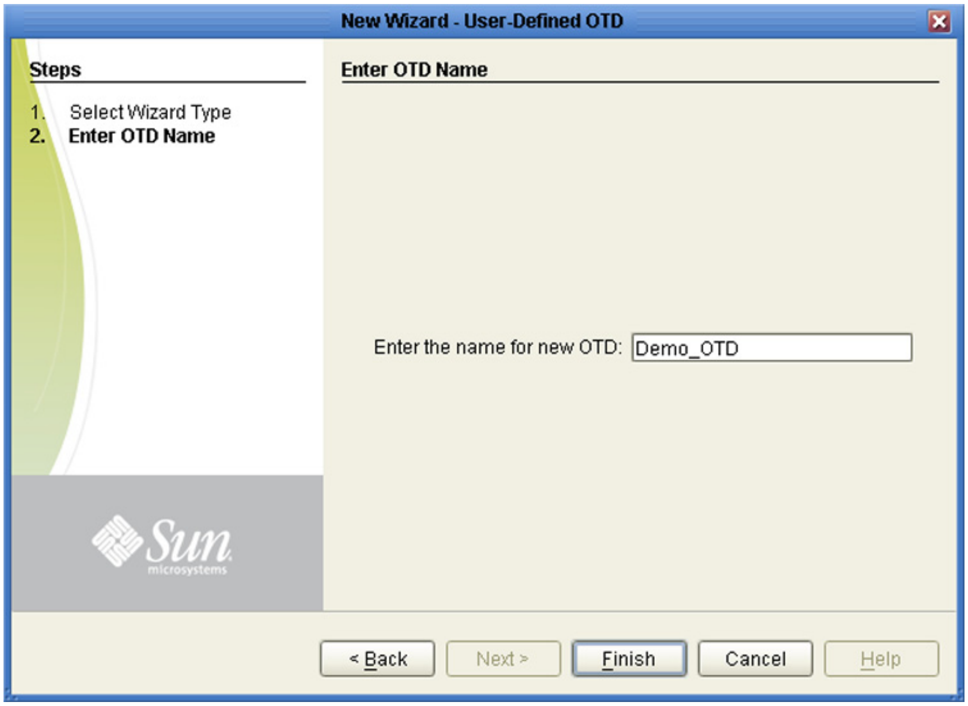
1. Select Wizard Type

Select Wizard Type

OTD Wizard	Description
DTD	Uses a DTD to create an OTD
User-Defined OTD	Allows the user to create a custom OTD
UD OTD from file	Uses a text file to create a custom OTD
XSD	Uses an XSD to create an OTD

2 Click **Next** to display the *Enter OTD Name* dialog, shown in Figure 156.

Figure 156 Enter OTD Name



Enter the name for new OTD: Demo_OTD

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 168). 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 36 Text File Variables

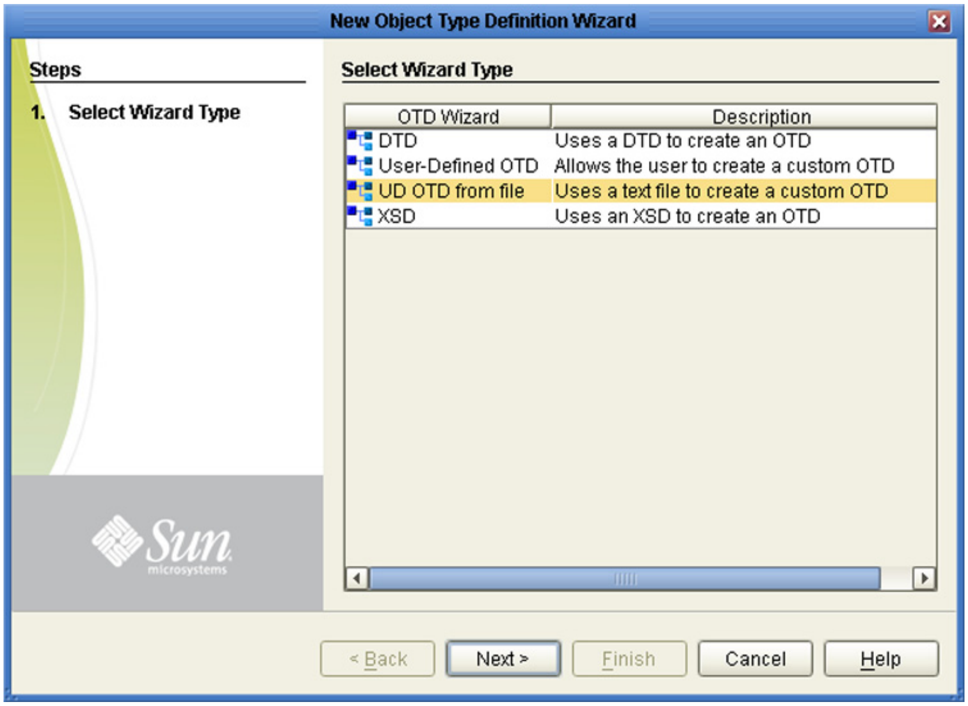
Variable	Value	Meaning
<option>	*	Repeating and optional.
	+	Repeating and not optional.
	?	Optional and not repeating.
<name>	<text>	Field name.
<space>	<none>	A blank space.
<length>	<number>	Fixed-field length.

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 157) to create an OTD using a source file.

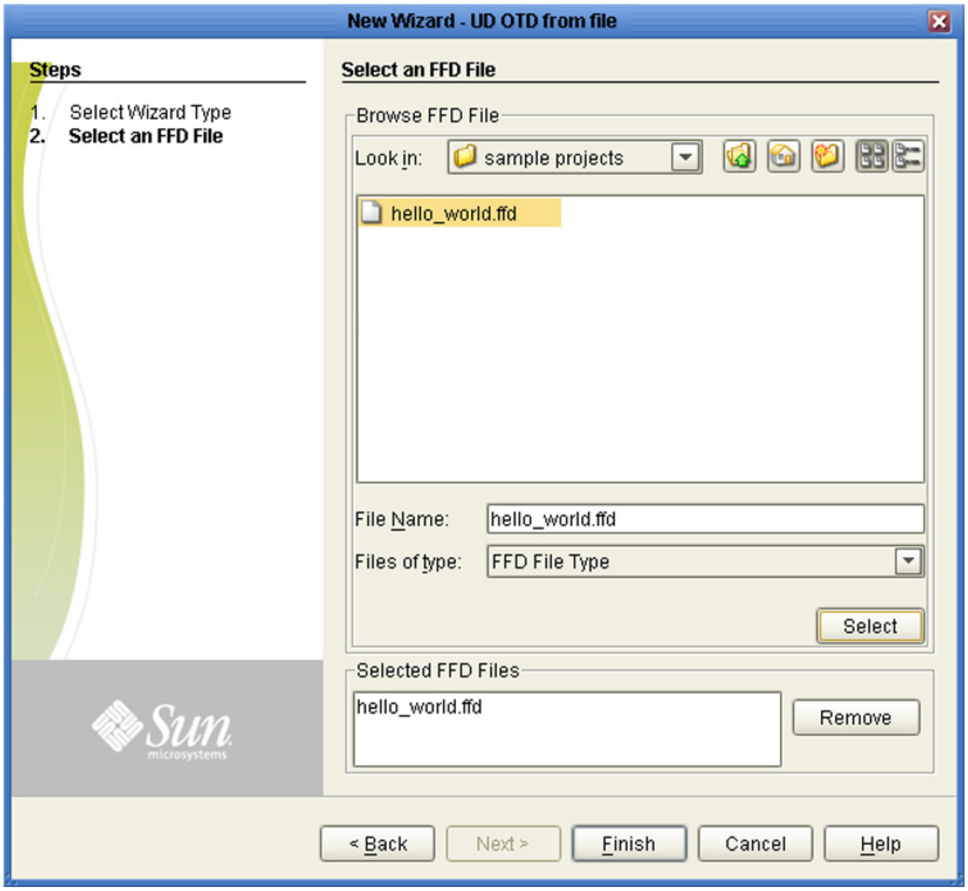
Figure 157 OTD Wizard Selection: UD OTD from File



OTD Wizard	Description
DTD	Uses a DTD to create an OTD
User-Defined OTD	Allows the user to create a custom OTD
UD OTD from file	Uses a text file to create a custom OTD
XSD	Uses an XSD to create an OTD

2 Click **Next** to display the *Select a File* dialog, shown in Figure 156.

Figure 158 Select a File Dialog



Look in: sample projects

File Name: hello_world.ffd

Files of type: FFD File Type

Selected FFD Files: hello_world.ffd

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.

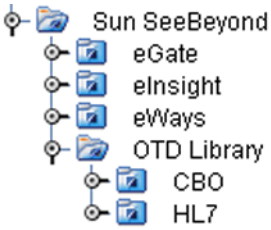
4 Click **Finish** to build the OTD(s) and add them to your Project.

5 The OTD Editor will open, displaying the new OTD(s) (see **Using the OTD Editor** on page 168). 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 159).

Figure 159 Project Explorer - Sun SeeBeyond Folder



Examples of predefined OTDs are listed below:

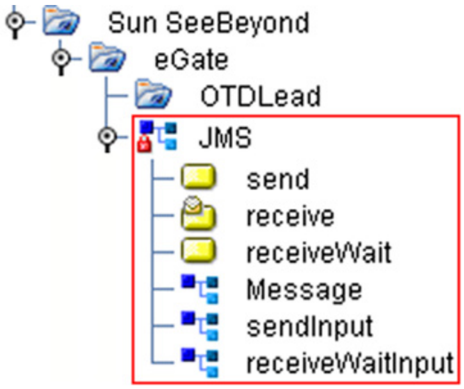
- The *JMS OTD*, which is installed along with Enterprise Designer, resides in the **eGate** folder (see **Using the JMS OTD** on page 206).
- The *Scheduler OTD*, which is installed along with Enterprise Designer, resides in the **eGate** folder (see **Using Scheduler OTDs** on page 209).
- OTD Libraries, which contain collections of predefined OTDs, reside in the **OTD Library** folder.
  - The *CBO OTD Library* is installed along with Enterprise Designer (see **Using the CBO OTD Library** on page 211).
  - Additional OTD libraries are available as separate products and must be installed using the Java CAPS Installer. The *HL7 OTD Library* is shown as an example in Figure 159. These libraries are described in their own user guides.

7.5.1 Using the 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.

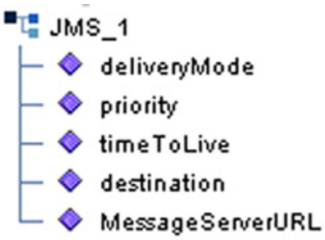
The JMS OTD is included with eGate Integrator, and is installed automatically. The template resides in the **Sun Seebeyond\eGate** folder in Project Explorer (see Figure 160). This is the location to which you browse when you add a JMS OTD to a Collaboration Definition (Java) using the Collaboration Definition Wizard.

Figure 160 Project Explorer - JMS OTD



Expanding the JMS OTD in the Collaboration Definition Editor exposes the property nodes shown in Figure 161.

Figure 161 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 37, and are described in detail in the *Sun SeeBeyond eGate™ Integrator JMS Reference Guide*.

Table 37 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)

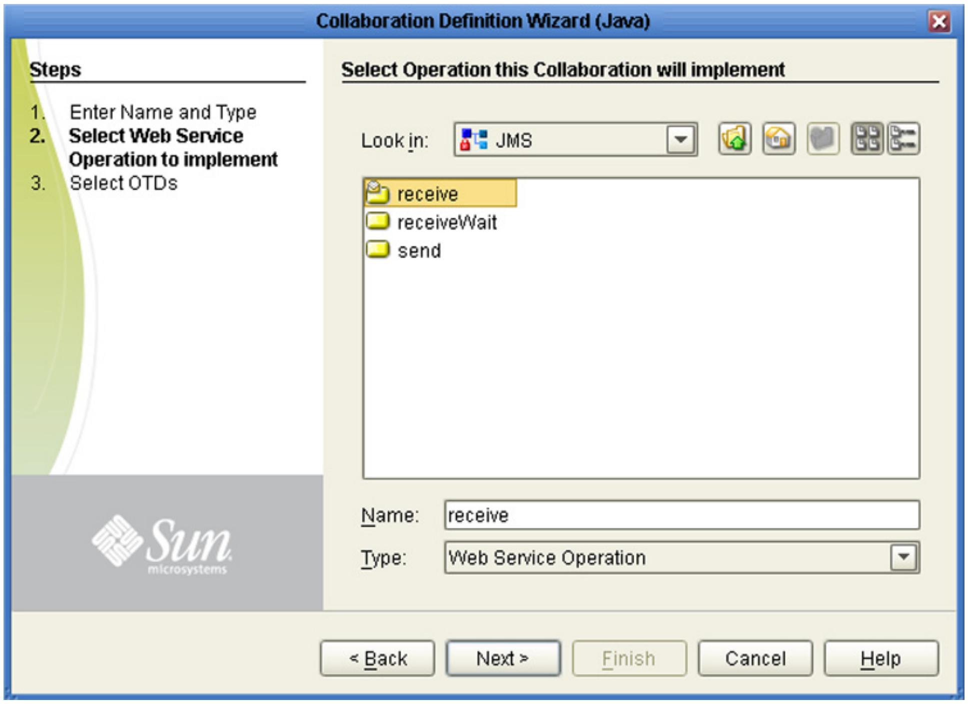
1 In Project Explorer, right-click the selected Project to display its context menu.

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 162).

Figure 162 Existing Web Service: Select Operation


Collaboration Definition Wizard (Java)
Steps
1. Enter Name and Type
2. Select Web Service Operation to implement
3. Select OTDs
Select Operation this Collaboration will implement
Look in:
JMS
receive
receiveWait
send
Sun microsystems
< Back
Next >
Finish
Cancel
Help

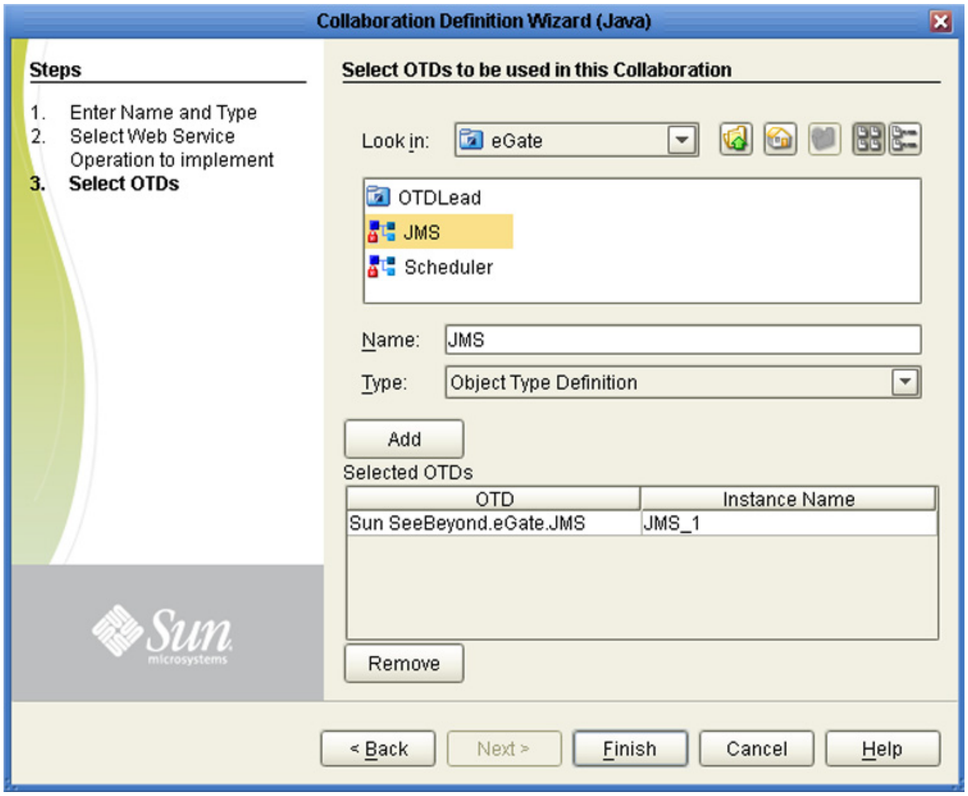
Name: receive

Type: Web Service 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 163).

Figure 163 Existing Web Service: Select OTDs



Collaboration Definition Wizard (Java)

Steps

1. Enter Name and Type
2. Select Web Service Operation to implement
3. **Select OTDs**

Select OTDs to be used in this Collaboration

Look in: eGate

- OTDLead
- JMS
- Scheduler

Name: JMS

Type: Object Type Definition

Selected OTDs

OTD	Instance Name
Sun SeeBeyond.eGate.JMS	JMS_1

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*.

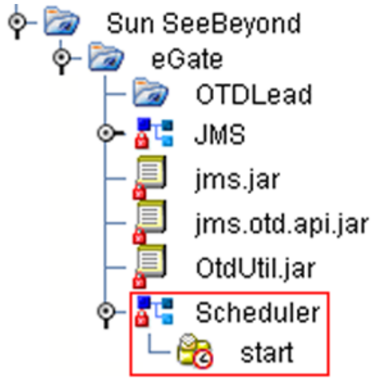
For more information on creating Java-based Collaboration Definitions, see **Creating a Java-based Collaboration Definition** on page 326.

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 123).

The Scheduler OTD is included with eGate Integrator, and is installed automatically. The template resides in the **Sun Seebeyond\eGate** folder in Project Explorer, below the JMS OTD and various JAR files (see Figure 160). This is the location you browse to when you add a Scheduler OTD to a Collaboration Definition (Java) using the Collaboration Definition Wizard.

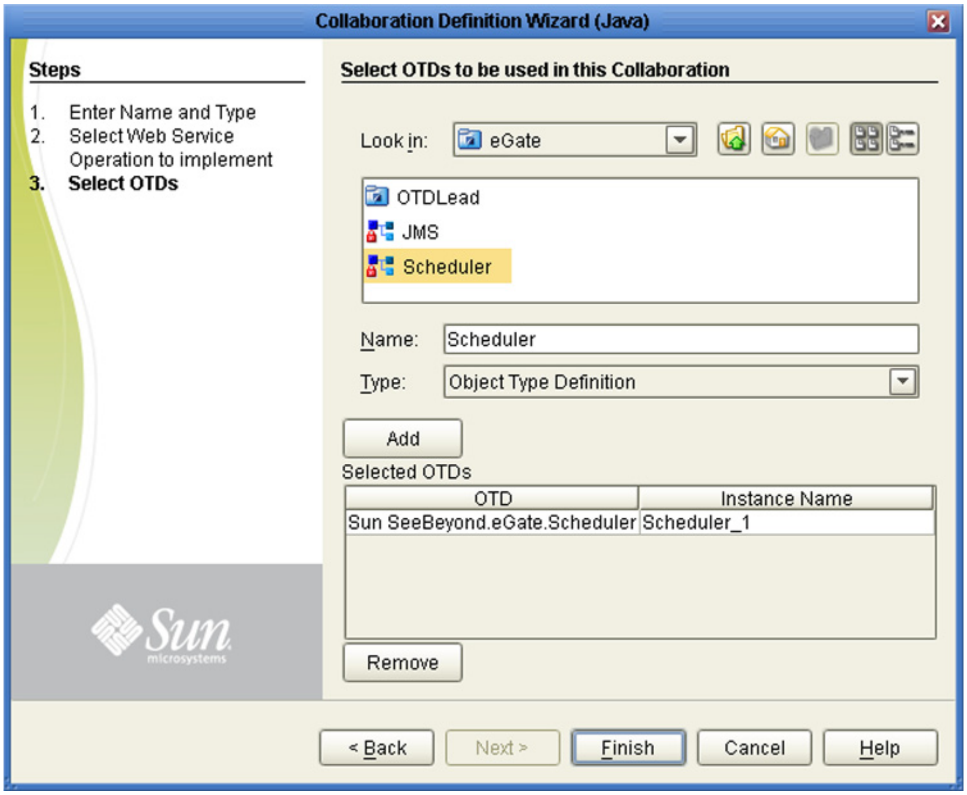
Figure 164 Project Explorer - Scheduler OTD



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 163).

Figure 165 Existing Web Service: Select OTDs


Collaboration Definition Wizard (Java)
Steps
1. Enter Name and Type
2. Select Web Service Operation to implement
3. Select OTDs
Select OTDs to be used in this Collaboration
Look in:
OTDLead
JMS
Scheduler
Add
Remove
< Back
Next >
Finish
Cancel
Help

Look in: eGate

Name: Scheduler

Type: Object Type Definition

Selected OTDs

OTD	Instance Name
Sun SeeBeyond.eGate.Scheduler	Scheduler_1

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 326.

7.5.3 Using the CBO OTD Library

The Canonical Business Object (CBO) OTD Library is a set of templates corresponding to the Open Applications Group's Integration Specification (OAGIS). This Library consists of a set of approximately 300 predefined OTDs, and is included with eGate Integrator.

OAGIS provides a horizontal business language that provides an open interface for interaction with vertical industry standards. The OAGIS CBOs embody a set of standardized objects to implement this interaction. Extensive information is available from the Open Applications Group's web site **http://www.openapplications.org.**

Installing the CBO OTD Library

The CBO OTD Library is included as a standard component in eGate Integrator, and requires uploading from the DVD to the Repository and then downloading to eGate.

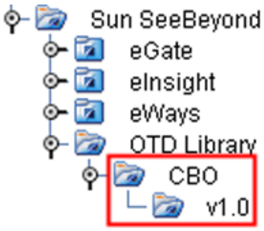
To install the CBO OTD Library

1. Follow the instructions for installing eGate Integrator as described in the *Sun Java™ Composite Application Platform Suite Installation Guide*.

   - After uploading **eGate Integrator** to the Repository, select **CBO_OTD_v1_0** from the list of uploadable components (along with any other components you need).
   - If eGate Integrator is already installed, you can add the library by following the instructions given in the **Installing New Product Components** chapter of the *Sun Java™ Composite Application Platform Suite Installation Guide*.

2. Once installed, the CBO OTD Library appears in the **Sun SeeBeyond > OTD Library** folder in Project Explorer (see Figure 166).

Figure 166 CBO OTD Library in Project Explorer



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 113, with the addition of the **Relaunch** option for DTD and XSD OTDs shown in Figure 167.

Figure 167 OTD Context Menu

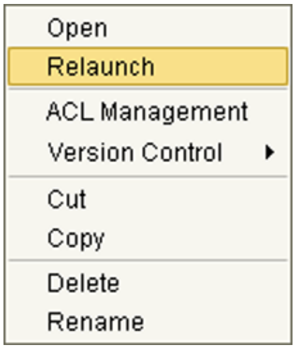


Table 38 OTD Context Menu Options

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 *Sun SeeBeyond eGate™ Integrator System Administration Guide*.
Version Control	Accesses the version control features for the selected Object Type Definition. See **Accessing Component Version Control Features** on page 82 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 within the same branch (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 **Moving and Replicating Repository Objects** on page 100 for additional information.

Table 38 OTD Context Menu Options

Option	Function
Copy	Copies the selected Object Type Definition, after which you can paste it to other Projects within the same branch (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 **Moving and Replicating Repository Objects** on page 100 for additional information.
Delete	Deletes the selected Object Type Definition, subject to the following conditions: ▪ You have *write* privileges for the Object Type Definition (see ACL Management, above). ▪ You have the Object Type Definition checked out to your workspace. ▪ The Object Type Definition is not checked out by anyone other than yourself. If these conditions are met, a dialog is presented in which you confirm that you want to delete the selected Object Type Definition. Clicking **Yes** then deletes the Object Type Definition.
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 102).*

Chapter 8

Object Type Definitions III

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

What's in This Chapter

- **Overview** on page 214
- **Creating and Managing OTD Nodes** on page 215
- **Creating and Using Templates** on page 219
- **Editing OTD Properties** on page 224
- **Specifying the Node Type** on page 230
- **Specifying Read/Write Access** on page 232
- **Matching Data Patterns** on page 233
- **Specifying Delimiters** on page 235
- **Creating a Delimiter List** on page 246

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 200, and the OTD Editor, which is described in **Using the OTD Editor** on page 168.

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 168). 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 218).

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

Figure 168 OTD Node Context Menu

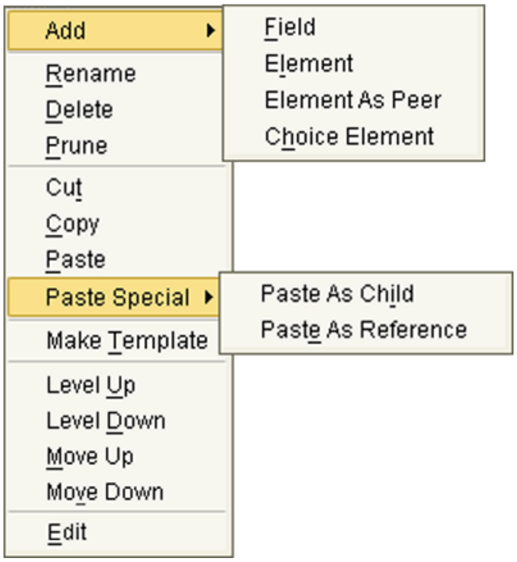


Table 39 OTD Node Context Menu Options

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. **Note:** Renaming the root node renames the OTD.
Delete		When a node is deleted, both the node and its descendants are removed.

Table 39 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 **Cutting, Copying, and Pasting Nodes** on page 218 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 **Cutting, Copying, and Pasting Nodes** on page 218 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 **Cutting, Copying, and Pasting Nodes** on page 218 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. *Paste As Child* is disabled for field nodes. See **Cutting, Copying, and Pasting Nodes** on page 218 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 **Cutting, Copying, and Pasting Nodes** on page 218 for additional information.
Make Template		Copies the selected node and its descendants and pastes them into the *Reference* panel as an internal template. It appears as a new OTD, with the selected node as the root. See **Creating and Using Templates** on page 219 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 **Moving Nodes** on page 217).
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 **Moving Nodes** on page 217).
Move Up		Moves the selected node up one line in the tree structure, at the same hierarchical level (see **Moving Nodes** on page 217).
Move Down		Moves the selected node down one line in the tree structure, at the same hierarchical level (see **Moving Nodes** on page 217).

Table 39 OTD Node Context Menu Options

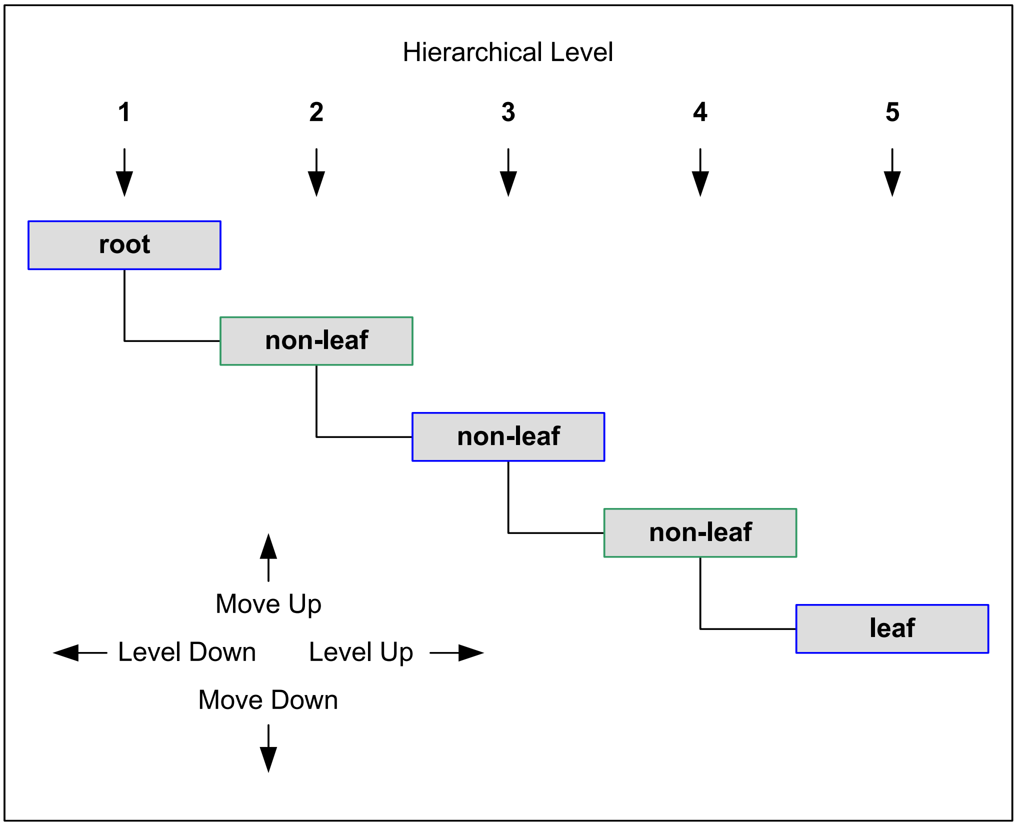
Option	Action
Edit	For external templates only, enables the properties for selected node to be edited (see **Cutting, Copying, and Pasting Nodes** on page 218).

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 39. To supplement these descriptions, the respective actions of these options is illustrated in Figure 169.

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 169 Node-moving Options



Important: *If you move an OTD node, you must reset the* **nodeType** *for that node. See* **Editing OTD Properties** *on page 224.*

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.

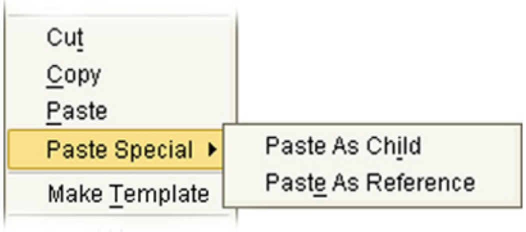
Note: *If nodes in the template are delimited but not specified individually, the delimiter properties in each pasted instance change to reflect the new parentage settings.*

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 170).

Figure 170 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 222.

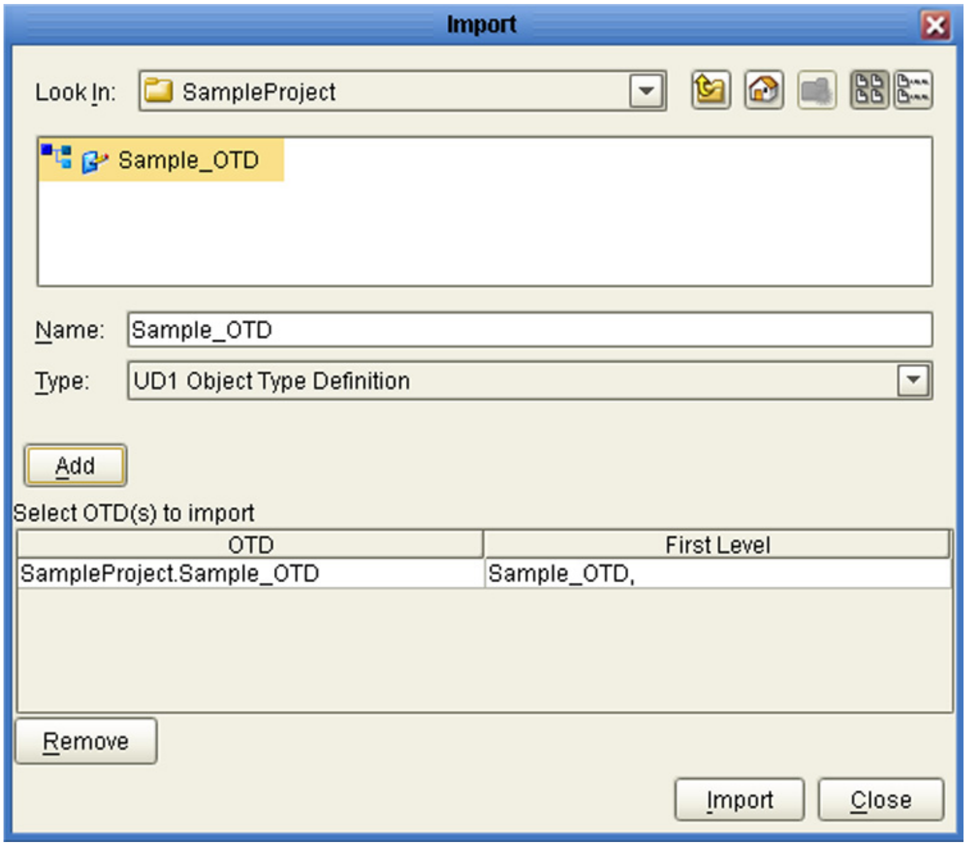
8.3.1 External Templates

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 169) to display the *Import* dialog, shown in Figure 171.

Figure 171 Import Dialog



Look In: SampleProject

Name: Sample_OTD

Type: UD1 Object Type Definition

Select OTD(s) to import

OTD	First Level
SampleProject.Sample_OTD	Sample_OTD,

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

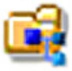
In the OTD Editor, different icons are used to differentiate between nodes that belong to the OTD itself, and those belonging to either internal or external templates. This information is important when you are contemplating a change to a node, since changing a node belonging to a template changes all instances of it. See also **Creating and Using Templates** on page 219.

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 172).

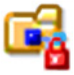
Figure 172 Element Node Created for OTD



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 173. 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 219).

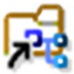
Figure 173 Element Node from External Template



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 174. All Element nodes within an internal template, however, are represented by the icon shown above in Figure 172.

Figure 174 Element Node from Internal Template

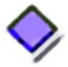


Field Nodes

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 175).

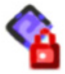
Figure 175 Field Node Created for OTD



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 176. 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 219).

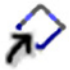
Figure 176 Field Node from External Template



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 177. All Field nodes within an internal template, however, are represented by the icon shown above in Figure 175.

Figure 177 Internal Template - Field Node



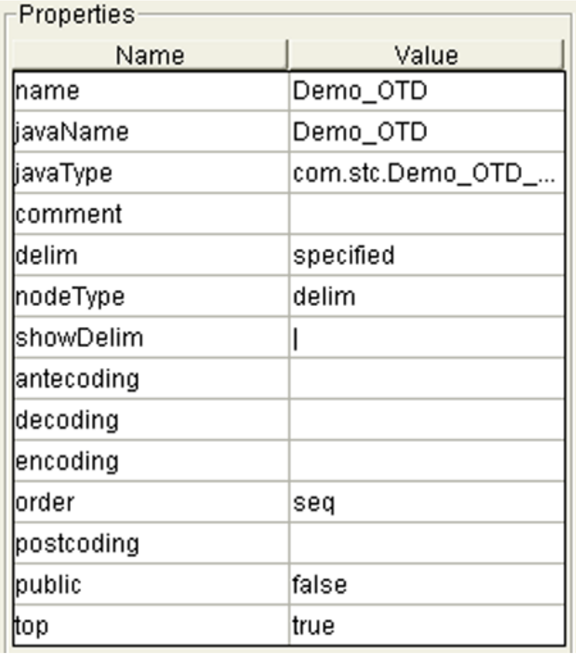
8.4 Editing OTD Properties

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 178.

Figure 178 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 40 Root Node Properties

Name	Description
name	Node display name (see **Understanding OTDs** on page 155).
javaName	Property accessor basename, automatically generated — *do not modify*. See **Understanding OTDs** on page 155.
javaType	Java type; automatically assigned, not editable.
comment	Free-form text (no run-time effect).
delim	Delimiter specification (see **Specifying Delimiters** on page 235). Default value is **not set**.
nodeType	Specifies the marshal/unmarshal format (see **Specifying the Node Type** on page 230). The default value is **delim**.

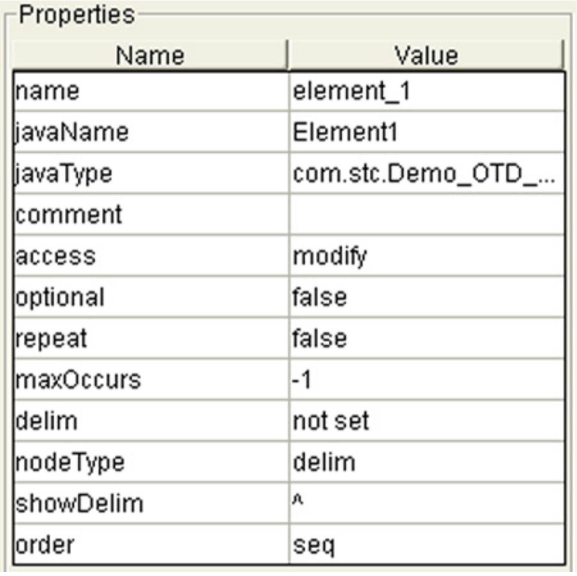
Name	Description
showDelim	Displayed only if **nodeType** is *delim*. The default value is **-none-**; once delimiters are specified, value field shows the delimiters. (Root nodes are not typically delimited, however.)
antecoding	Specifies the *input data* coding (see **Specifying Data Conversion and Encoding** on page 160). If this property is not specified, the value specified for the **decoding** property will be used for the input data. This property is displayed only when the *top* property is set to **true**.
decoding	Specifies the *unmarshal* coding (see **Specifying Data Conversion and Encoding** on page 160). (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 *top* property is set to **true**.
encoding	Specifies the *marshal* coding (see **Specifying Data Conversion and Encoding** on page 160). This property is displayed only when the *top* property is set to **true**.
length	Displayed only if **nodeType** is *fixed*. Specifies the length of the field; the default value is **0**.
order	Specifies the ordering of the root node's children. ▪ **seq** specifies that the child nodes must appear in sequence. ▪ **any** specifies that the child nodes can appear in any order. ▪ **mix**
postcoding	Specifies the *output data* coding (see **Specifying Data Conversion and Encoding** on page 160). If this property is not specified, the value specified for the **encoding** property will be used for the output data. This property is displayed only when the *top* property is set to **true**.
public	For future use, not currently active; **false** by default.
top	Specifies whether or not marshal/unmarshal is supported (**true** or **false**). The default value is **true**.

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 179.

Figure 179 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 41 Element Node Properties

Name	Description
name	Element display name.
javaName	Property accessor basename, automatically generated — *do not modify*.
javaType	Java type; automatically assigned, not editable.
comment	Free-form text (no run-time effect).
access	Access specification (see **Specifying Read/Write Access** on page 232).
optional	Specifies whether or not the node can be absent from an instance. Clicking the *Value* field toggles between **true** and **false**. Not applicable if the element is the child of a *choice element* node.
repeat	Specifies whether or not the node can appear multiple times. Clicking the *Value* field toggles between **true** and **false**. Not applicable if the element is the child of a *choice element* 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 **Specifying Delimiters** on page 235).

Name	Description
nodeType	Governs the marshal/unmarshal format. The default value is **delim**, but is defaulted to parent nodeType if parent nodeType is *fixed* or *trans*.
showDelim	Displayed only if **nodeType** is *delim*. The default value is **-none-**; once delimiters are specified, value field shows the delimiters. Does not apply to *choice element* nodes.
length	Displayed only if **nodeType** is *fixed*. Specifies the length of the field; the default value is **0**.
order	Specifies the ordering of the selected node's children during the unmarshaling process. ▪ **seq** specifies that the child nodes must appear in the sequence given in the metadata. ▪ **any** specifies that the child nodes must remain grouped, but the groups can appear in any order. ▪ **mix** specifies that the child nodes can appear in any order. See the illustration below for additional information. **Note:** Does not apply to *choice element* nodes.

Order Property

To illustrate how the **order** property works, consider the simple tree structure shown in Figure 180, 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 42.

Figure 180 Order Property Example

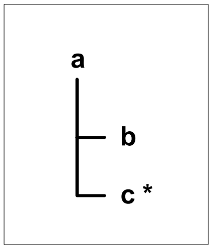


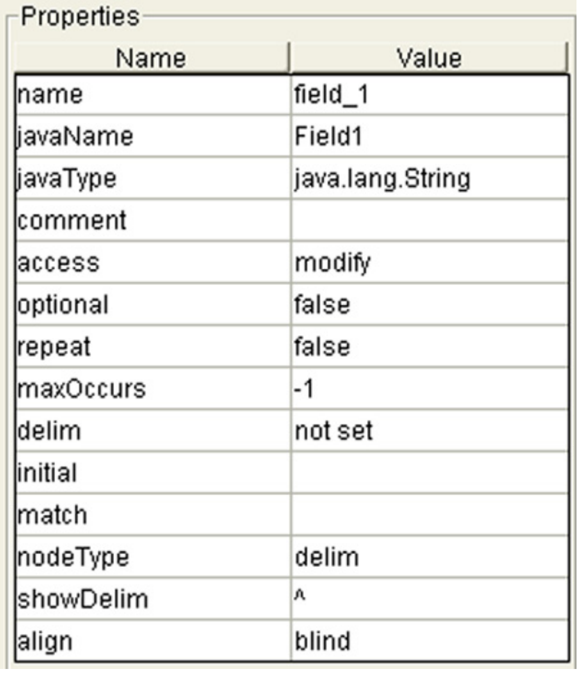
Table 42 Order Property Example

Value	Allowed Node Order
seq	b, c1, c2
any	b, c1, c2, **or** c1, c2, b
mix	b, c1, c2, **or** c1, c2, b, **or** c1, b, c2

8.4.2 Field Node Properties

The set of properties associated with Field nodes is shown in Figure 181.

Figure 181 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 43 Field Node Properties

Name	Description
name	Field display name.
javaName	Property accessor basename, automatically generated — *do not modify*.
javaType	Java type; can be either **java.lang.String** or byte array (**byte[]**).
comment	Free-form text (no run-time effect).
access	Access specification (see **Specifying Read/Write Access** on page 232).
optional	Specifies whether or not the field can be absent from an instance. Clicking the *Value* field toggles between **true** and **false**. Not applicable if the field is the child of a *choice element* node.
repeat	Specifies whether or not the node can appear multiple times. Clicking the *Value* field toggles between **true** and **false**. Not applicable if the field is the child of a *choice element* 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 **Specifying Delimiters** on page 235).

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 **nodeType** is *delimited* or *fixed*, defines a byte pattern used to perform a match during the unmarshal operation, as further specified by the **align** property (see **Defining Byte Patterns** on page 233).
nodeType	Specifies the marshal/unmarshal format. The default value is **delim**, but is defaulted to parent nodeType if parent nodeType is *fixed* or *trans*.
showDelim	Displayed only if **nodeType** is *delim*. The default value is **-none-**; once delimiters are specified, value field shows the delimiters.
align	Specifies the byte alignment criteria for the **match** property (see **Specifying Pattern Alignment** on page 233).
decoding	Displayed only if **nodeType** is *fixed*. Specifies the *unmarshal* coding (see **Specifying Data Conversion and Encoding** on page 160). (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 **nodeType** is *fixed*. Specifies the *marshal* coding (see **Specifying Data Conversion and Encoding** on page 160).
length	Displayed only if **nodeType** is *fixed*. Specifies the length of the field; the default value is **0**.

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 182). Descriptions of the property options are listed in Table 44.

Figure 182 nodeType Property Menu (Root/Element Example)

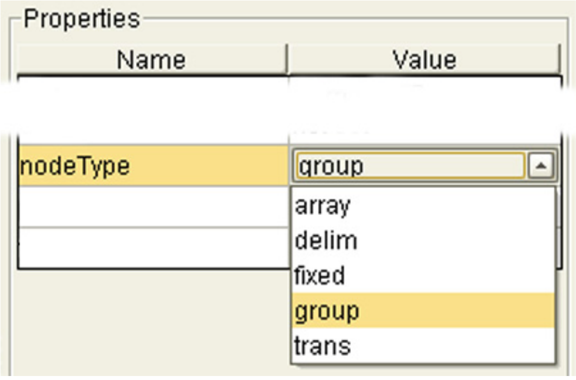


Table 44 nodeType Property Options

Option	Description
alter	Applies only to *Choice Element* nodes. Alter (*alternate*) 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 *repeat* delimiter. The last occurrence may be terminated by a *normal* delimiter. Does not apply to *Choice Element* nodes.
delim	Delim (*delimited*) structure. If repeated, occurrences are separated by a *normal* delimiter. Does not apply to *Choice Element* nodes. See **Specifying Delimiters** on page 235.
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 *Choice Element* nodes.
group	Group provides organizational grouping for purposes such as repetition. Does not apply to *Choice Element* or *Field* nodes.
trans	Trans (*transient*) appears only in an internal tree as a scratchpad field. It does not appear in external data representation, and can only have *trans* 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 (seeTable 45). This rule does not apply to *Choice Element* nodes.

Table 45 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 183 Access Property Menu

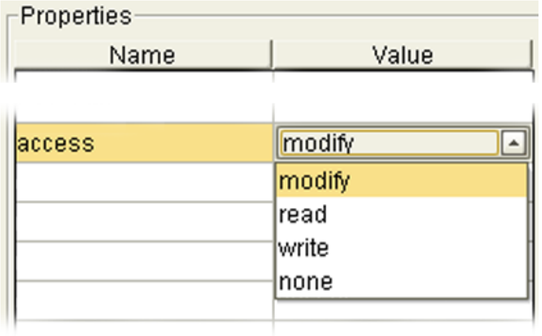


Table 46 Access Parameter Options

Option	Description
modify	Read and write access, generates both *get* and *set* methods. This is the default setting.
read	Read access only, generates *get* method.
write	Write access only, generates *set* method.
none	Neither read nor write access, does not generate either *get* or *set* method.

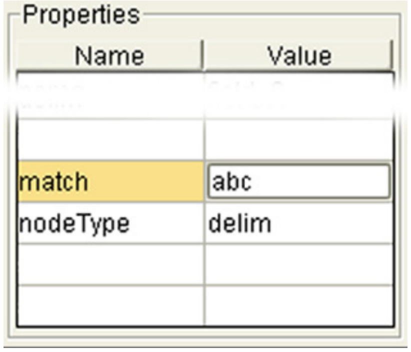
8.7 Matching Data Patterns

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 184 (provides a reference for Table 47).

Figure 184 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 185 Align Property Menu

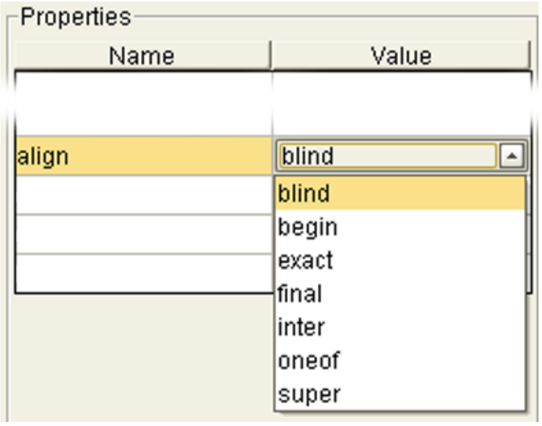


Table 47 Align Parameter Options

Option	Description
blind	Always performs a match (default value). Any value set for the **match** property is ignored.
begin	When the leading bytes of an input byte sequence match the value set for the **match** property (for example, [**abc……**]), 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, [**abc**]), 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 **match** property (for example, [**……abc**]), 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 **match** property, (for example, [**…abc…**]), the unmarshal method matches the field to the input byte sequence.
oneof	If the value set for the **match** property is a repeating pattern of the form **<separator><value>…** (for example, [**\mon\wed\fri**]), and the input byte sequence contains a byte pattern that matches one of the **<value>** entries (for example, [**wed**]), 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 **match** property (for example, [**bc**]), 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

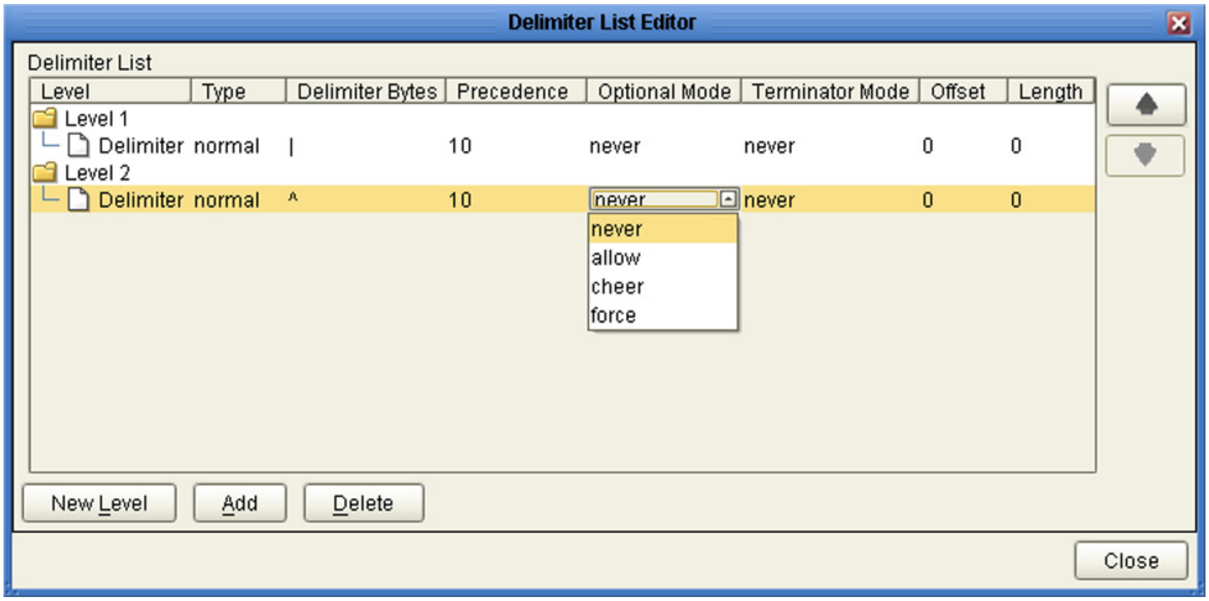
You can define a set of delimiters — a *delimiter list* — for any node in the hierarchical data structure. This delimiter list is used in the external data representation for that node and its descendents. A delimiter list defined for any node overrides the effect of any ancestor node's delimiter list on the node and its descendents.

Delimiters are defined using the Delimiter List Editor (see Figure 186). 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 246.

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 186.

- Type
- Optional Mode
- Terminator Mode

Figure 186 Delimiter List Editor



Level	Type	Delimiter Bytes	Precedence	Optional Mode	Terminator Mode	Offset	Length
Level 1							
Delimiter	normal	|	10	never	never	0	0
Level 2							
Delimiter	normal	^	10	never	never	0	0

Table 48 Delimiter List Editor Command Buttons

Button	Command	Action
New Level	New Level	Adds a new level after the selected level.
Add	Add Delimiter	Adds a new delimiter after the selected delimiter, or to the bottom of list under the selected level.
Delete	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	Close	Closes the editor and saves your entries.

8.8.2 Delimiter Properties

Table 49 Delimiter Properties

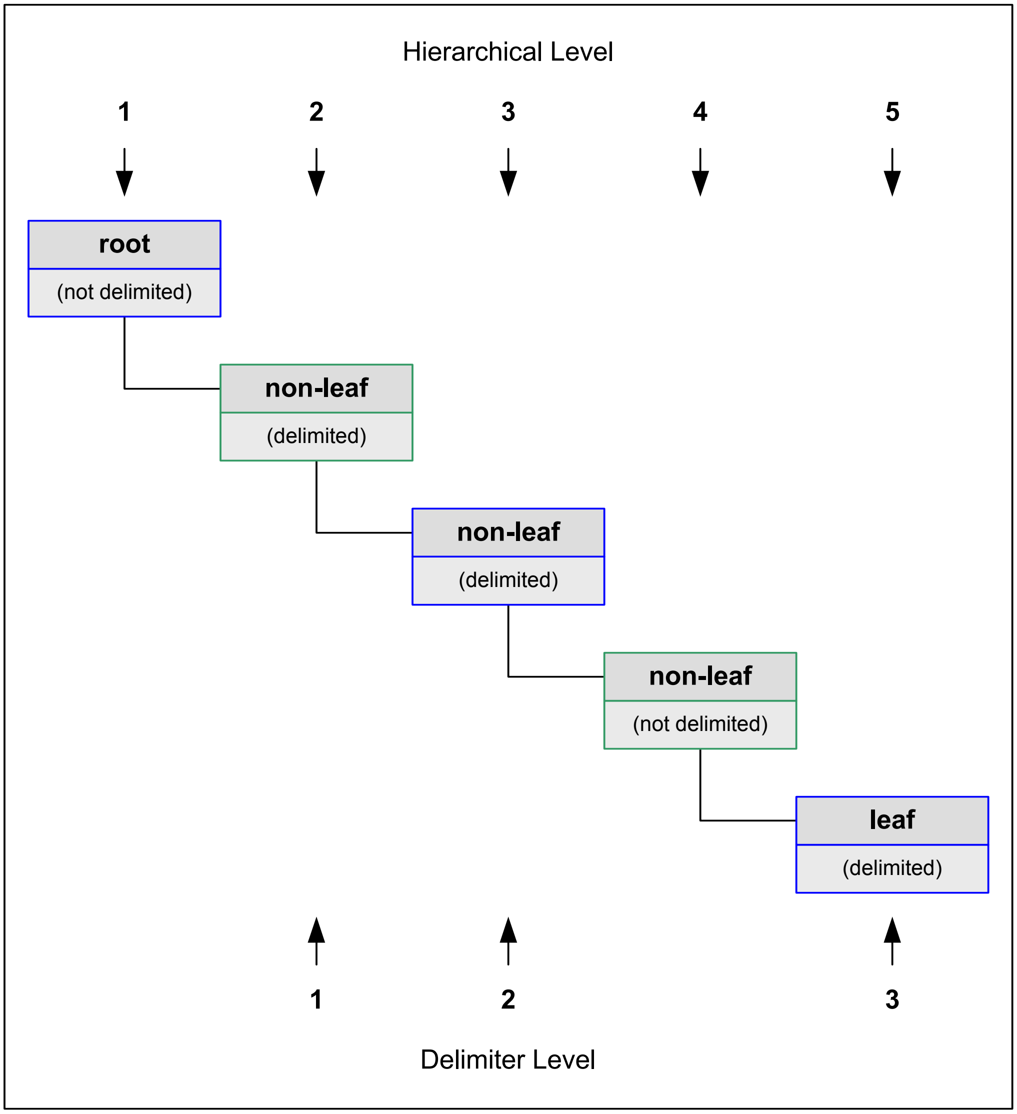
Property	Description
Level	Delimiter levels, assigned consecutively to delimited nodes in the OTD node hierarchy (see **Delimiter Levels** on page 237).
Type	See **Delimiter Type** on page 239.
Delimiter Bytes	Delimiter characters (see **Delimiter Bytes** on page 240). **Note:** Entering a value for *Length* (see below) clears this field.
Precedence	See **Precedence** on page 242.
Optional Mode	See **Optional Mode** on page 243.
Terminator Mode	See **Terminator Mode** on page 245.
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 **0**.
Length	Length of the data field in bytes. Value must be positive integer. Entering a value clears the *Delimiter Bytes* 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 187 and the following example.

Figure 187 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 246 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 50 Delimiter Type Options

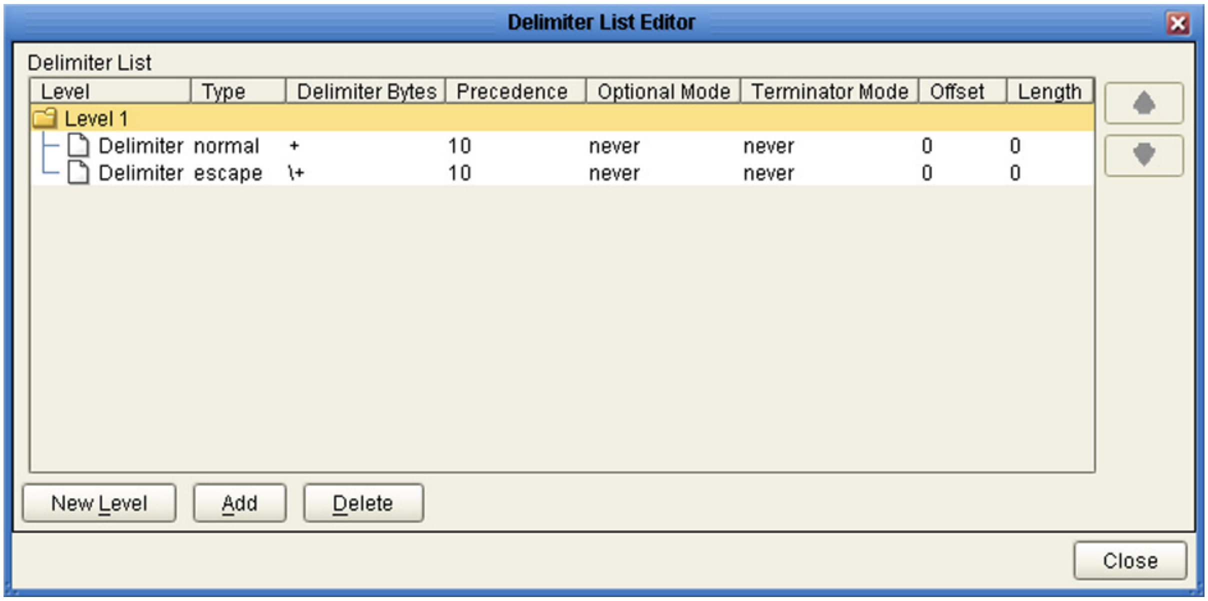
Option	Description
escape	Escape sequence.
repeat	Array separator.
normal	Terminator.

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 188), 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 188 Delimiter Type - Escape



Level	Type	Delimiter Bytes	Precedence	Optional Mode	Terminator Mode	Offset	Length
Level 1							
Delimiter	normal	+	10	never	never	0	0
Delimiter	escape	\+	10	never	never	0	0

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 239). 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 51 lists the currently supported escape sequences.

Table 51 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**.

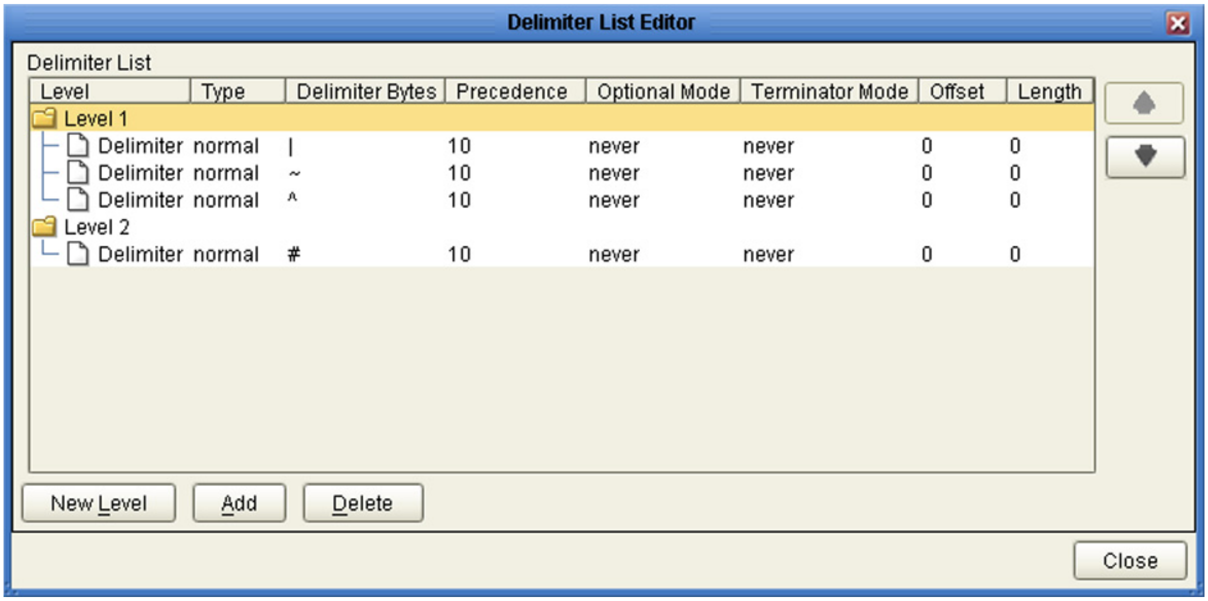
8.8.6 Multiple Delimiters

You can specify multiple delimiters at a given level; for example, if you specify **|**, **~**, and **^** as delimiters for a specific level (see Figure 189), 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 189 Multiple Delimiter Example



Level	Type	Delimiter Bytes	Precedence	Optional Mode	Terminator Mode	Offset	Length
Level 1							
Delimiter	normal	|	10	never	never	0	0
Delimiter	normal	~	10	never	never	0	0
Delimiter	normal	^	10	never	never	0	0
Level 2							
Delimiter	normal	#	10	never	never	0	0

8.8.7 Precedence

Precedence (see **Figure 199 on page 249**) 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.

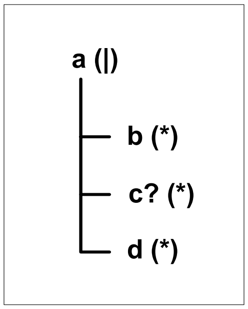
Table 52 Optional Mode Options

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. **Note:** Only this option allows trailing delimiters for a sequence of absent optional nodes.

As illustrative examples, consider the tree structures shown in Figure 190 and Figure 191, 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.)

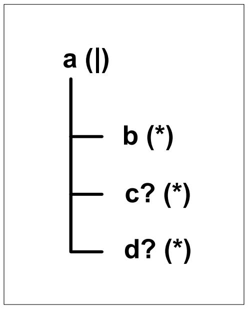
Figure 190 Optional Mode Property (Example 1)



Option	Input	Output
never	**b*d|**	**b*d|**
allow	**b**d|**	**b*d|**
cheer	**b**d|**	**b**d|**
force	**b**d|**	**b**d|**

- **Example 2:** Child nodes **c** and **d** are both *optional*.

Figure 191 Optional Mode Property (Example 2)



Option	Input	Output
never	**b|**	**b|**
allow	**b|**, **b*|**, or **b**|**	**b|**
cheer	**b|**, **b*|**, or **b**|**	**b**|**
force	**b**|**	**b**|**

8.8.9 Terminator Mode

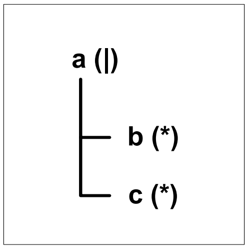
The *Terminator Mode* property specifies how delimiters are to be handled for a specific terminator node in the OTD tree.

Table 53 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 192, where the node **a** has a pipe (**|**) as its delimiter, and its child nodes **b** and **c** have asterisks (*) as their delimiters.

Figure 192 Terminator Mode Property Example

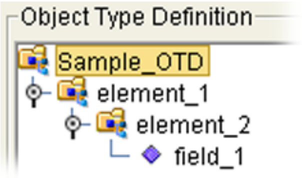


Option	Input	Output
never	**c|**	**c|**
allow	**c|** or **c*|**	**c|**
cheer	**c|** or **c*|**	**c*|**
force	**c*|**	**c*|**

8.9 Creating a Delimiter List

As an example, we shall create a delimiter list for the simple OTD structure shown in Figure 193.

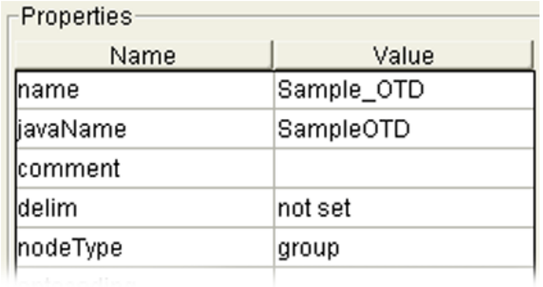
Figure 193 Sample OTD Tree



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 194.

Figure 194 Initial Root Node Properties

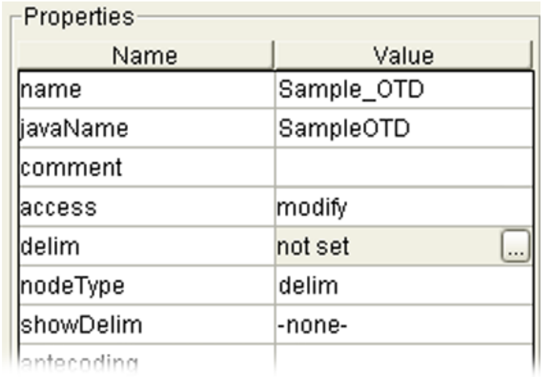

Properties

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 195.

Figure 195 Activated delim Value Field

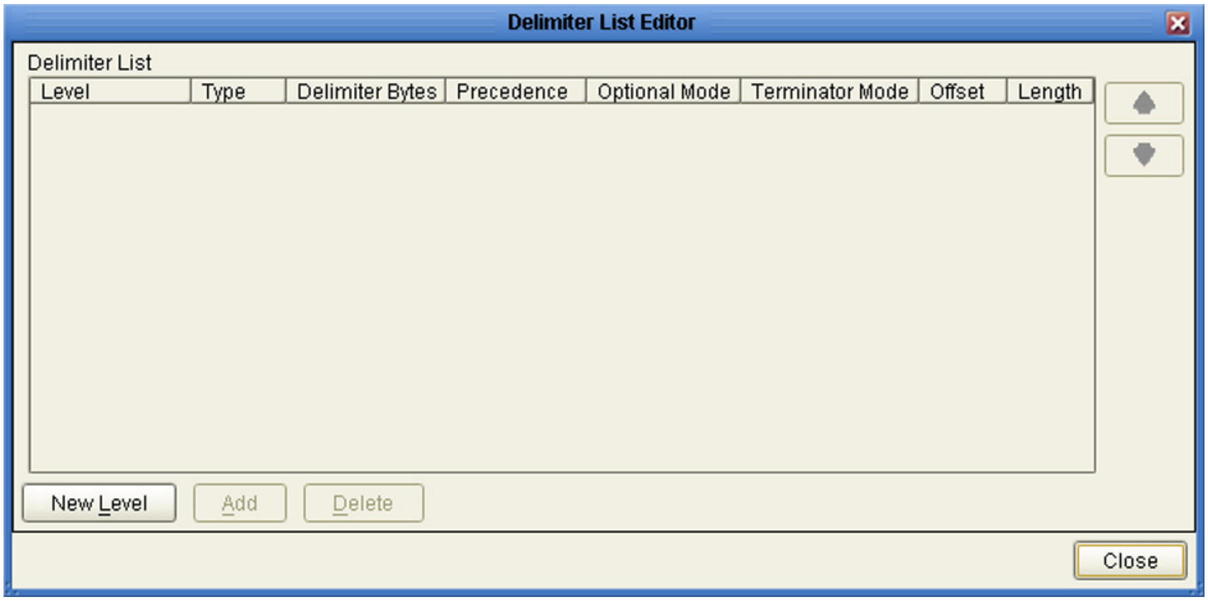


Properties

Name	Value
name	Sample_OTD
javaName	SampleOTD
comment	
access	modify
delim	not set ...
nodeType	delim
showDelim	-none-
antecoding	

4 Click the ellipsis button to display the Delimiter List Editor, which is initially blank (see Figure 196).

Figure 196 Delimiter List Editor - Delimiters Not Set



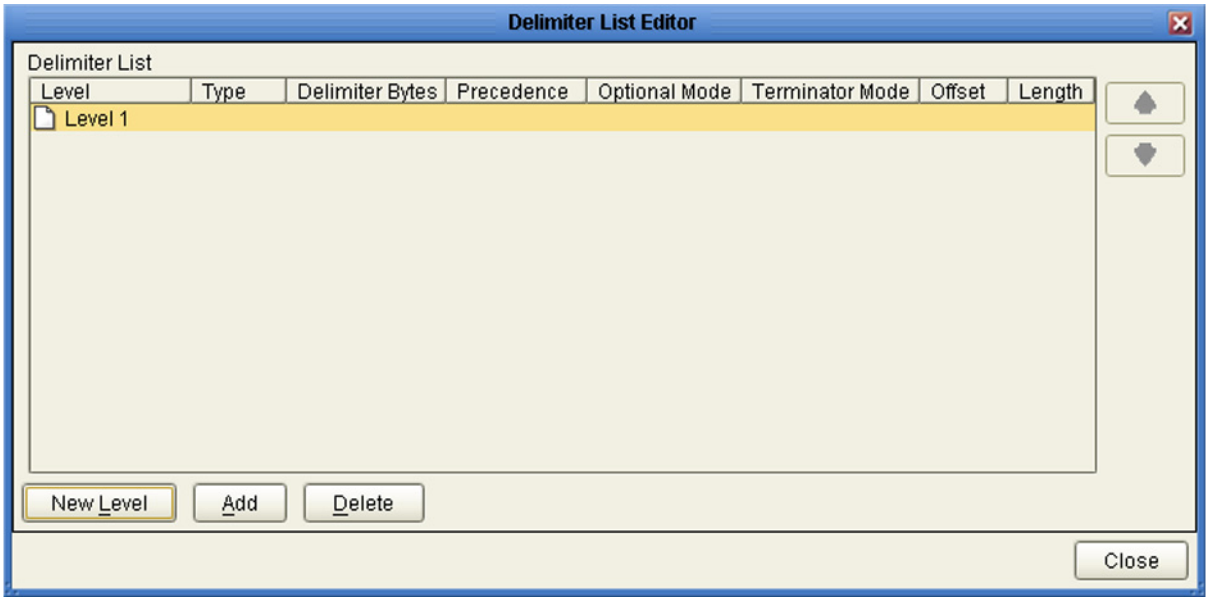
Delimiter List Editor

Delimiter List

Level	Type	Delimiter Bytes	Precedence	Optional Mode	Terminator Mode	Offset	Length

5 Click **New Level** to add a level to the delimiter list, as shown in Figure 197.

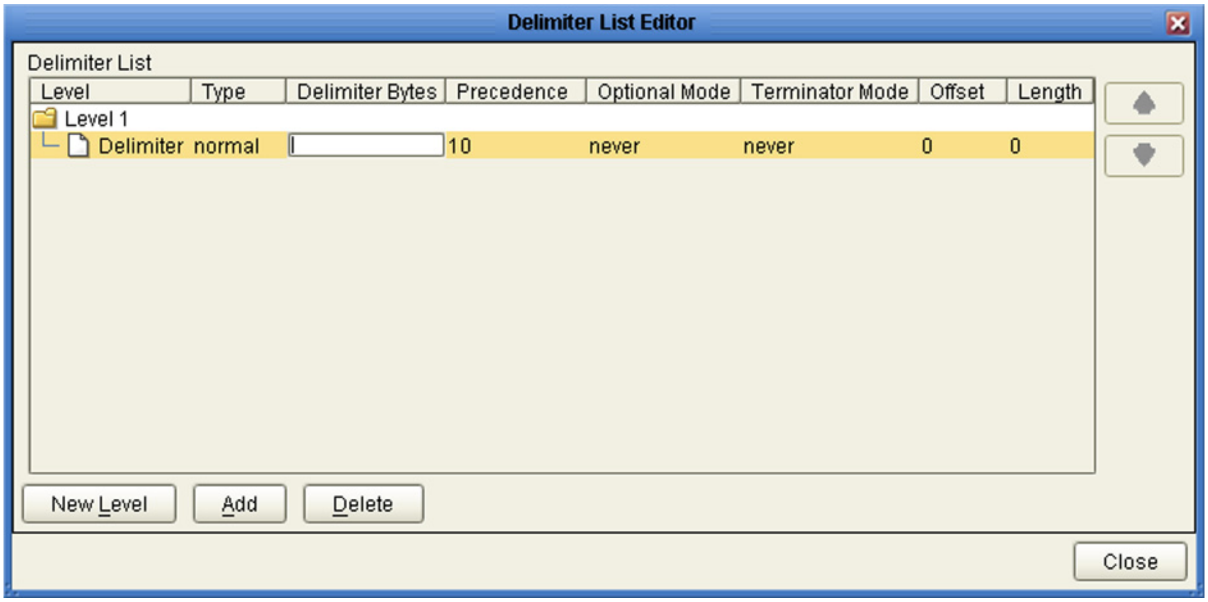
Figure 197 Delimiter List Editor - Insert Level



Level	Type	Delimiter Bytes	Precedence	Optional Mode	Terminator Mode	Offset	Length
Level 1							

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 198).

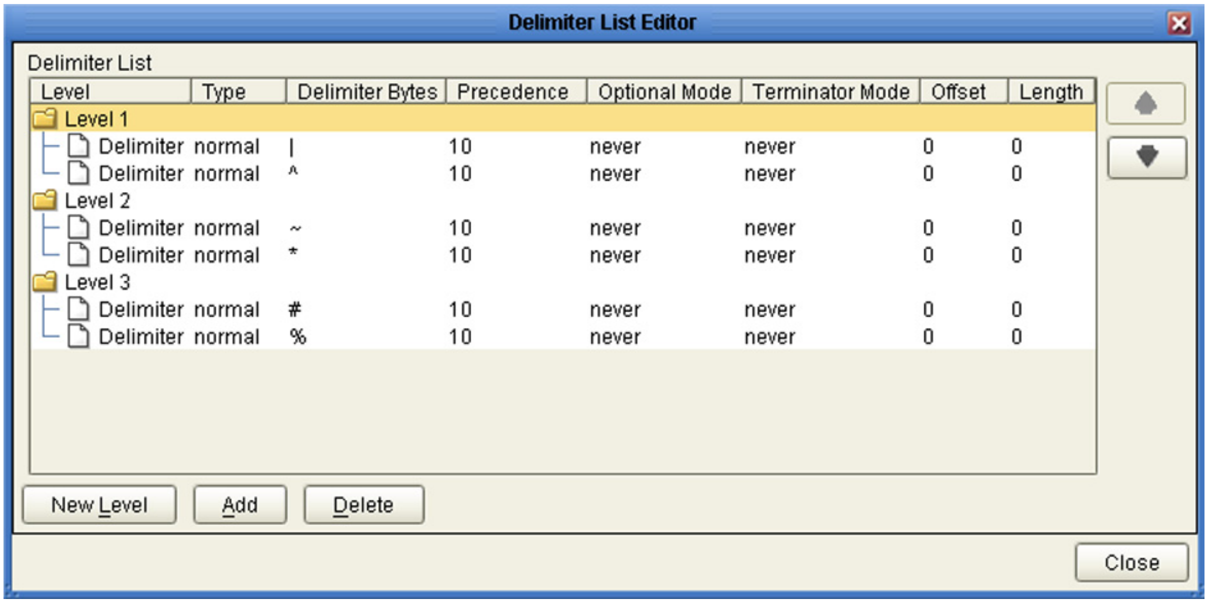
Figure 198 Delimiter List Editor - Add Delimiter



Level	Type	Delimiter Bytes	Precedence	Optional Mode	Terminator Mode	Offset	Length
Level 1							
Delimiter	normal		10	never	never	0	0

7 Continue adding levels and delimiters as required (see Figure 199).

Figure 199 Delimiter List Editor - Add Levels and Delimiters



Delimiter List Editor

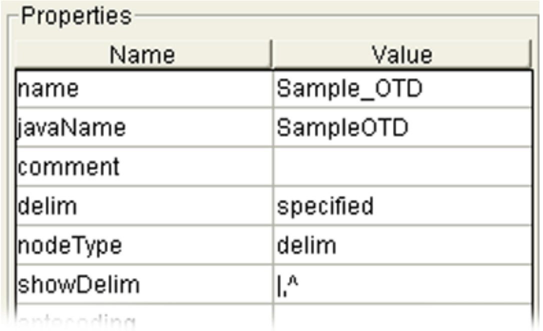
Delimiter List

Level	Type	Delimiter Bytes	Precedence	Optional Mode	Terminator Mode	Offset	Length
Level 1							
Delimiter	normal	|	10	never	never	0	0
Delimiter	normal	^	10	never	never	0	0
Level 2							
Delimiter	normal	~	10	never	never	0	0
Delimiter	normal	*	10	never	never	0	0
Level 3							
Delimiter	normal	#	10	never	never	0	0
Delimiter	normal	%	10	never	never	0	0

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 200.

Figure 200 Root Node - Delimiter Specified



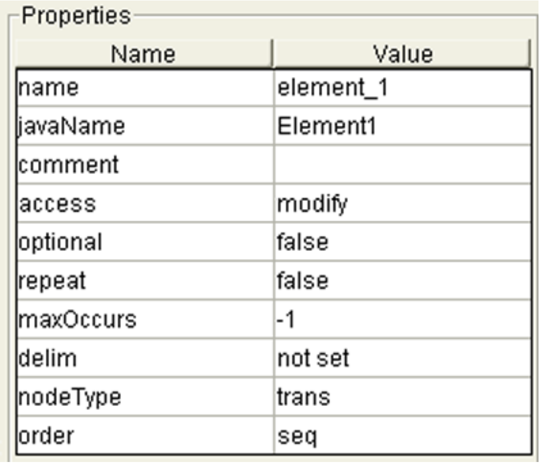
Properties

Name	Value
name	Sample_OTD
javaName	SampleOTD
comment	
delim	specified
nodeType	delim
showDelim	|,^

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 201. 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 201 Element_1 Node Properties

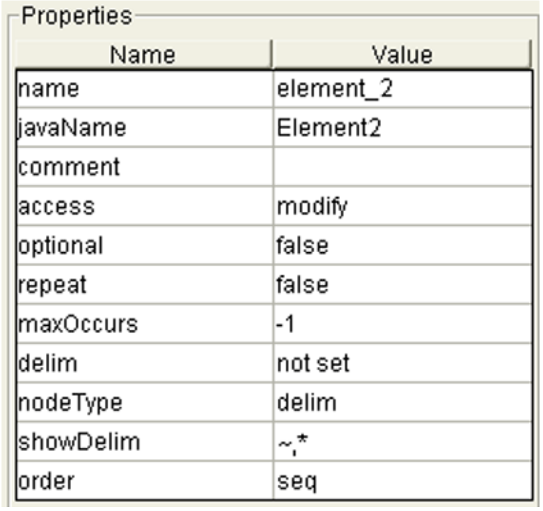


Properties

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 202. 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 202 Element_2 Node Properties

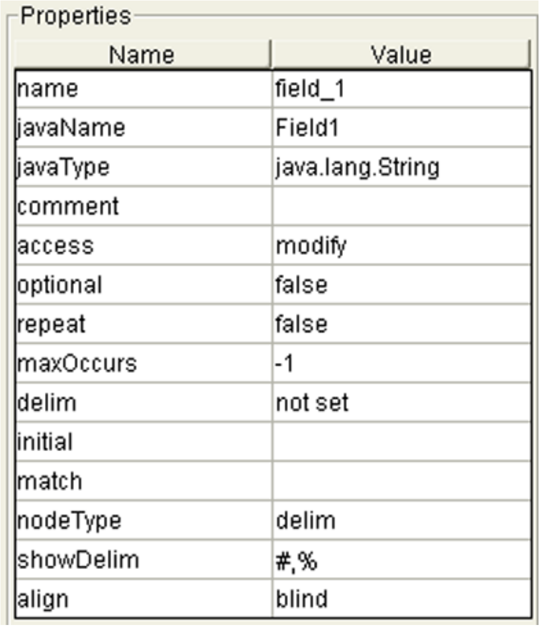


Properties

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 203. Again, the **delim** property remains *not set*.

Figure 203 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 171).

Chapter 9

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 252
- **Using the Collaboration Definition Context Menu** on page 254
- **Using the Collaboration Editor (Java)** on page 256
- **Graphical Collaboration Methods (Java)** on page 273
- **Using the Class Browser** on page 292
- **Using the Method Dialog** on page 297
- **Auto Type Conversion** on page 300
- **Using the Collaboration Tester (Java)** on page 306
- **Using the Java Debugger** on page 313

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 though 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 102).*

As with other eGate Integrator components, it is essential to manage versions of Collaboration Definitions carefully. See **Managing Component Versions** on page 81 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 204 Collaboration Definition Context Menu

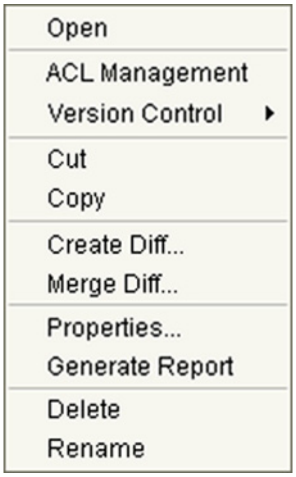


Table 54 Collaboration Definition Context Menu Options

Option	Function
Open	Opens the appropriate Collaboration Editor, showing the selected Collaboration Definition. See **Using the Collaboration Editor (Java)** on page 256.
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 *Sun SeeBeyond eGate™ Integrator System Administration Guide*.
Version Control	Accesses the version control features for the selected component. See **Accessing Component Version Control Features** on page 82 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 within the same branch (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 **Moving and Replicating Repository Objects** on page 100 for additional information.

Table 54 Collaboration Definition Context Menu Options

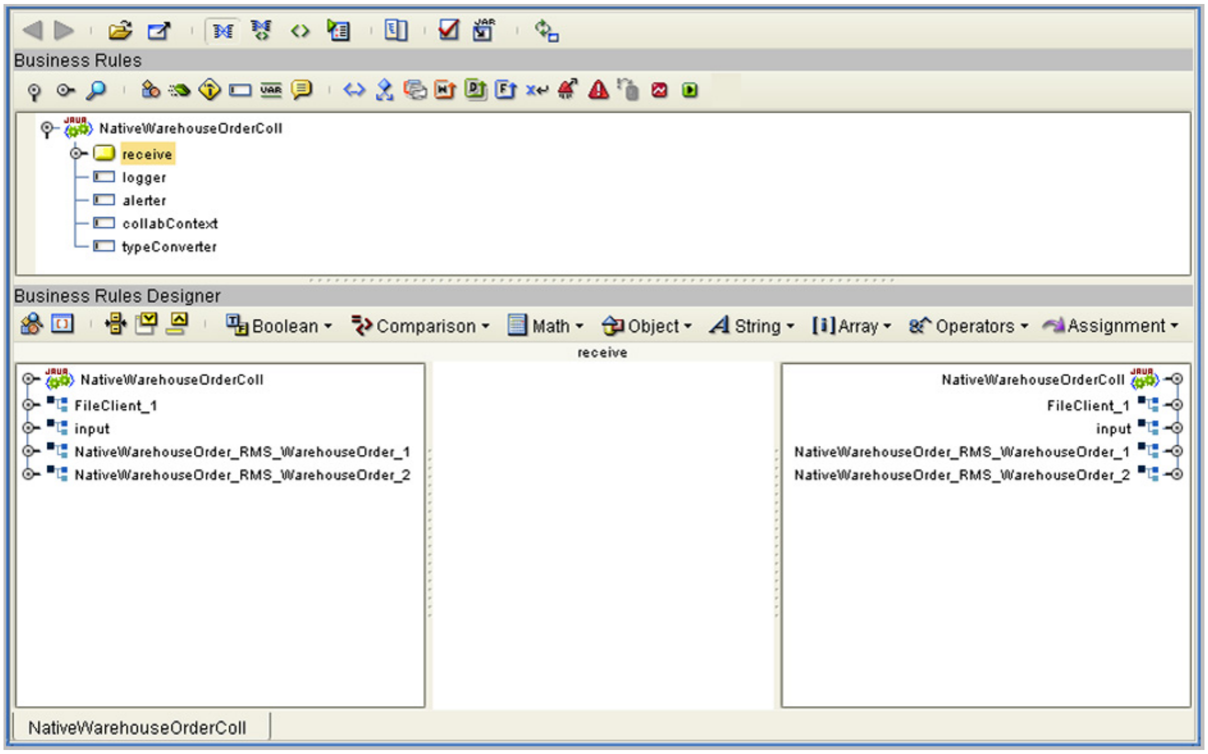
Option	Function
Copy	Copies the selected Collaboration Definition (Java only), after which you can paste it to other Projects within the same branch (multiple times). All changes must be committed before you can copy the Collaboration. You can copy and paste a Collaboration even when another user has the Collaboration checked out. See **Moving and Replicating Repository Objects** on page 100 for additional information.
Create Diff	Presents a dialog with which you can create a difference file representing two versions of the selected Collaboration Definition. See **Merging Changes from Another Version** on page 97.
Merge Diff	Presents a dialog with which you can merge modifications into a specific version of the selected Collaboration Definition. See **Merging Changes from Another Version** on page 97.
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	Deletes the selected Collaboration Definition, subject to the following conditions: ▪ You have *write* privileges for the component (see ACL Management, above). ▪ You have the component checked out to your workspace. ▪ The component is not checked out by anyone other than yourself. If these conditions are met, a dialog is presented in which you confirm that you want to delete the selected component. Clicking **Yes** then deletes the component.
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.

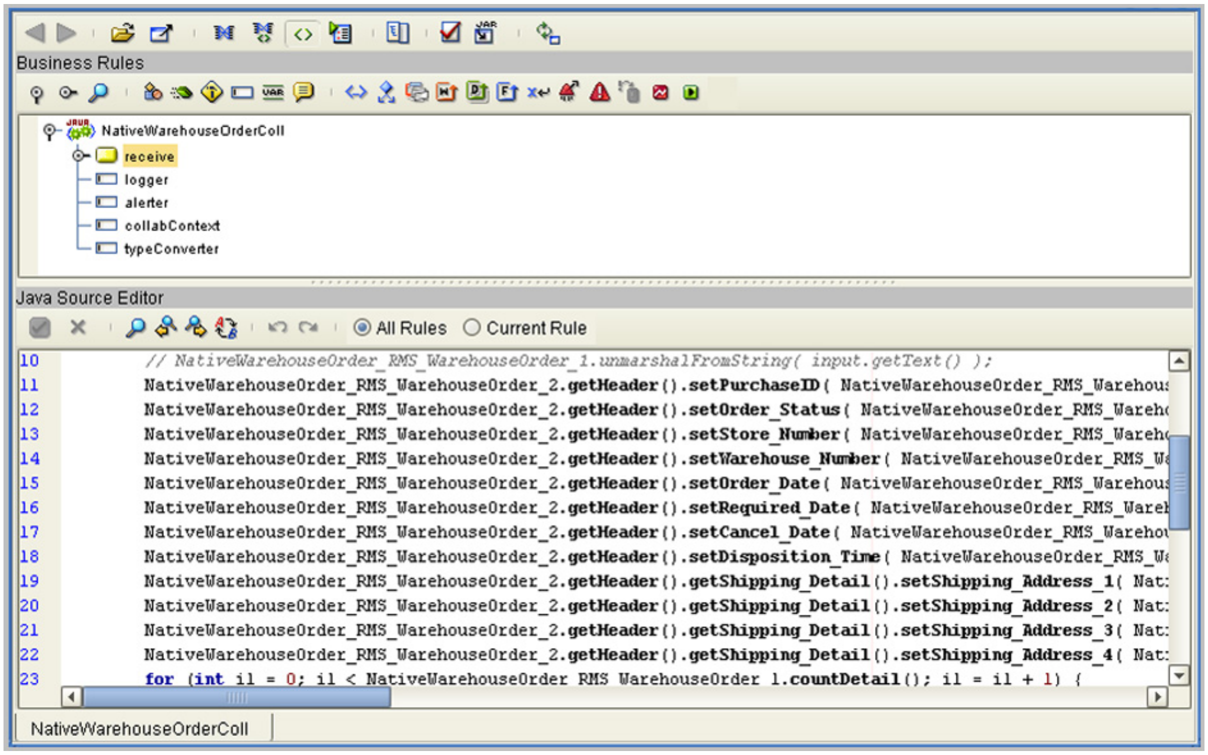
After you create a Java-based Collaboration Definition file using the Collaboration Definition Wizard (Java), the Collaboration Editor (Java) appears in the editor panel of the Enterprise Designer (see Figure 205). By default, the Business Rules Editor and Business Rules Designer panels are displayed.

Figure 205 Collaboration Editor (Java)



Selecting the *Source Code Mode* icon (see **Collaboration Editor Toolbar Icons** on page 258) substitutes the Java Source Editor for the Business Rules Designer, as shown in Figure 206.

Figure 206 Collaboration Editor (Java) - Source Code Mode



```
        // NativeWarehouseOrder_RMS_WarehouseOrder_1.unmarshalFromString( input.getText() );
        NativeWarehouseOrder_RMS_WarehouseOrder_2.getHeader().setPurchaseID( NativeWarehouseOrder_RMS_Warehous
        NativeWarehouseOrder_RMS_WarehouseOrder_2.getHeader().setOrder_Status( NativeWarehouseOrder_RMS_Wareho
        NativeWarehouseOrder_RMS_WarehouseOrder_2.getHeader().setStore_Number( NativeWarehouseOrder_RMS_Wareho
        NativeWarehouseOrder_RMS_WarehouseOrder_2.getHeader().setWarehouse_Number( NativeWarehouseOrder_RMS_Wa
        NativeWarehouseOrder_RMS_WarehouseOrder_2.getHeader().setOrder_Date( NativeWarehouseOrder_RMS_Warehous
        NativeWarehouseOrder_RMS_WarehouseOrder_2.getHeader().setRequired_Date( NativeWarehouseOrder_RMS_Wareh
        NativeWarehouseOrder_RMS_WarehouseOrder_2.getHeader().setCancel_Date( NativeWarehouseOrder_RMS_Warehou
        NativeWarehouseOrder_RMS_WarehouseOrder_2.getHeader().setDisposition_Time( NativeWarehouseOrder_RMS_Wa
        NativeWarehouseOrder_RMS_WarehouseOrder_2.getHeader().getShipping_Detail().setShipping_Address_1( Nat:
        NativeWarehouseOrder_RMS_WarehouseOrder_2.getHeader().getShipping_Detail().setShipping_Address_2( Nat:
        NativeWarehouseOrder_RMS_WarehouseOrder_2.getHeader().getShipping_Detail().setShipping_Address_3( Nat:
        NativeWarehouseOrder_RMS_WarehouseOrder_2.getHeader().getShipping_Detail().setShipping_Address_4( Nat:
        for (int i1 = 0; i1 < NativeWarehouseOrder_RMS_WarehouseOrder_1.countDetail(); i1 = i1 + 1) {
```

Selecting the *Advanced Mode* icon (see **Collaboration Editor Toolbar Icons** on page 258) 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 261.
- **Business Rules Designer** on page 266.
- **Java Source Editor** on page 264.

9.3.1 Collaboration Editor Toolbar Icons

Table 55 Collaboration Editor Toolbar Icons

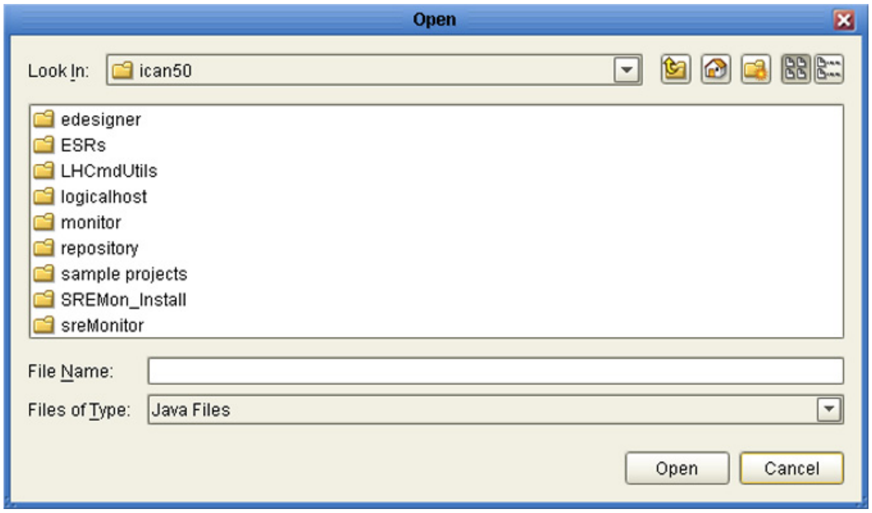
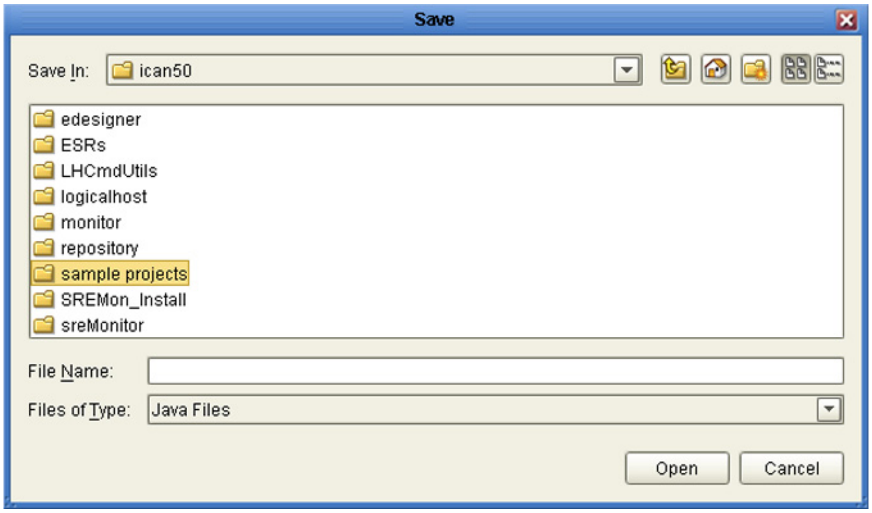
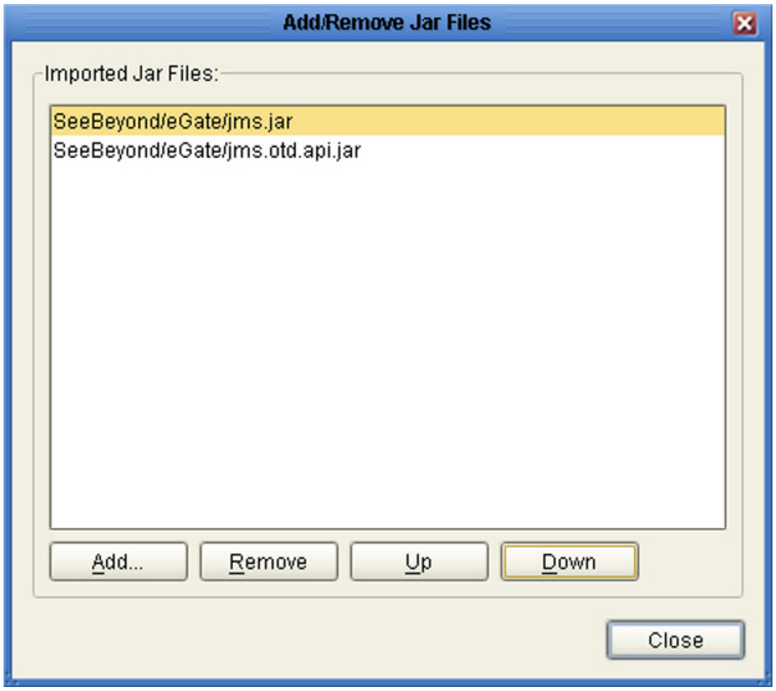
Icon	Command	Function
◁	Back	Steps back in your usage history for the current session.
▷	Forward	Steps forward in your usage history for the current session.
📂	Import File	Presents a dialog with which you can locate and select a Collaboration Definition (Java) to import. When you import a file, the imported code replaces any existing code.  Look In: ican50 edesigner ESRs LHCmdUtils logicalhost monitor repository sample projects SREMon_Install sreMonitor File Name: Files of Type: Java Files
↗	Export File	Presents a dialog with which you can save the selected Collaboration Definition (Java) to a file.  Save In: ican50 edesigner ESRs LHCmdUtils logicalhost monitor repository sample projects SREMon_Install sreMonitor File Name: Files of Type: Java Files

Table 55 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 **Using the Collaboration Tester (Java)** on page 306).
	Business Rules on Left	Changes the editor layout to display the Business Rules panel to the left of the Business Rules Designer. Toggles with *Business Rules on Top*.
	Business Rules on Top	Changes the editor layout to display the Business Rules panel above the Business Rules Designer (default setting). Toggles with *Business Rules on Left*.
✓	Validate	Verifies that changes made to the Java code are valid.

Table 55 Collaboration Editor Toolbar Icons

<table>
<tr><th>Icon</th><th>Command</th><th>Function</th></tr>
<tr><td>JAR</td><td>Import JAR File</td><td>Presents a dialog with which you can import a <b>.jar</b> file (see <b>Importing Third-Party Java Classes</b> on page 362).



Add/Remove Jar Files

Imported Jar Files:

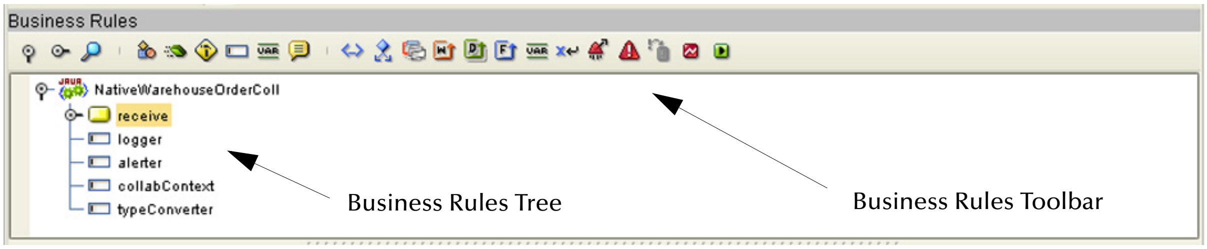
SeeBeyond/eGate/jms.jar
SeeBeyond/eGate/jms.otd.api.jar

</td></tr>
<tr><td>⟳</td><td>Refresh Collaboration</td><td>Refreshes the Collaborations from the Repository.</td></tr>
</table>

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 207). 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 207 Business Rules Editor



Business Rules Editor Toolbar Icons

Table 56 Business Rules Editor Toolbar Buttons

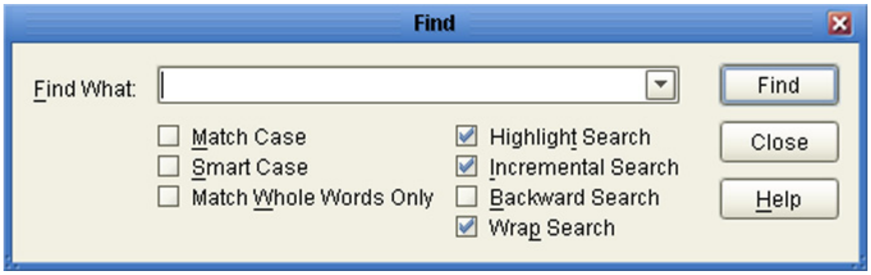
Icon	Command	Function
	Expand All Rules	Expands the tree to show all rules.
	Collapse All Rules	Collapses all rules in the tree.
	Find	Presents a dialog in which you can enter text to search for in the Java Source Editor.  Find Find What: Find Close Help ☐ Match Case ☐ Smart Case ☐ Match Whole Words Only ☑ Highlight Search ☑ Incremental Search ☐ Backward Search ☑ Wrap Search
	Class	Presents the *Create Class* dialog (see **Creating Classes** on page 337).
	Method	Presents a dialog with which you can add a new method to the Business Rules tree (see **Creating Methods** on page 339).

Table 56 Business Rules Editor Toolbar Buttons

Icon	Command	Function
	Constructor	Presents the *Create Constructor* dialog (see Creating Constructors on page 343).
	Field	Presents a dialog with which you can add a new field to the Business Rules tree (see Creating Fields on page 346).
VAR	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) **rule** statement to the Business Rules tree. Use this command as a safeguard regarding positioning.
	If-Then	Adds an **if-then** statement to the Business Rules tree.
	Switch	Adds a **switch** statement to the Business Rules tree, which provides an alternative to the **if-then** statement.
W	While	Adds a **while** 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 **while** statement is selected.
D	Do-While	Adds a **do-while** 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 **do-while** statement is selected.
F	For	Adds a **for** statement to the Business Rules tree, starting a specific iteration (repetition) of business rules.
	Return	Adds a **return** statement to the Business Rules tree.
	Throw	Adds a **throw** statement for throwing exceptions.
	Try	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 (see Using Try-Catch on page 368). Automatically enables the **Catch** icon.

Table 56 Business Rules Editor Toolbar Buttons

Icon	Command	Function
	Catch	Adds a **catch** statement to an existing **try** statement (disabled in the absence of a **try** statement).
	Break	Breaks out of a business rule loop, for example when using a **while**, **for**, or **switch** command.
▶	Continue	Continues the execution of business rules in a **for** or **while** 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.

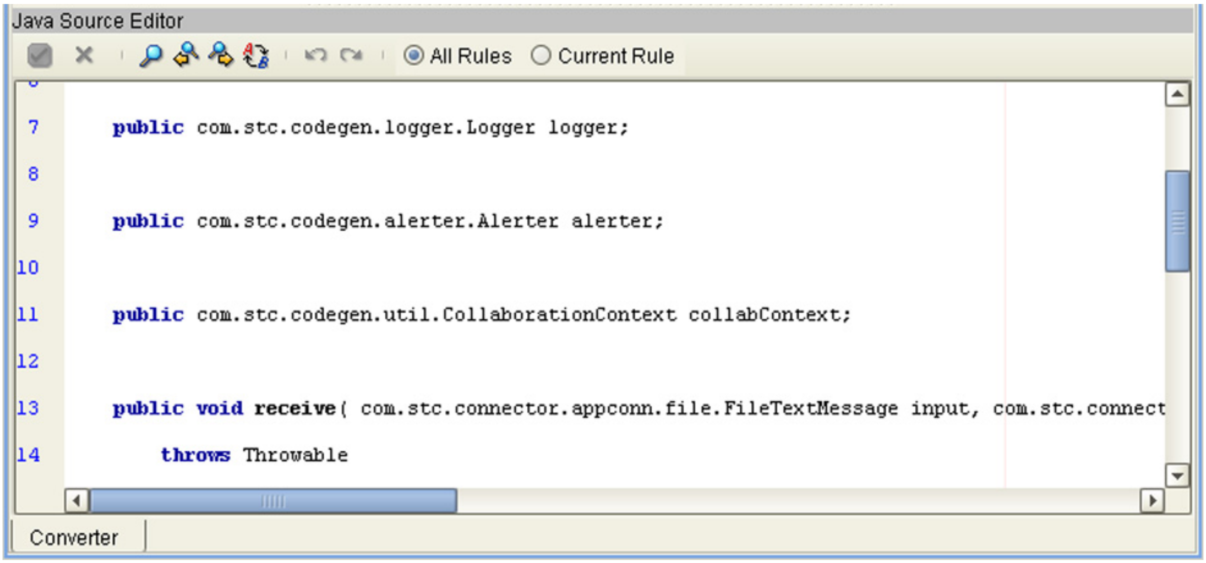
Note: *When adding a method as a sibling to another method, make sure you add it after the last rule in the preceding code block. Otherwise, it will be incorporated into the code block as a child of the preceding method.*

9.3.3 Java Source Editor

The Java Source Editor (see Figure 208) 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 208 Java Source Editor



```
    public com.stc.codegen.logger.Logger logger;

    public com.stc.codegen.alerter.Alerter alerter;

    public com.stc.codegen.util.CollaborationContext collabContext;

    public void receive( com.stc.connector.appconn.file.FileTextMessage input, com.stc.connect

        throws Throwable
```

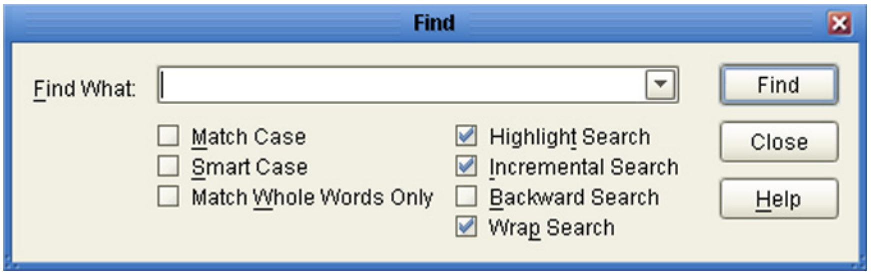
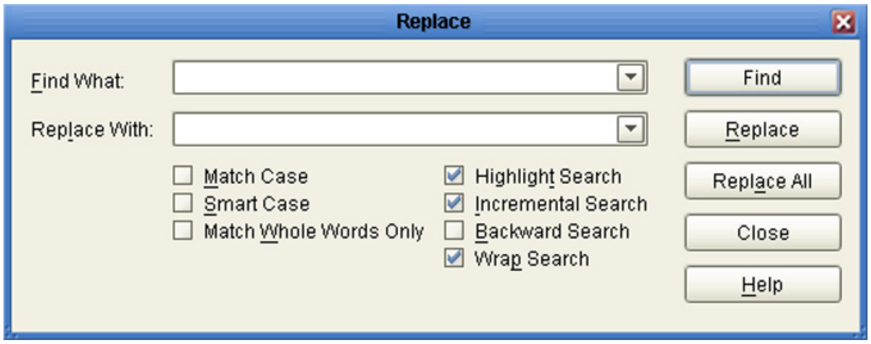
Code completion in the Java Source Editor shows standard Java library classes, classes from OTD **.jar** files, and classes from any imported third-party **.jar** files.

Java Source Editor Toolbar Icons

Table 57 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.

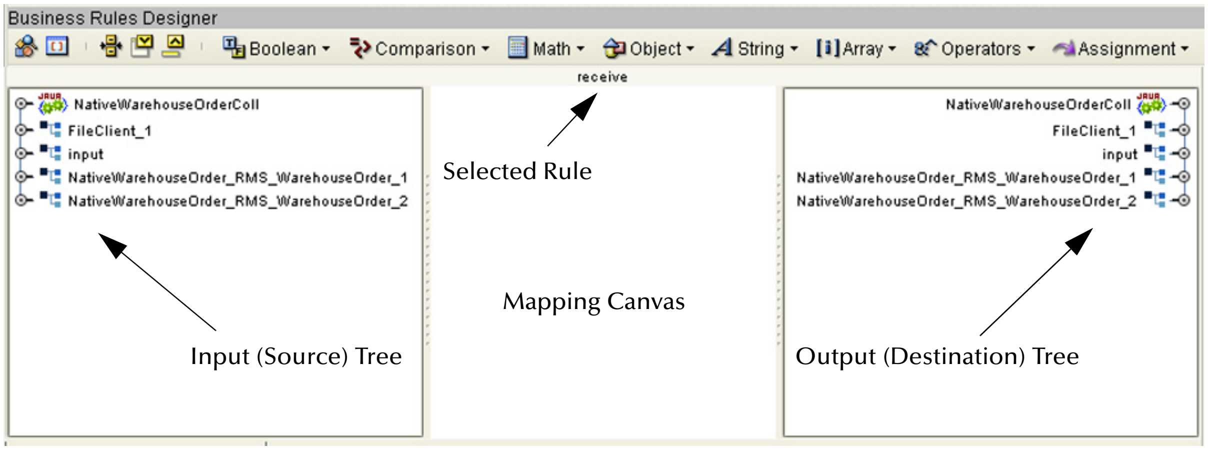
Table 57 Java Source Editor Toolbar Icons

Icon	Command	Function
🔍	Find	Opens the **Find** dialog, where you can enter text to search for in the Java Source Editor.  Find Find What: ☐ Match Case ☐ Smart Case ☐ Match Whole Words Only ☑ Highlight Search ☑ Incremental Search ☐ Backward Search ☑ Wrap Search
🔍⇦	Find Previous	Searches backward for a previous instance of the text entered in the **Find** dialog.
🔍⇨	Find Next	Searches forward for the next instance of the text entered in the **Find** dialog.
AB	Replace	Opens the **Replace** dialog, where you can enter text to search for, and to replace with, in the Java Source Editor.  Replace Find What: Replace With: ☐ Match Case ☐ Smart Case ☐ Match Whole Words Only ☑ Highlight Search ☑ Incremental Search ☐ Backward Search ☑ Wrap Search
↶	Undo	Undoes the last operation.
↷	Redo	Restores the last operation, if undone.

9.3.4 Business Rules Designer

The Business Rules Designer (see Figure 209) 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 267). These icons also appear in the context menu for each rule in the Business Rules Editor.

Figure 209 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 58 Business Rules Designer Toolbar Icons

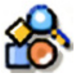
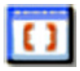
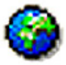
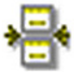
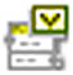
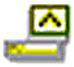
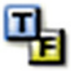
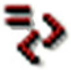
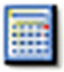
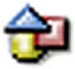
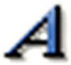
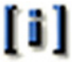
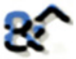
Icon	Command	Function
	Class Browser	Invokes the Class Browser (see **Using the Class Browser** on page 292).
	New Array	Presents a dialog with which you can add an array to the Collaboration Definition (see **Creating Arrays** on page 348).
	Encoding Converter	Presents the *Encoding Converter Methods* dialog (see **Defining Encoding Converter Methods** on page 372). Icon is shown only if you enable the extended language options (see **Language** on page 56).
	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 *Boolean* methods menu.
	Comparison	Displays the *Comparison* methods menu.
	Math	Displays the *Math* methods menu.
	Object	Displays the *Object* methods menu.
	String	Displays the *String* methods menu.
	Array	Displays the *Array* methods menu.
	Operator	Displays the *Operator* methods menu.

Table 58 Business Rules Designer Toolbar Icons

Icon	Command	Function
↷	Assignment	Displays the *Assignment* methods menu.

Node Menu

Right-clicking a node in the source or destination tree displays the menu shown in Figure 210, which duplicates several toolbar commands for the selected node.

Figure 210 Node Menu

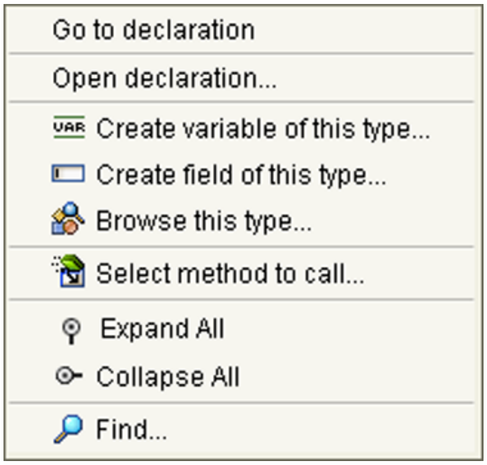


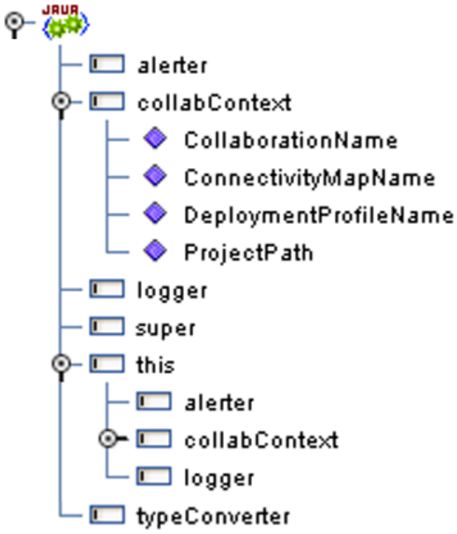
Table 59 Business Rules Designer Node Menu Options

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 *Create Variable* dialog (see **Creating and Modifying Variables** on page 350).
Create field of this type	Presents the *Create Field* dialog (see **Creating Fields** on page 346).
Browse this type	Presents the *Class Browser* dialog (see **Using the Class Browser** on page 292).
Select method to call	Presents the method dialog, listing methods appropriate to the selected node (see **Using the Method Dialog** on page 297).
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 211).

Figure 211 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 60 Method Access Nodes

Node	Subnode	Description
alerter		Allows access to **alert** methods, as described in **Generating Alerts** on page 355.
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 **logging** methods, as described in **Creating Log Entries** on page 359.
super		Allows you to assign the class to another Java class (superclass).
this		Allows you to call other methods defined for *this* Collaboration without having to edit the Java code. Subnodes duplicate the nodes described above.

Table 60 Method Access Nodes

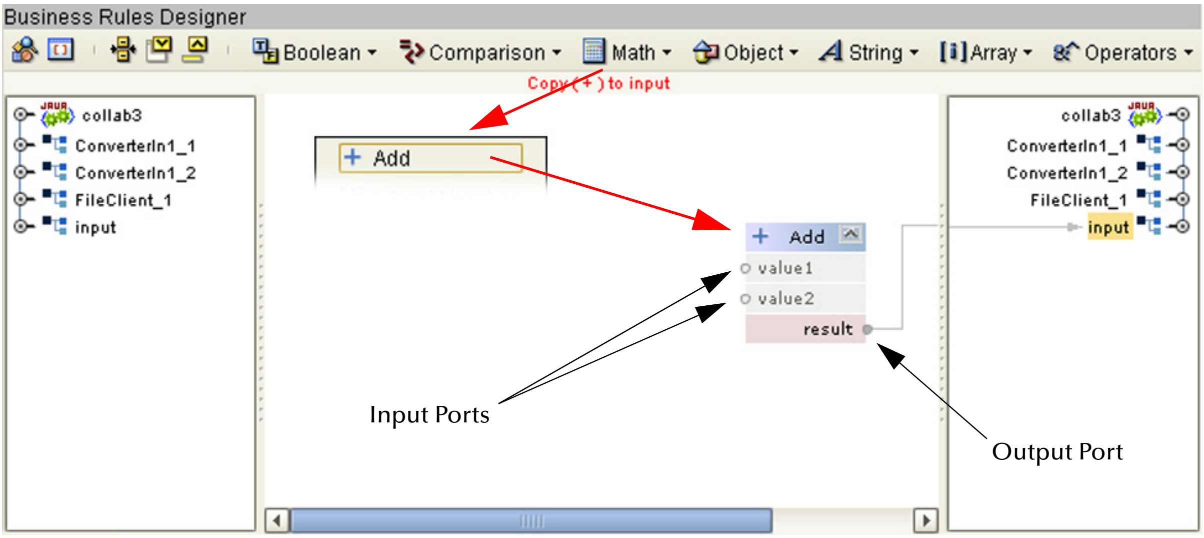
Node	Subnode	Description
typeConverter		Allows access to **datatype conversion** methods, as described in **Auto Type Conversion** on page 300.

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 273). 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 272).

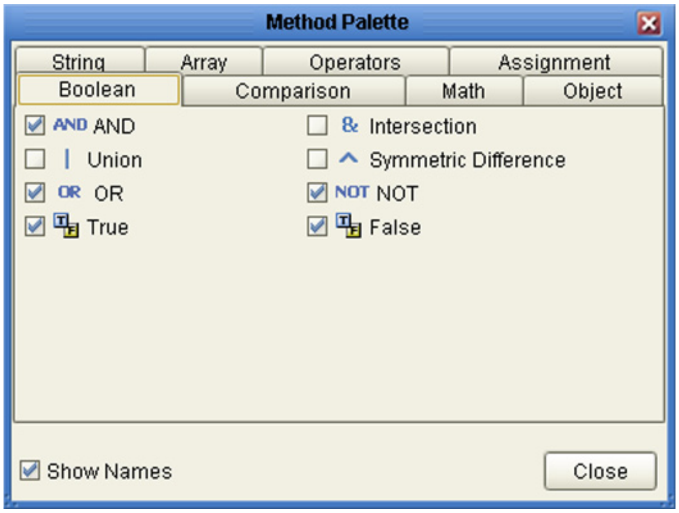
As an example, clicking **Math** in the toolbar displays a menu listing mathematical methods. Clicking **Add** in the menu places an **Add** method box on the mapping canvas. The method box contains input and output ports which you connect to the appropriate nodes in the OTD tree structures by dragging the cursor.

Figure 212 Business Rules Designer: Method Menus and Boxes



Clicking the **Settings** option in any menu **displays the Method Palette dialog shown in** Figure 213**, 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 213 Collaboration Method Palette (Java)



☑ AND AND
□ | Union
☑ OR OR
☑ True
□ & Intersection
□ ^ Symmetric Difference
☑ NOT NOT
☑ False

☑ Show Names

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 212 on page 270, 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 214); you can collapse them (see Figure 215) 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 214 Expanded Method Box

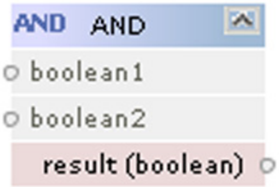
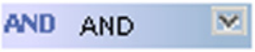


Figure 215 Collapsed Method Box



Note: *Tooltips, derived from the associated Javadocs, appear when you hover over a field in a method box.*

Placing Collaboration Methods

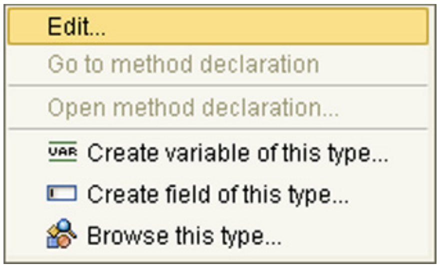
Collaboration Method Boxes are placed on the mapping canvas in any of three ways — pick the one that best suits the occasion and your operating style:

- 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 268).
- 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 297).
- Click a method category in the toolbar to display the corresponding menu, and click the desired option (see **Collaboration Method Menus** on page 270). 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 216 Method Box Context Menu - Literal


Edit...
Go to method declaration
Open method declaration...
Create variable of this type...
Create field of this type...
Browse this type...

9.4 Graphical Collaboration Methods (Java)

9.4.1 Array Methods

Figure 217 Array Methods Menu

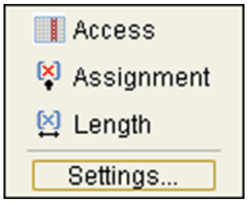


Table 61 Array Operation Collaboration Methods (Java)

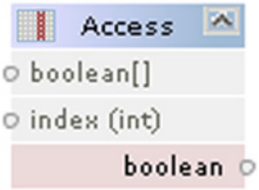
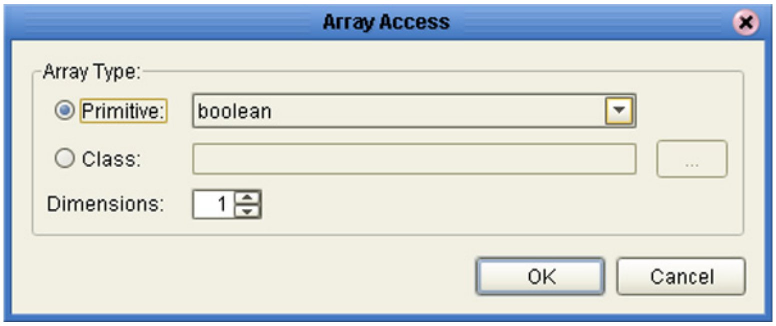
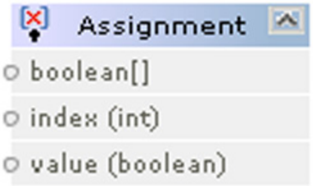
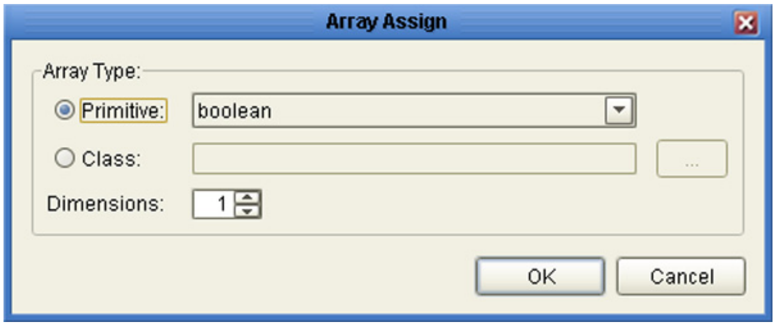
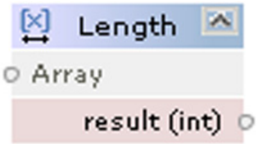
Method Box	Description/Usage
	Selecting **Access** first presents the *Array Access* dialog, in which you specify the desired array type (can be either *primitive* or *class*) and dimensions. Clicking **OK** places the **Access** method box on the mapping canvas. At run time, the **Access** method returns the element located at the specified index within the array, where the index represents the number of elements from the beginning of the array.  Array Access Array Type: ● Primitive: boolean ○ Class: Dimensions: 1 OK Cancel

Table 61 Array Operation Collaboration Methods (Java)

Method Box	Description/Usage
 	Selecting **Assignment** first presents the *Array Assign* dialog, in which you specify the desired array type (can be either *primitive* or *class*) and dimension. Clicking **OK** places the **Assignment** method box on the mapping canvas.At run time, the **Assignment** 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.  Array Assign Array Type: ● Primitive: boolean ○ Class: Dimensions: 1
 	At run time, the **Length** method returns the length (number of elements) of the array.

9.4.2 Assignment

The Assignment methods perform the indicated operation and then assign the result back into the variable attribute.

Figure 218 Assignment Menu

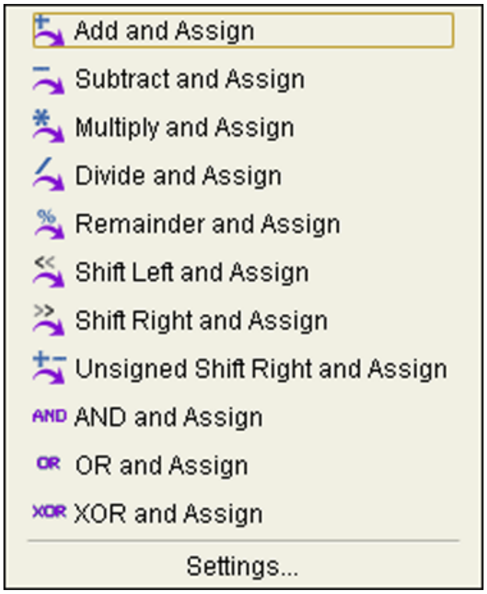


Table 62 Assignment (Java)

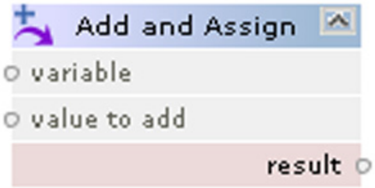
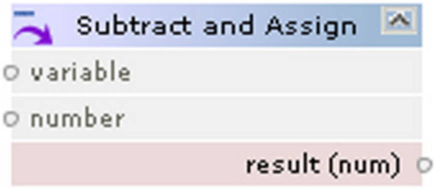
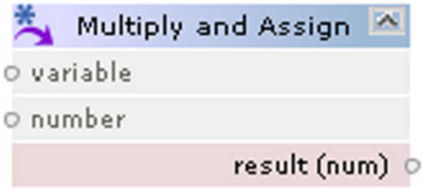
Method Box	Description/Usage
	At run time, the **Add and Assign** method adds a value to a numerical or string variable, and then assigns the sum back into the variable.
	At run time, the **Subtract and Assign** method subtracts the numerical value of *number* from the numerical value of *variable*, and then assigns the difference back into *variable*.
	At run time, the **Multiply and Assign** method multiplies the numerical value of *number* and the numerical value of *variable*, and then assigns the product back into *variable*.

Table 62 Assignment (Java)

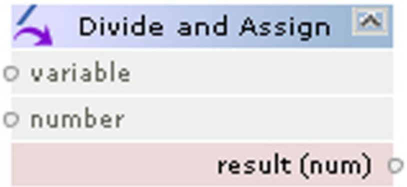
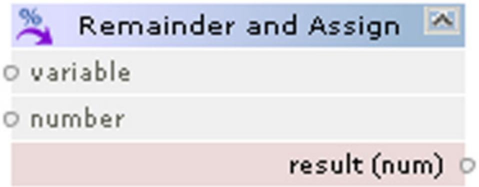
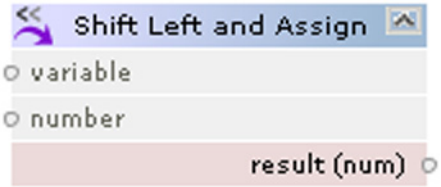
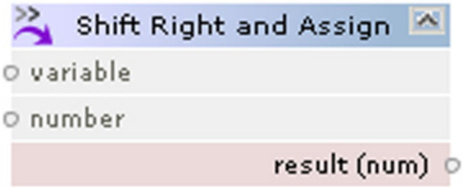
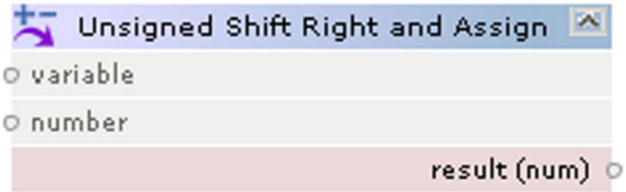
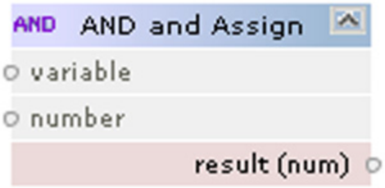
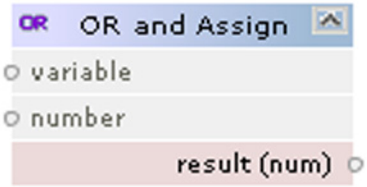
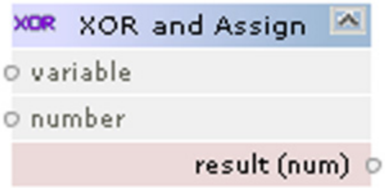
Method Box	Description/Usage
	At run time, the **Divide and Assign** method divides the numerical value of *variable* by the numerical value of *number*, and then assigns the quotient back into *variable*.
 % Remainder and Assign variable number result (num)	At run time, the **Remainder and Assign** method divides the numerical value of *variable* by the numerical value of *number*, and then assigns the remainder back into *variable*.
 << Shift Left and Assign variable number result (num)	At run time, the **Shift Left and Assign** method shifts the value of *variable* to the left by *number* of places, and then assigns the result back into *variable*. (This operation is equivalent to multiplication by power of two.)
 >> Shift Right and Assign variable number result (num)	At run time, the **Shift Right and Assign** method shifts the value of *variable* to the right by *number* of places, and then assigns the result back into *variable*. New left bits are filled with the value of the high-order bit so that the algebraic sign is retained.
 Unsigned Shift Right and Assign variable number result (num)	At run time, the **Unsigned Shift Right and Assign** method shifts the value of *variable* to the right by *number* of places, and then assigns the result back into *variable*. 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.)
	At run time, the **AND and Assign** method performs a bitwise AND between *variable* and *number*, and then assigns the result back into *variable*. This operation returns Boolean true if both operands evaluate to true, and false if both operands evaluate to false.

Table 62 Assignment (Java)

Method Box	Description/Usage
	At run time, the **OR and Assign** method performs a bitwise OR between *variable* and *number*, and then assigns the result back into *variable*. This operation returns Boolean true if either or both operands evaluate to true; otherwise, returns Boolean false.
	At run time, the **XOR and Assign** method performs a bitwise XOR between *variable* and *number*, and then assigns the result back into *variable*. This operation returns Boolean true if both operands are different; otherwise, returns Boolean false.

9.4.3 Boolean Methods

Figure 219 Boolean Methods Menu

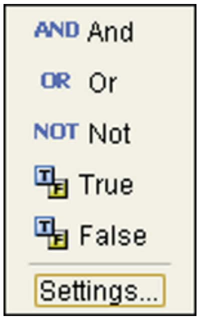
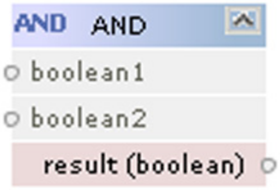
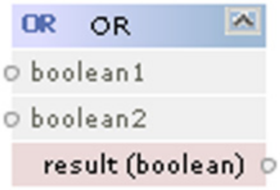
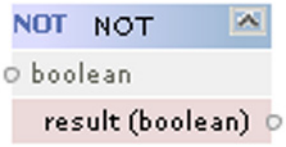
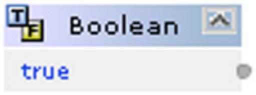
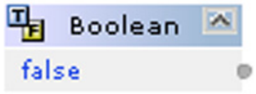
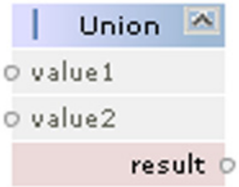
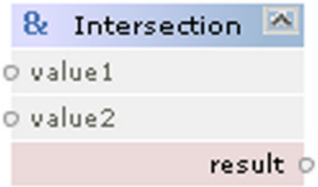
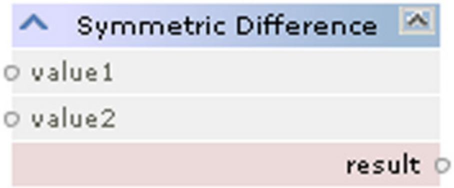


Table 63 Boolean Collaboration Methods (Java)

Method Box	Description/Usage
	At run time, the **AND** method returns Boolean true if both *boolean1* and *boolean2* are true; otherwise, returns Boolean false.
	At run time, the **OR** method returns Boolean false if both *boolean1* and *boolean2* are false; otherwise, returns Boolean true.
	At run time, the **NOT** method returns the inverse of *boolean*.
	At run time, the **True** method always returns Boolean true.
	At run time, the **False** method always returns Boolean false.

The following methods are available from both the *Boolean* menu and the *Operators* menu.

Table 64 Additional Boolean Collaboration Methods (Java)

Method Box	Description/Usage
	At run time, the **Union** operator computes the union of the input values (not shown by default).
	At run time, the **Intersection** operator computes the set intersection of bit or Boolean values (not shown by default).
	At run time, the **Symmetric Difference** operator computes the set symmetric difference of bit or Boolean values (not shown by default).

9.4.4 Comparison Methods

Figure 220 Comparison Methods Menu

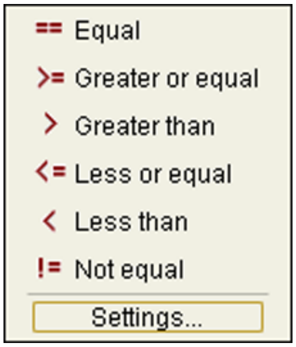


Table 65 Comparison Collaboration Methods (Java)

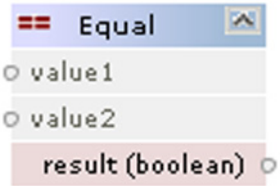
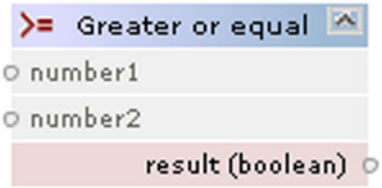
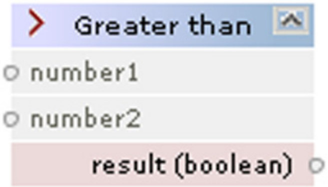
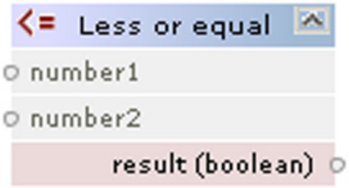
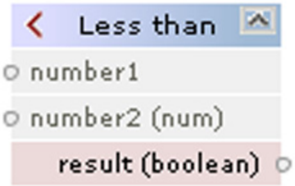
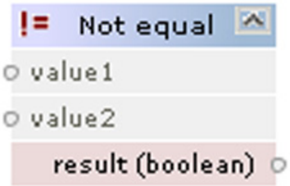
Method Box	Description/Usage
	At run time, the **Equal** method returns Boolean true if the value of *number1* is equal to the value of *number2*; otherwise, returns Boolean false.
 >= Greater or equal number1 number2 result (boolean)	At run time, the **Greater or equal** method returns Boolean true if the value of *number1* is greater than or equal to the value of *number2*; otherwise, returns Boolean false.
	At run time, the **Greater than** method returns Boolean true if the value of *number1* is greater than the value of *number2*; otherwise, returns Boolean false.
	At run time, the **Less or equal** method returns Boolean true if the value of *number1* is less than or equal to the value of *number2*; otherwise, returns Boolean false.

Table 65 Comparison Collaboration Methods (Java)

Method Box	Description/Usage
	At run time, the **Less than** method returns Boolean true if the value of *number1* is less than the value of *number2*; otherwise, returns Boolean false.
	At run time, the **Not equal** method returns Boolean true if the value of *number1* is not equal to the value of *number2*; otherwise, returns Boolean false.

9.4.5 Math Methods

Figure 221 Math Methods Menu

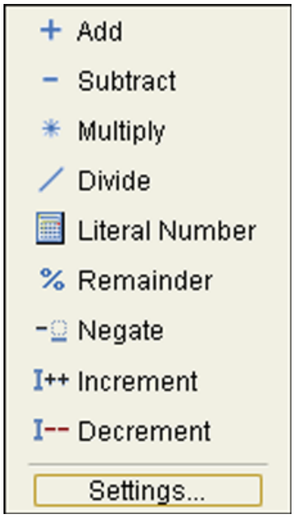


Table 66 Math Collaboration Methods (Java)

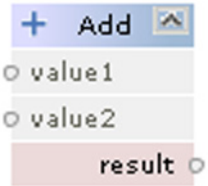
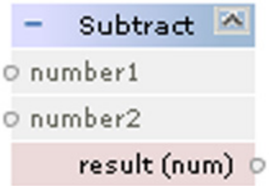
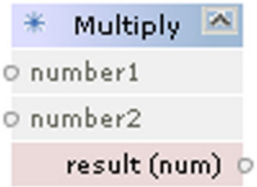
Method Box	Description/Usage
	At run time, the **Add** method returns the sum of *value1* and *value2*.
	At run time, the **Subtract** method subtracts the numerical value of *number2* from the numerical value of *number1*, returns the difference.
	At run time, the **Multiply** method returns the product of the numerical value of *number1* and the numerical value of *number2*.

Table 66 Math Collaboration Methods (Java)

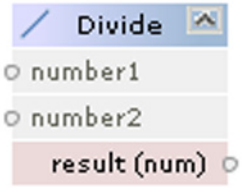
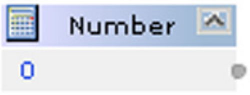
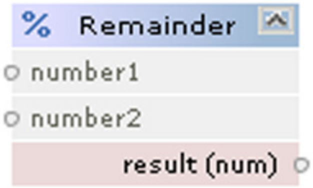
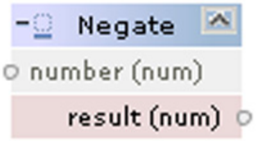
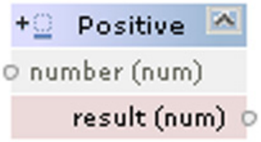
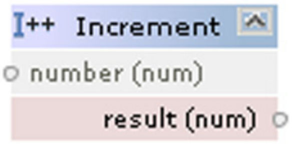
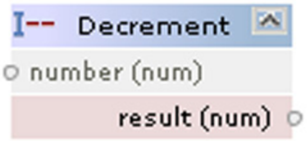
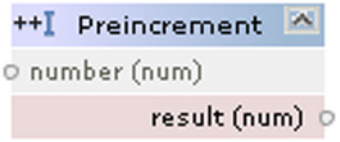
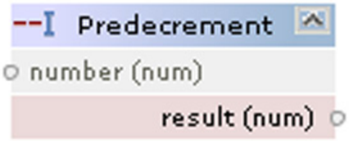
Method Box	Description/Usage
	At run time, the **Divide** method returns the quotient of the numerical value of *number1* divided by the numerical value of *number2*.
	At run time, the literal **Number** method returns the specified numeric value. Double-clicking in the value field enables the field for editing. The default value is zero (**0**).
	At run time, the **Remainder** method divides the numerical value of *number1* by the numerical value of *number2*, returns the remainder.
	At run time, the **Negate** method returns the input number as a negative value.
	The **Positive** 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 **Increment** method increments the input number by one, following other operations.
	At run time, the **Decrement** method decrements the input number by one, following other operations.
	At run time, the **Preincrement** 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 66 Math Collaboration Methods (Java)

Method Box	Description/Usage
	At run time, the **Predecrement** 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.

9.4.6 Object Methods

Figure 222 Object Methods Menu

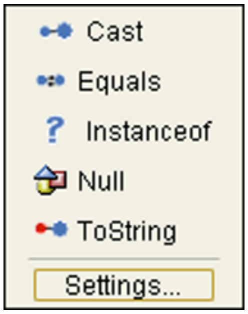


Table 67 Object Collaboration Methods (Java)

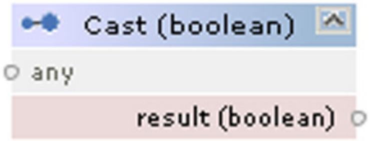
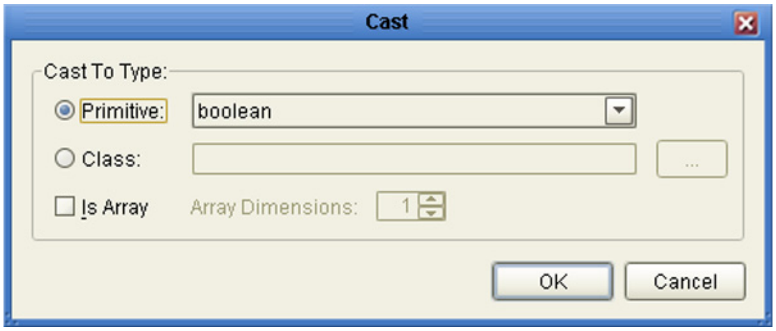
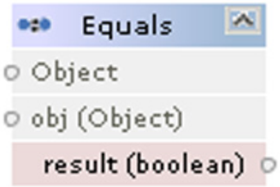
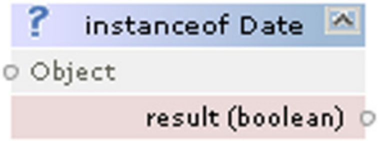
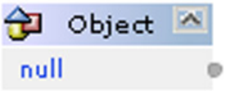
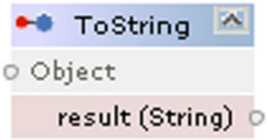
Method Box	Description/Usage
	Selecting **Cast** first presents the *Cast* dialog, in which you specify the desired type (can be either *primitive* or *class*) and dimension (if it is an array). Clicking **OK** places the **Cast** method box on the mapping canvas. At run time, the **Cast** method converts the given type (*any*) to the type specified in the dialog (**boolean** is shown as an example).  Cast Cast To Type: ● Primitive: boolean ○ Class: ☐ Is Array Array Dimensions: 1 OK Cancel
	At run time, the **Equals** method returns Boolean true if *Object* is logically equal to *obj*, otherwise, returns Boolean false.
	Selecting **Instanceof** first presents the *Class Browser* dialog, in which you specify the desired class (see Using the Class Browser on page 292). Clicking **OK** places the **instanceof** method box on the mapping canvas. At run time, the **Instanceof** method returns Boolean true if *Object* is of the specified type; otherwise, returns Boolean false. **Date** is shown as an example.

Table 67 Object Collaboration Methods (Java)

Method Box	Description/Usage
	At run time, the **Null** method always returns a null value.
	At run time, the **ToString** method converts *Object* into a string.

9.4.7 Operators

Figure 223 Operators Menu

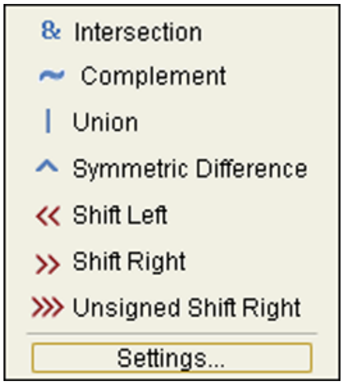


Table 68 Operators (Java)

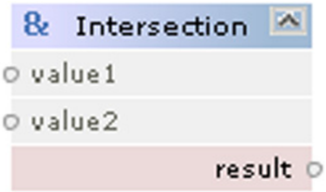
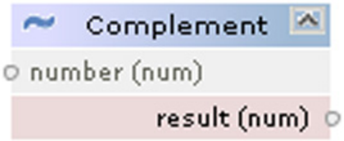
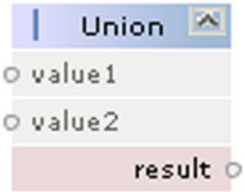
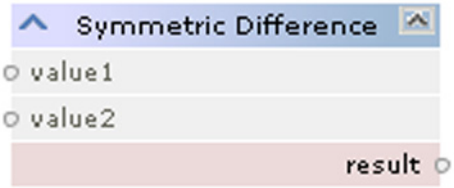
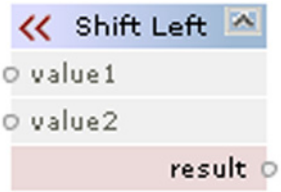
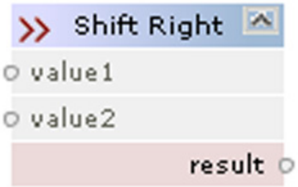
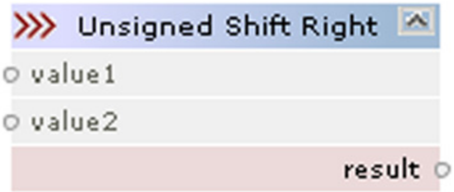
Operator Box	Description/Usage
	At run time, the **Intersection** operator computes the set intersection of bit or Boolean values. (Also listed on the Boolean menu, but not shown by default.)
	At run time, the **Complement** operator computes the complement of a bit value by inverting each bit of the input number.
	At run time, the **Union** operator computes the union of the input values. (Also listed on the Boolean menu, but not shown by default.)
	At run time, the **Symmetric Difference** operator computes the set symmetric difference of bit or Boolean values. (Also listed on the Boolean menu, but not shown by default.)

Table 68 Operators (Java)

Operator Box	Description/Usage
	At run time, the **Shift Left** operator shifts bits to the left; equivalent to multiplication by power of two.
	At run time, the **Shift Right** 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 **Unsigned Shift Right** 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 224 String Methods Menu

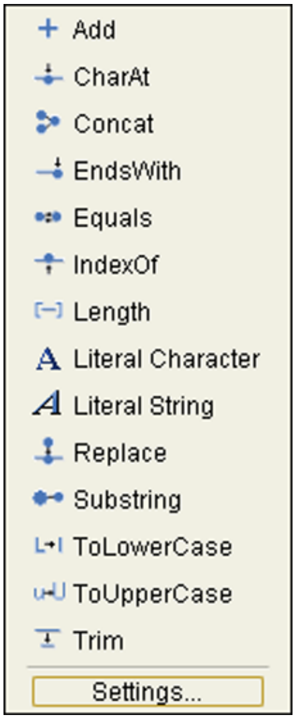


Table 69 String Collaboration Methods (Java)

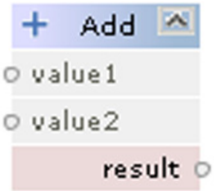
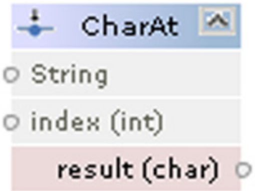
Method Box	Description/Usage
	At run time, if *value1* is a string, the **Add** method converts *value2* to a string and concatenates it to *value1* to produce the *result*.
	At run time, the **CharAt** method returns the character at the specified index, where *index* is the number of characters from the beginning of *String*.

Table 69 String Collaboration Methods (Java)

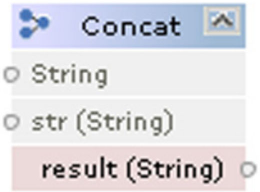
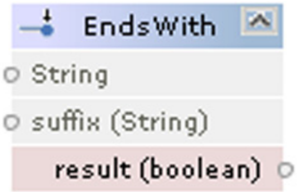
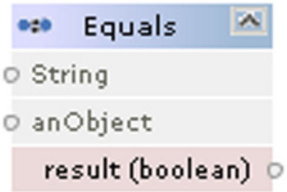
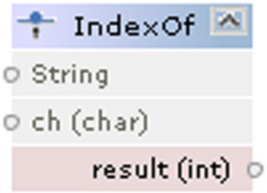
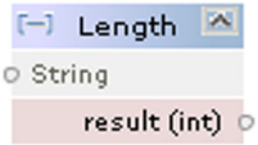
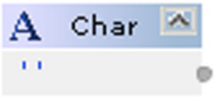
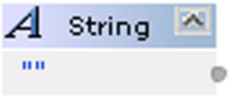
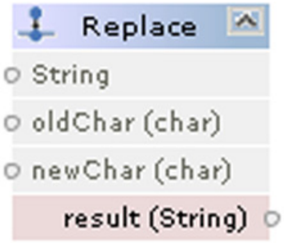
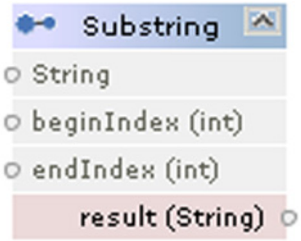
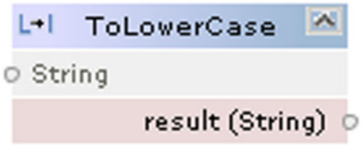
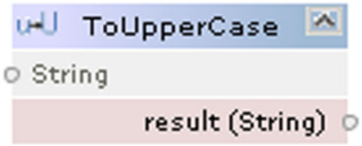
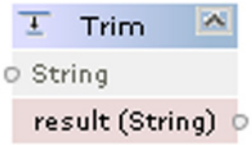
Method Box	Description/Usage
	At run time, the **Concat** method returns the string created by concatenating *str* to the end of *String*.
	At run time, the **EndsWith** method returns Boolean true if *String* ends with the string *suffix*; otherwise, returns Boolean false.
	At run time, the **Equals** method compares *String* with *anObject* 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 **IndexOf** method returns the index within *String* corresponding to the location of the specified character (*ch*), where the index represents the number of characters from the beginning of *String*.
	At run time, the **Length** method returns the length (number of characters) of *String*.
	At run time, the **Literal Character** 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 **Literal String** 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. **Note:** Use *Ctrl+Enter* to insert a line break.

Table 69 String Collaboration Methods (Java)

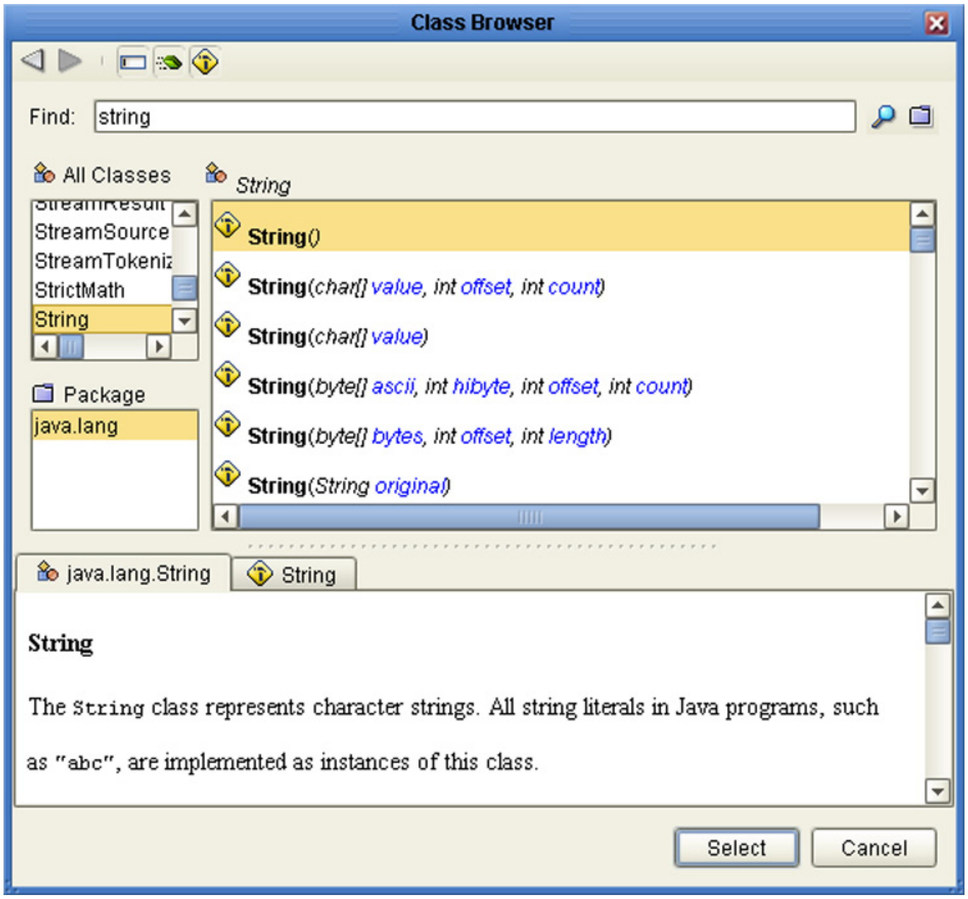
Method Box	Description/Usage
 Replace String oldChar (char) newChar (char) result (String)	At run time, the **Replace** method returns a new string in which all occurrences of *oldChar* are replaced with *newChar*.
 Substring String beginIndex (int) endIndex (int) result (String)	At run time, the **Substring** method returns a string that is a substring of *String*, beginning from *beginIndex* (inclusive) and ending at *endIndex* (exclusive). The indices represent the number of characters from the beginning of *String*.
	At run time, the **ToLowerCase** method converts all characters in *String* to lower case, using the rules of the default locale.
	At run time, the **ToUpperCase** method converts all characters in *String* to upper case, using the rules of the default locale.
	At run time, the **Trim** method trims leading and trailing whitespace from *String*.

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 225 Class Browser Icon

Figure 226 Class/Method Browser



Find: string

All Classes: StreamResult, StreamSource, StreamTokeniz, StrictMath, String

Package: java.lang

String

String()

String(char[] value, int offset, int count)

String(char[] value)

String(byte[] ascii, int hibyte, int offset, int count)

String(byte[] bytes, int offset, int length)

String(String original)

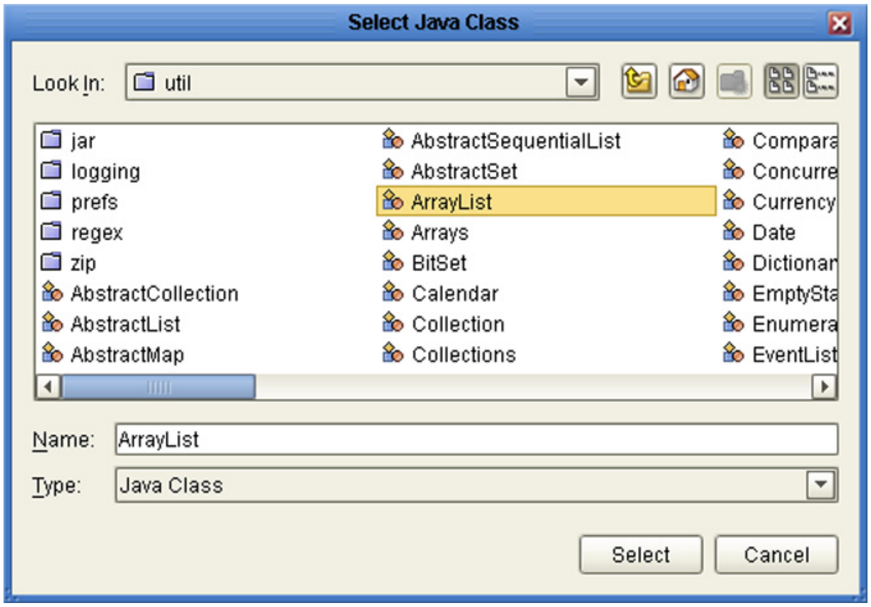
java.lang.String | String

String

The String class represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.

9.5.1 Class Browser Toolbar Icons

Table 70 Method Browser Toolbar Icons

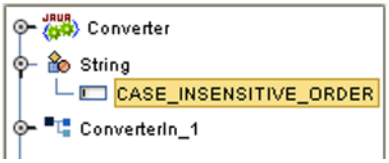
Icon	Command	Function
◁	Step Back	Steps back in your usage history for the current session.
▷	Step Forward	Steps forward in your usage history for the current session.
▭	Show Fields	Displays or hides all fields in the selected class. By default, all fields are displayed; you must click the icon to remove 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 **Find** text box.
📁	Browse Java Package Hierarchy	Presents the *Select Java Class* dialog, with which you can locate classes contained in another installed package. Clicking **Select** displays the contents of the package in the Class Browser.  Select Java Class Look In: jar logging prefs regex zip AbstractCollection AbstractList AbstractMap AbstractSequentialList AbstractSet ArrayList Arrays BitSet Calendar Collection Collections Compara Concurre Currency Date Dictionar EmptySta Enumera EventList Select Cancel Look In: util Name: ArrayList Type: Java Class

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 295).
- If you are adding a field to the Collaboration, it adds the class and field to the source OTD, as shown in Figure 227.

Figure 227 Class and Field

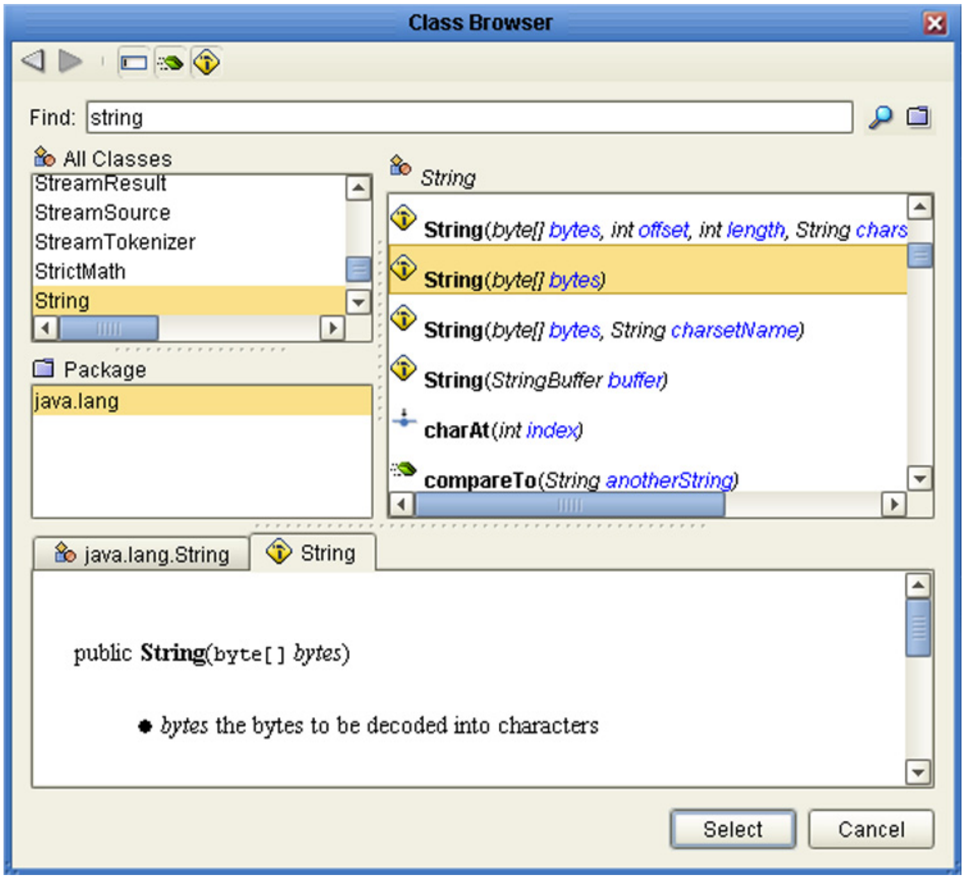


- If you are creating a variable, it adds the variable to the tree in the Business Rules editor (see **Creating and Modifying Variables** on page 350).

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 228).

Figure 228 Class Browser - Select Constructor



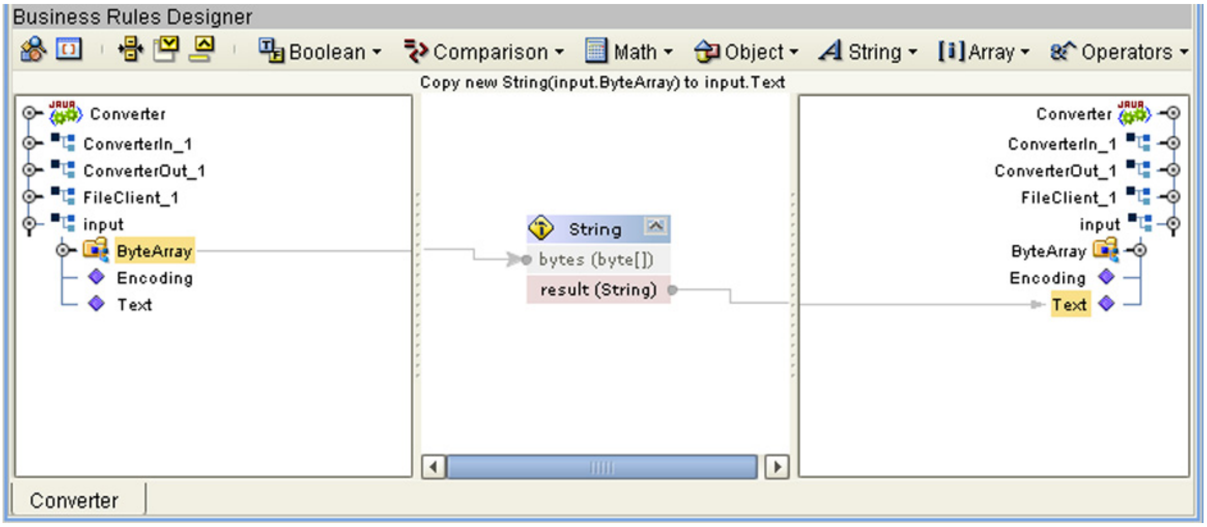
Find: string

public **String**(`byte[]` *bytes*)

- *bytes* the bytes to be decoded into characters

3. Click **Select** to place the constructor box on the Business Rules Designer mapping canvas as shown in Figure 229.

Figure 229 Constructor Box on Mapping Canvas



4 Link the constructor box ports to the appropriate OTD nodes, as shown.

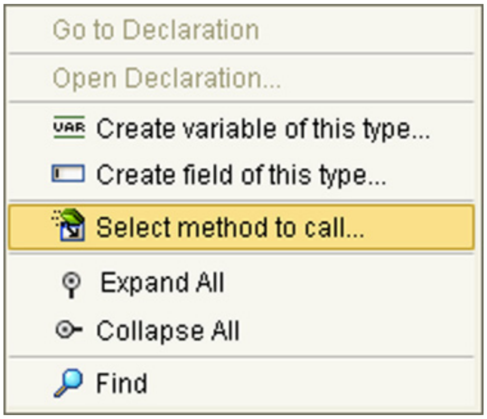
5 **Save** to the Repository.

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:

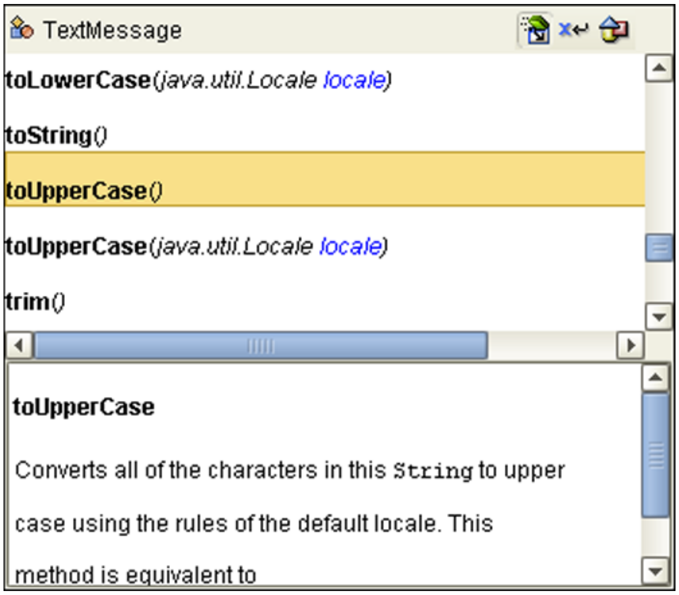
- Right-click on the node to display the context menu and click **Select a method to call** (see Figure 230), or
- Left-click on the node and drag the cursor to the mapping panel.

Figure 230 Node Context Menu



Either action will display the method list for that specific node type, such as that shown in Figure 231 (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 231 Method Dialog (for type TextMessage)



TextMessage

toLowerCase(*java.util.Locale locale*)

toString()

toUpperCase()

toUpperCase(*java.util.Locale locale*)

trim()

toUpperCase

Converts all of the characters in this `String` to upper case using the rules of the default locale. This method is equivalent to

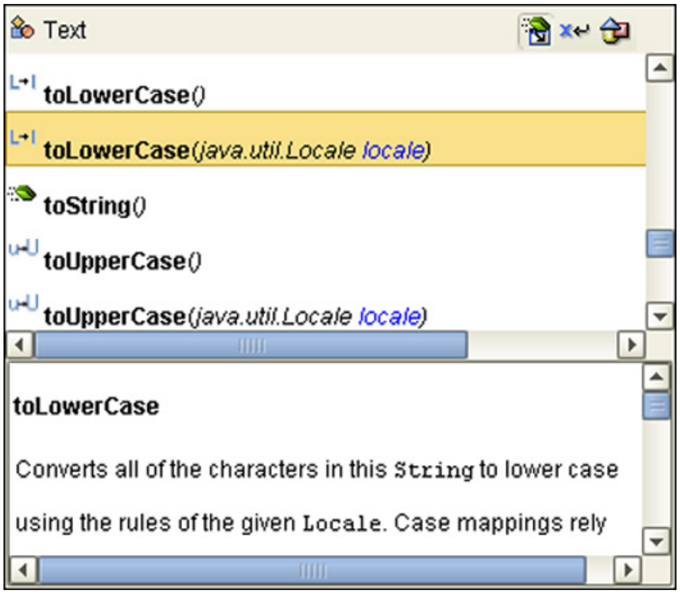
Table 71 Method Dialog Toolbar Icons

Icon	Command	Function
	Show Methods	Displays a list of methods **having** the same type as the selected node. This is the default display when the dialog is launched from the *left* panel of the Business Rules Designer or a method box *output* port.
X↵	Return	Displays a list of methods **returning** the same type as the selected node. This is the default display when the dialog is launched from the *right* panel of the Business Rules Designer or a method box *input* port.
OR A		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 **Creating and Modifying Literals** on page 354.

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 232), either right-click on the node (and release) or left-click on the node and drag to the mapping panel.

Figure 232 Method List for Text



Text

toLowerCase()

toLowerCase(java.util.Locale locale)

toString()

toUpperCase()

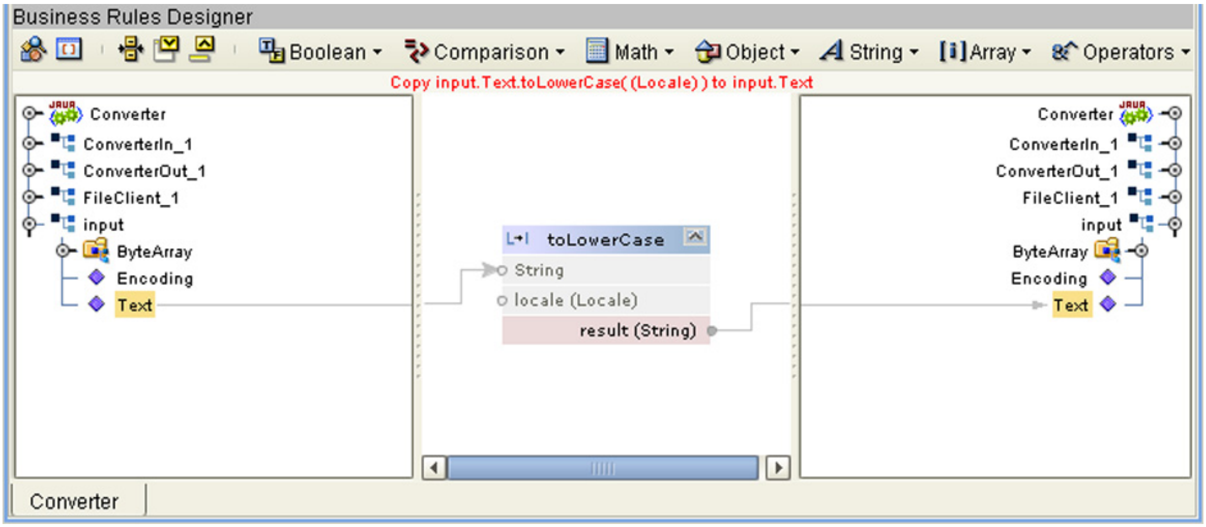
toUpperCase(java.util.Locale locale)

toLowerCase

Converts all of the characters in this String to lower case using the rules of the given Locale. Case mappings rely

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 233, then link the method ports and **Save** to the Repository.

Figure 233 Method Box in Mapping Panel



Copy input.Text.toLowerCase((Locale)) to input.Text

9.7 Auto Type Conversion

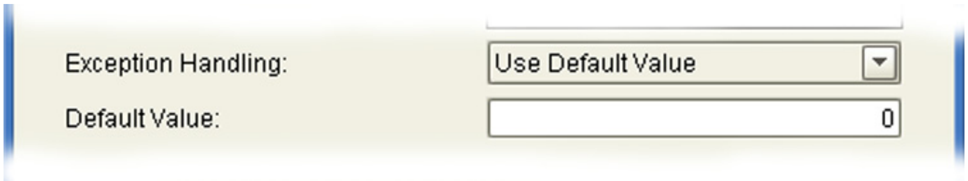
When you map between OTD nodes having different data types, if a conversion method exists for that pair of data types, a dialog is automatically presented in which you can specify certain properties associated with the conversion. The primary content of the dialog is specific to the conversion, examples 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 234). 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 234 Exception Handling Options



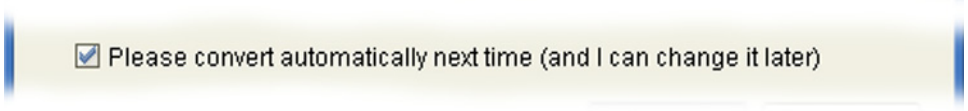
Exception Handling: Use Default Value
Default Value: 0

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 235). 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 235 Automatic Conversion Option



☑ Please convert automatically next time (and I can change it later)

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 72. You can also access these methods from the **typeConverter** node (see **Method-Access Nodes** on page 269). Attempting to map between data types for which no type conversion method exists displays an error notice.

Table 72 Supported Datatype Conversions

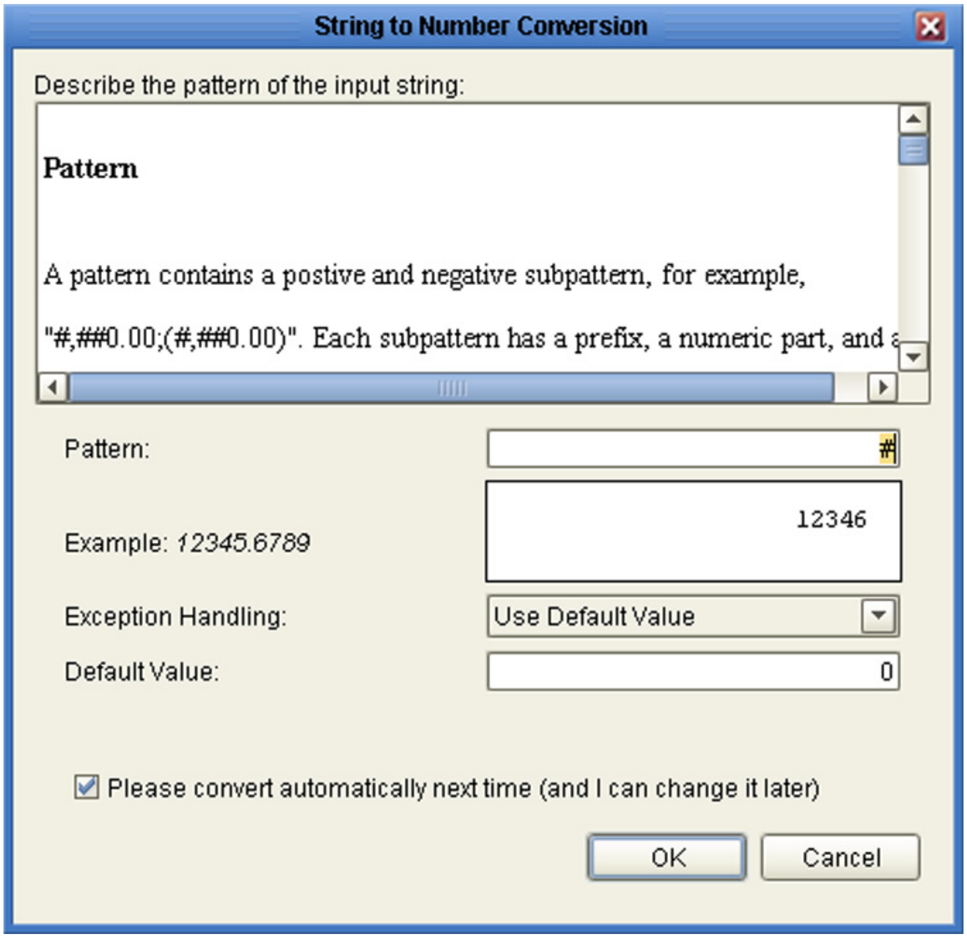
String to Byte Array	Byte Array to String
String to Date	Date to String
String to DateTime	DateTime to String
String to Double	Double to String
String to Float	Float to String
String to Int	Int to String
String to Long	Long to String
String to Short	Short to String
String to SQL Date	String to Timestamp

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 236, where you can specify the pattern of the input string.

Figure 236 Type Conversion - String to Number



String to Number Conversion

Describe the pattern of the input string:

Pattern

A pattern contains a postive and negative subpattern, for example,

"#,##0.00;(#,##0.00)". Each subpattern has a prefix, a numeric part, and a

Pattern: #

Example: *12345.6789* 12346

Exception Handling: Use Default Value

Default Value: 0

☑ Please convert automatically next time (and I can change it later)

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 73 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

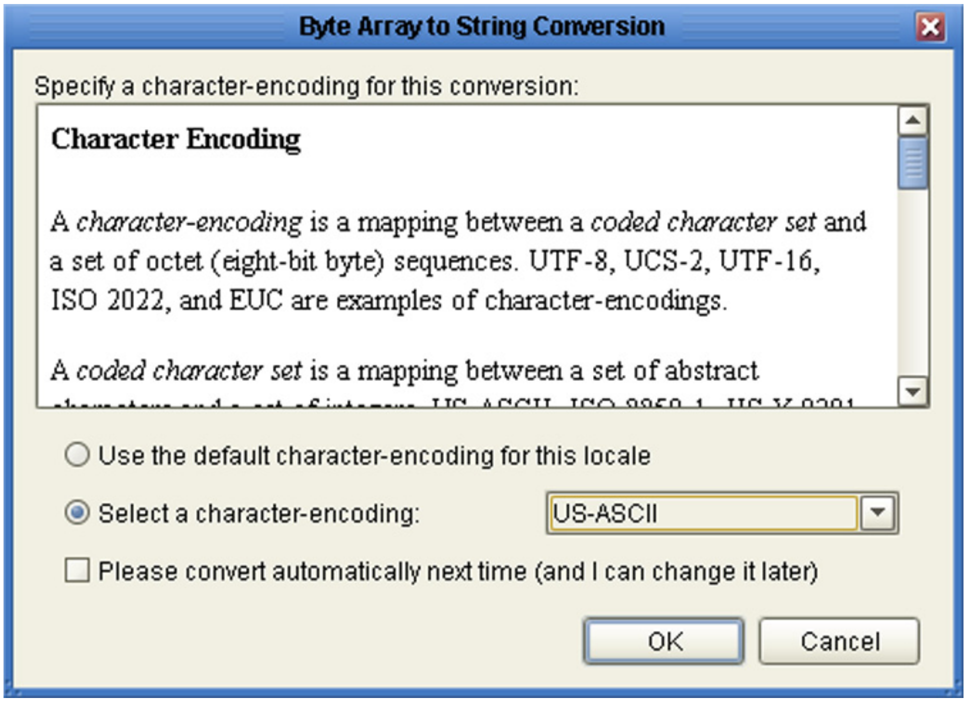
If you have selected *Default Value* for the exception handling, specify an unquoted character string. The output string will be set to this string value.

9.7.2 Character Encoding

Mapping between a *ByteArray* node and a *Text* node automatically presents a dialog such as that shown in Figure 237, where you can specify the character encoding.

Note: *This encoding is independent of the encoding specified for the OTD.*

Figure 237 Type Conversion - Character Encoding



Byte Array to String Conversion

Specify a character-encoding for this conversion:

Character Encoding

A *character-encoding* is a mapping between a *coded character set* and a set of octet (eight-bit byte) sequences. UTF-8, UCS-2, UTF-16, ISO 2022, and EUC are examples of character-encodings.

A *coded character set* is a mapping between a set of abstract

○ Use the default character-encoding for this locale

● Select a character-encoding: US-ASCII

□ Please convert automatically next time (and I can change it later)

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.*

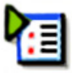
9.8 Using the Collaboration Tester (Java)

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 313 for a description of the Java Debugger, which works with deployed Collaborations.*

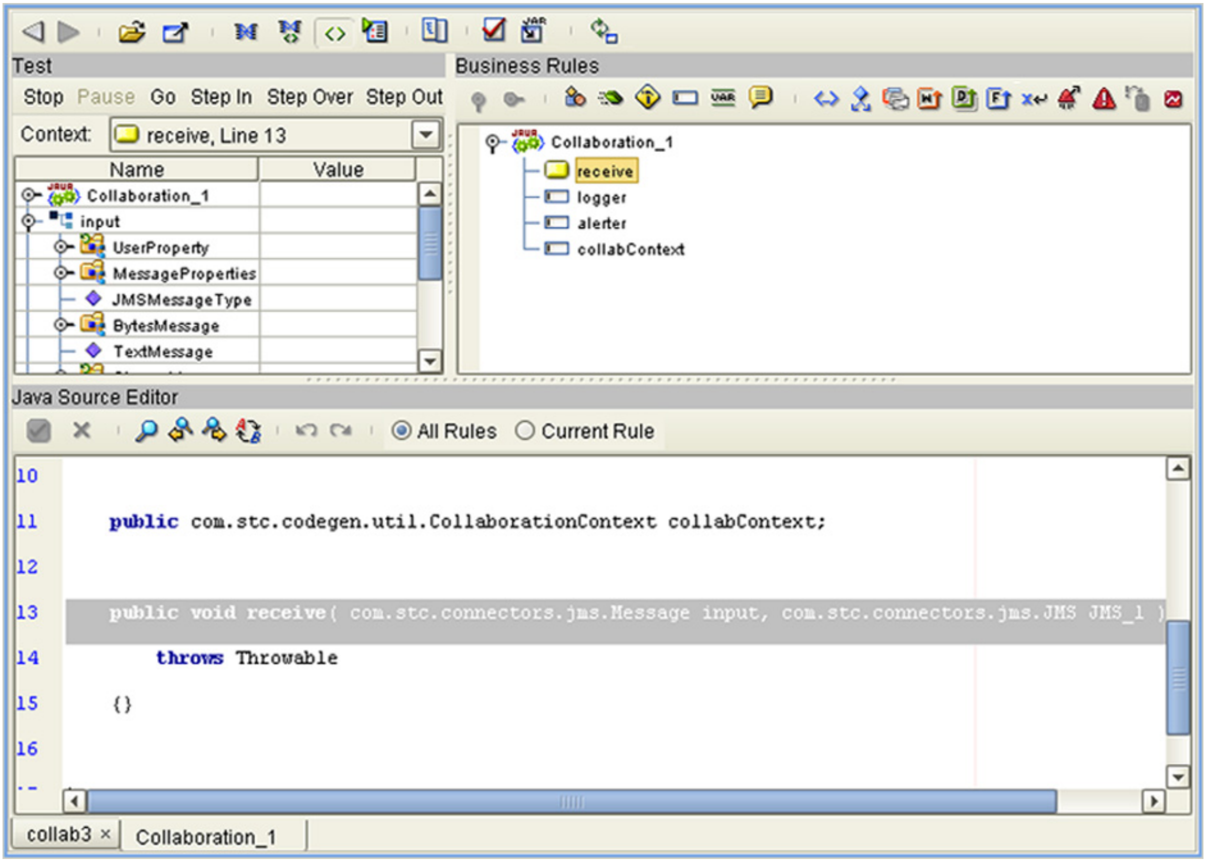
Selecting the **Test Mode** icon in the main Collaboration Editor toolbar (see Figure 238) displays the *Test* panel, which appears at the left of the Business Rules editor (see Figure 239).

Figure 238 Test Mode Icon



Note: *The function of this icon toggles between* **display** *and* **hide**, *depending upon the current state of the tester.*

Figure 239 Collaboration Tester (Java)



Context: receive, Line 13

Name	Value
Collaboration_1	
input	
UserProperty	
MessageProperties	
JMSMessageType	
BytesMessage	
TextMessage	

Collaboration_1
- receive
- logger
- alerter
- collabContext

● All Rules ○ Current Rule

```
    public com.stc.codegen.util.CollaborationContext collabContext;

    public void receive( com.stc.connectors.jms.Message input, com.stc.connectors.jms.JMS JMS_1 )
        throws Throwable
    {}
```

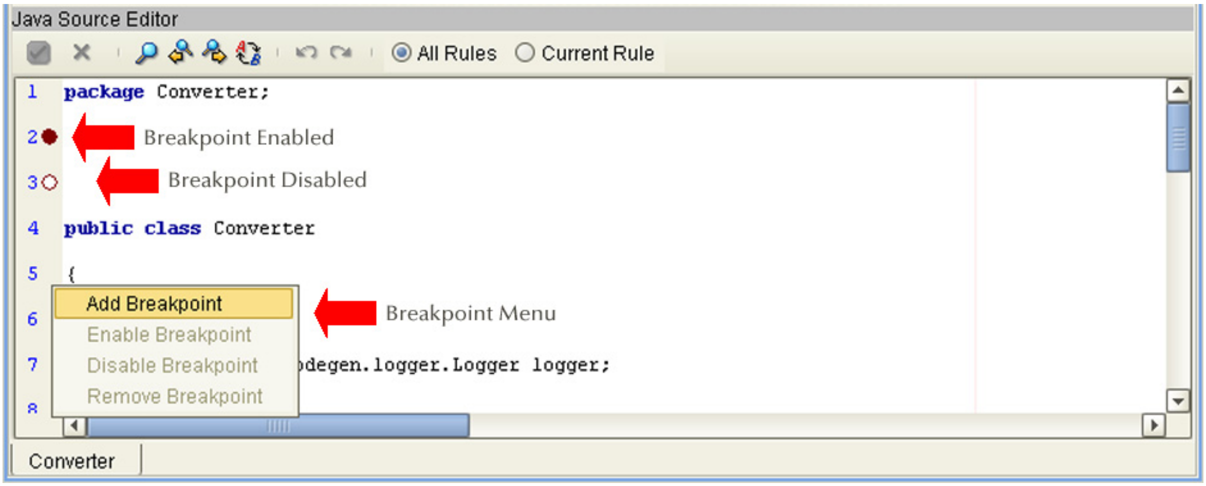
Table 74 Collaboration Tester (Java) Commands

Command	Description
Start / Stop	Toggles to start or stop test procedure. After stopping, clicking **Start** refreshes the tree.
Pause	Pauses test procedure on current line.
Go	Proceed to next step.
Step In	Steps into Java code by one hierarchical level.
Step Over	Steps to next line of code, at same hierarchical level.
Step Out	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 240.

Figure 240 Breakpoints in Java Code



```
package Converter;


public class Converter
{

degen.logger.Logger logger;
```

Alternatively, right-clicking next to a line number displays a breakpoint menu from which you can select the desired action (see Figure 241). In the absence of an existing breakpoint, the menu appears as shown in Figure 240. Right-clicking on an existing breakpoint enables the other options, as appropriate.

Figure 241 Breakpoint Menu

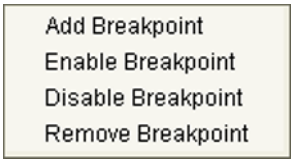


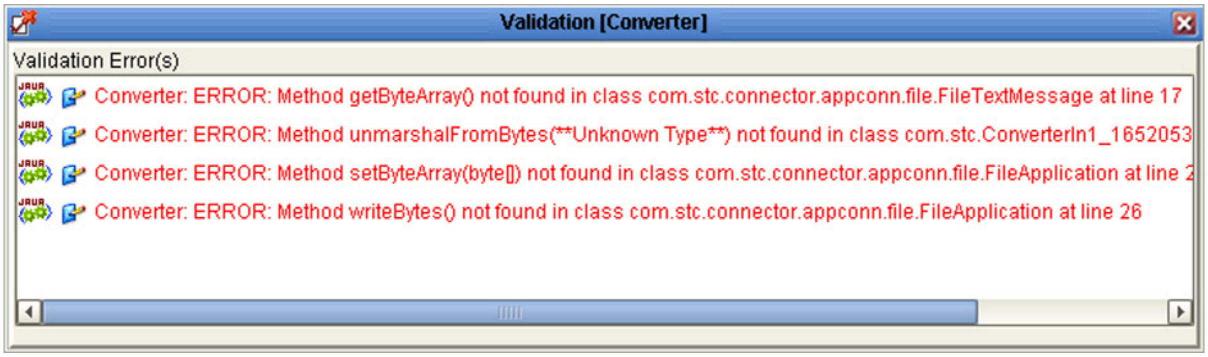
Table 75 Breakpoint Menu Options

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 242). The test will not proceed until these errors are resolved.

Figure 242 Validation Panel



```
Converter: ERROR: Method getByteArray() not found in class com.stc.connector.appconn.file.FileTextMessage at line 17
Converter: ERROR: Method unmarshalFromBytes(**Unknown Type**) not found in class com.stc.ConverterIn1_1652053
Converter: ERROR: Method setByteArray(byte[]) not found in class com.stc.connector.appconn.file.FileApplication at line 2
Converter: ERROR: Method writeBytes() not found in class com.stc.connector.appconn.file.FileApplication at line 26
```

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.

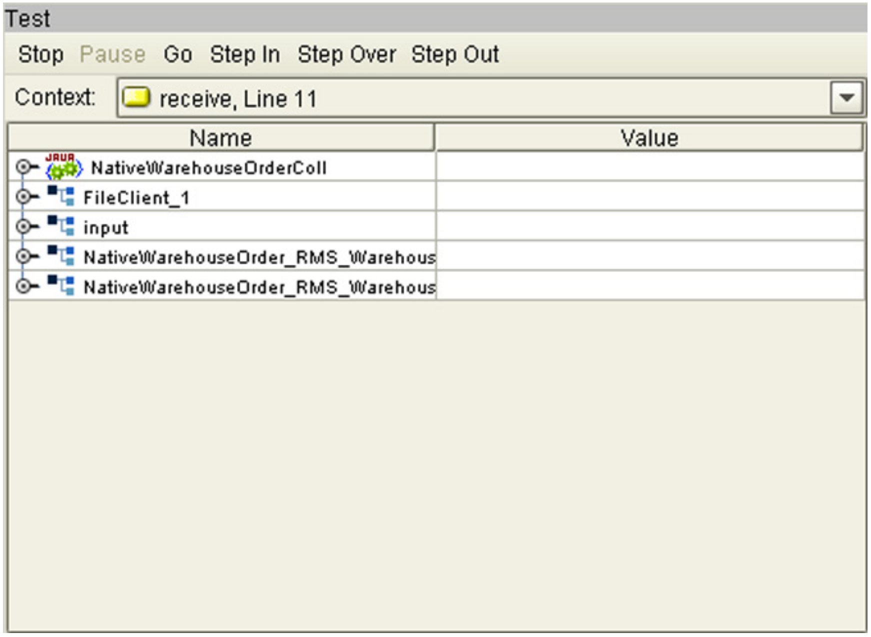
Figure 243 Source Code Mode Icon

3 Click the **Test Mode** icon to display the Collaboration Tester.

Figure 244 Test Mode Icon

4 In the *Test* panel, click **Start** to initialize the Tester. The Collaboration tree will be displayed (see Figure 245), and the Tester will advance to the first rule (highlighted in the *Source Code* panel).

Figure 245 Initial Collaboration Tree

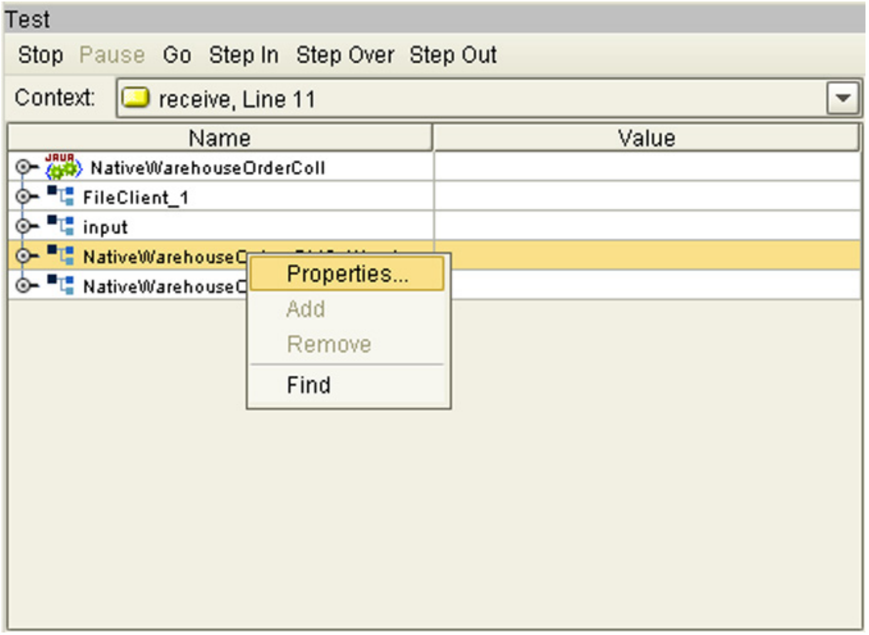


Context: receive, Line 11

Name	Value
NativeWarehouseOrderColl	
FileClient_1	
input	
NativeWarehouseOrder_RMS_Warehous	
NativeWarehouseOrder_RMS_Warehous	

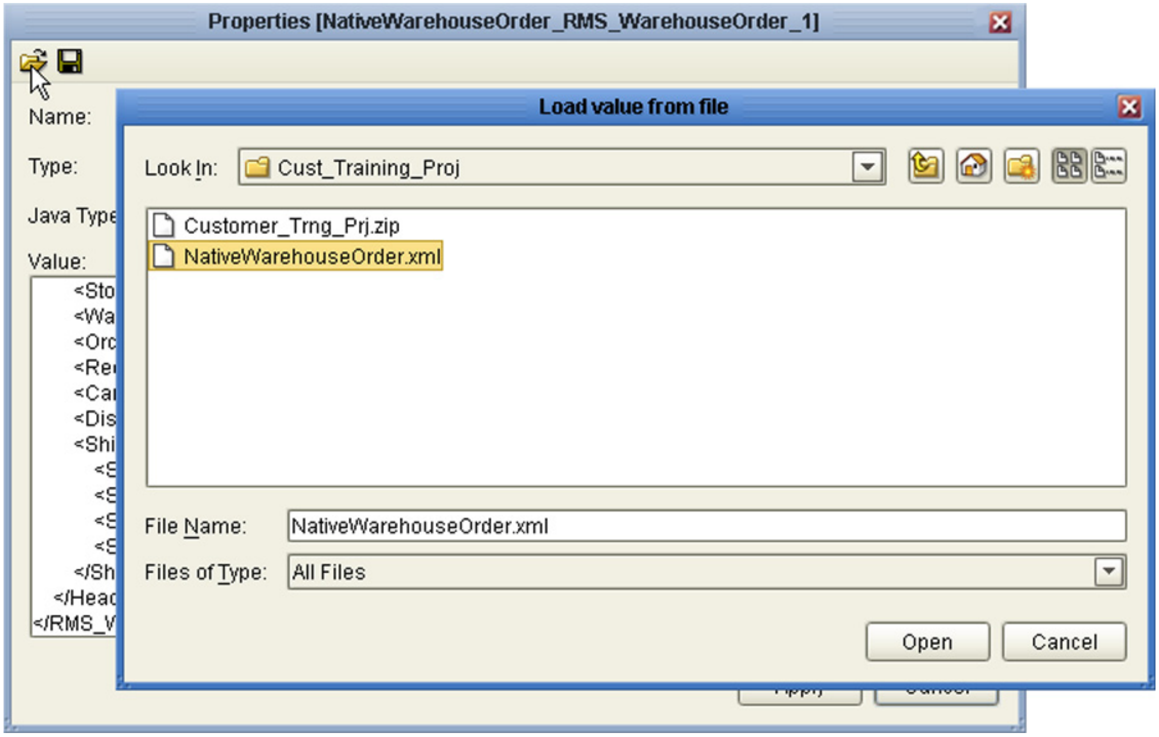
5 In the tester, right-click an OTD root node to display its context menu, and select **Properties** (see Figure 246).

Figure 246 Root Node Context Menu



6 In the *Properties* dialog, invoke the File Browser (see Figure 247).

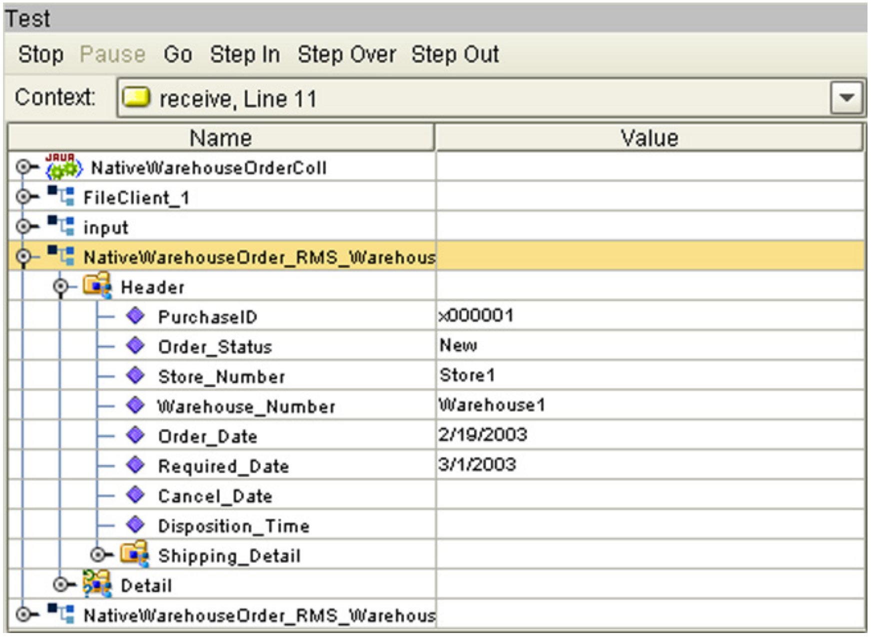
Figure 247 Selecting Input Data File



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 248).

Figure 248 Populated Value Fields



Context: receive, Line 11

Name	Value
NativeWarehouseOrderColl	
FileClient_1	
input	
NativeWarehouseOrder_RMS_Warehous	
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_Warehous	

- 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.

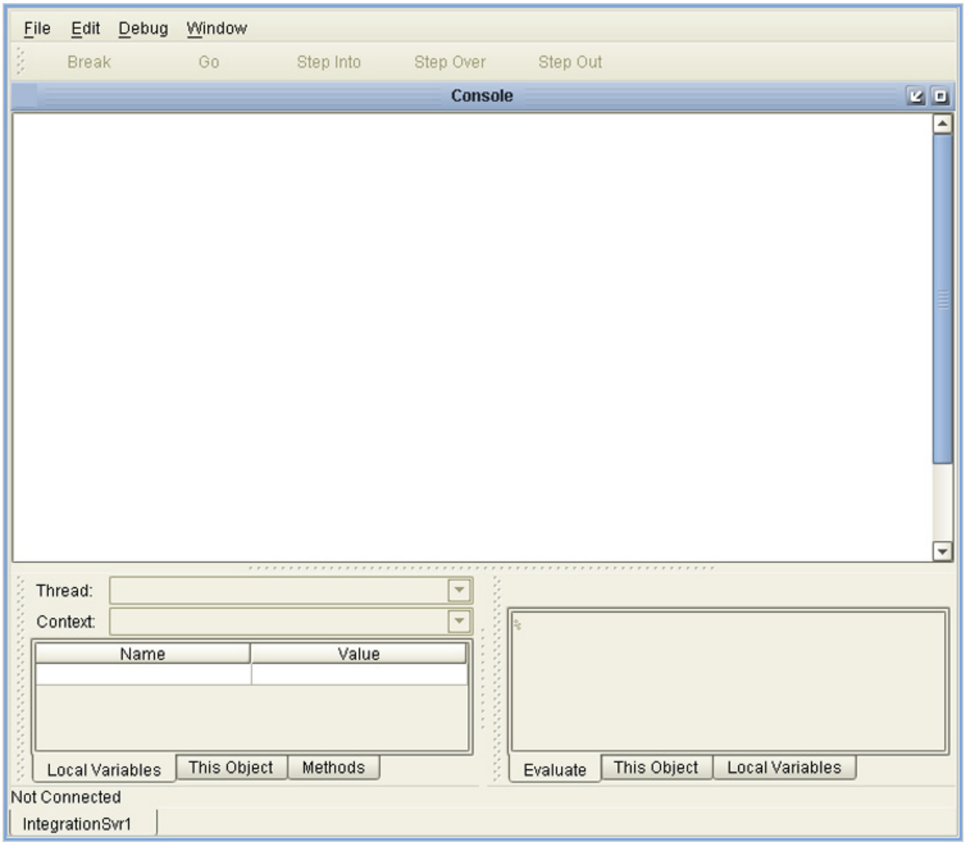
The Java Debugger functions in much the same way as the Collaboration Tester (see **Using the Collaboration Tester (Java)** on page 306), but operates in a run-time Environment. Both are displayed within the Collaboration Editor (Java).

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 249).

Figure 249 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 250), which presents the *Attach to JVM* dialog (see Figure 251).

Figure 250 File Menu

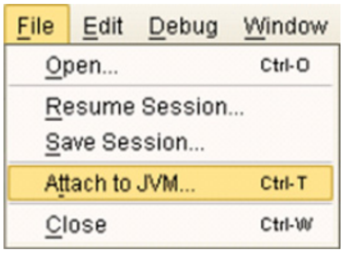
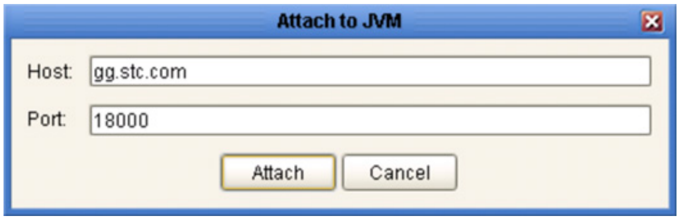


Figure 251 Attach to JVM Dialog


Attach to JVM

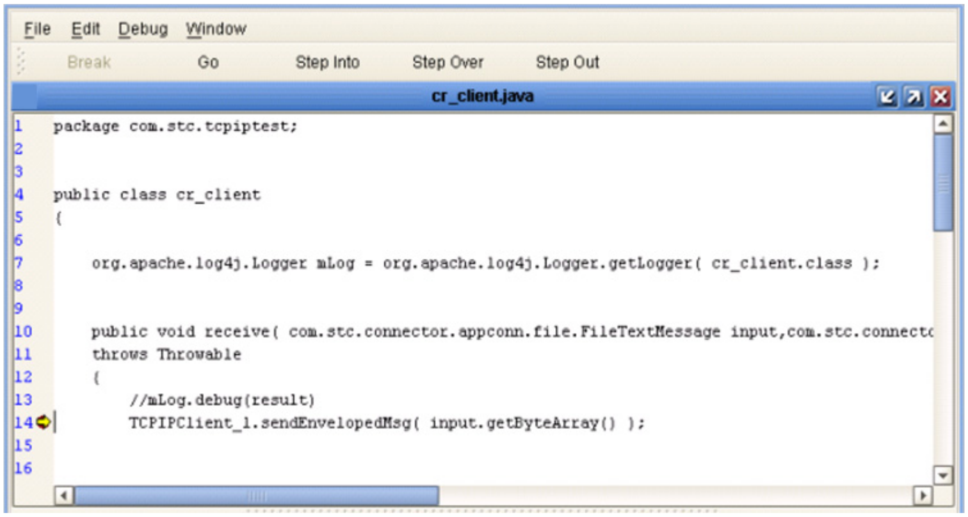
Host: gg.stc.com

Port: 18000

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 252).

Figure 252 Java Source Code Display



```
package com.stc.tcpiptest;


public class cr_client
{

    org.apache.log4j.Logger mLog = org.apache.log4j.Logger.getLogger( cr_client.class );


    public void receive( com.stc.connector.appconn.file.FileTextMessage input,com.stc.connecto
    throws Throwable
    {
        //mLog.debug(result)
        TCPIPClient_1.sendEnvelopedMsg( input.getByteArray() );
```

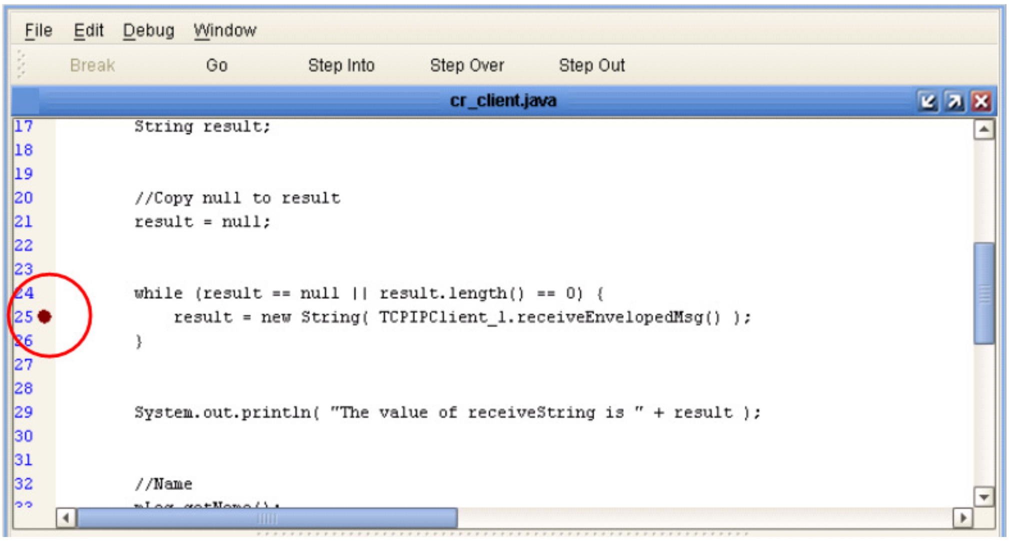
5 You can now set breakpoints to assist in examining and debugging the code.

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 253).

Figure 253 Breakpoint Example



```
        String result;


        //Copy null to result
        result = null;


        while (result == null || result.length() == 0) {
            result = new String( TCPIPClient_1.receiveEnvelopedMsg() );
        }


        System.out.println( "The value of receiveString is " + result );


        //Name
        mLog.getName();
```

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 254). A dialog is presented with which you can select the desired method (seeFigure 255).

Figure 254 Debug Menu

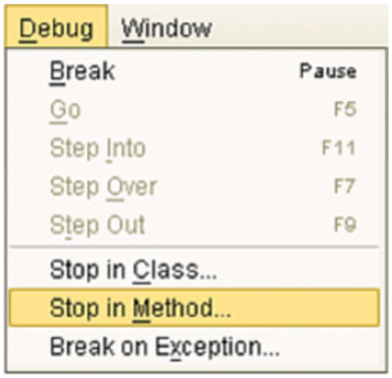
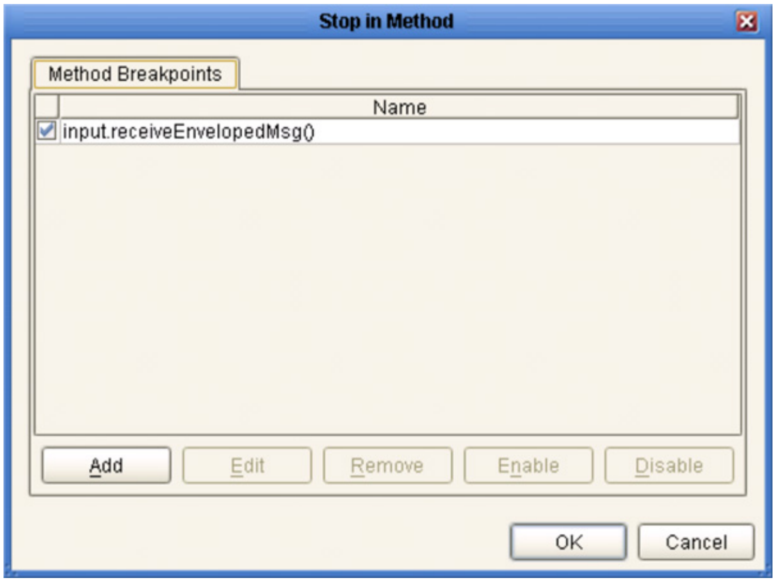


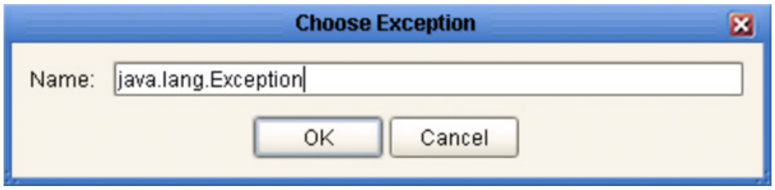
Figure 255 Stop in Method Dialog



☑ input.receiveEnvelopedMsg()

B To break on an exception, select **Break on Exception** from the Debug menu, which presents a *Choose Exception* dialog (see Figure 256). All occurrences of the specified exception are then trapped and reported (see Figure 257).

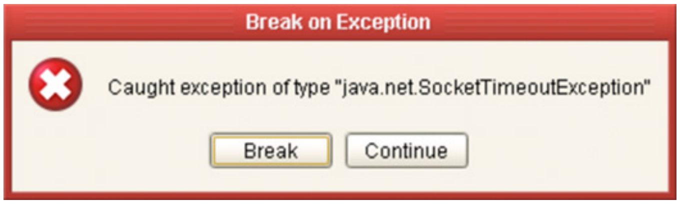
Figure 256 Choose Exception Dialog



Choose Exception

Name: java.lang.Exception

Figure 257 Break on Exception Dialog



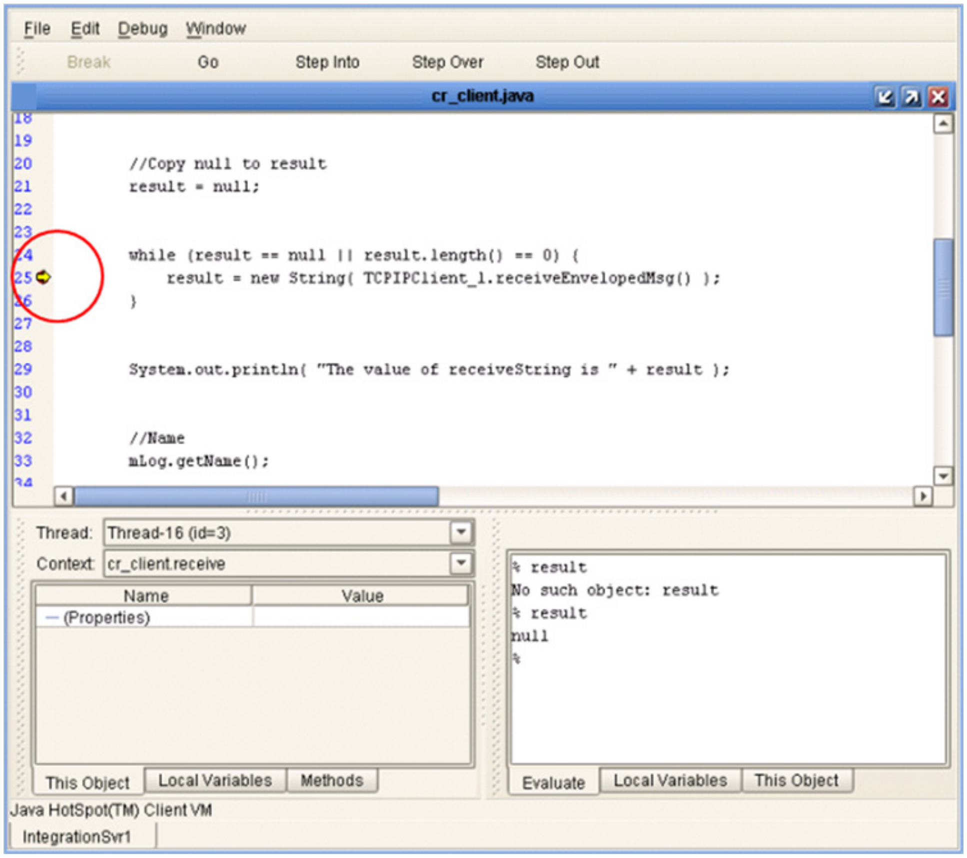
Break on Exception

Caught exception of type "java.net.SocketTimeoutException"

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 258).

Figure 258 Breakpoint Indicator



```
        //Copy null to result
        result = null;


        while (result == null || result.length() == 0) {
            result = new String( TCPIPClient_1.receiveEnvelopedMsg() );
        }


        System.out.println( "The value of receiveString is " + result );


        //Name
        mLog.getName();
```

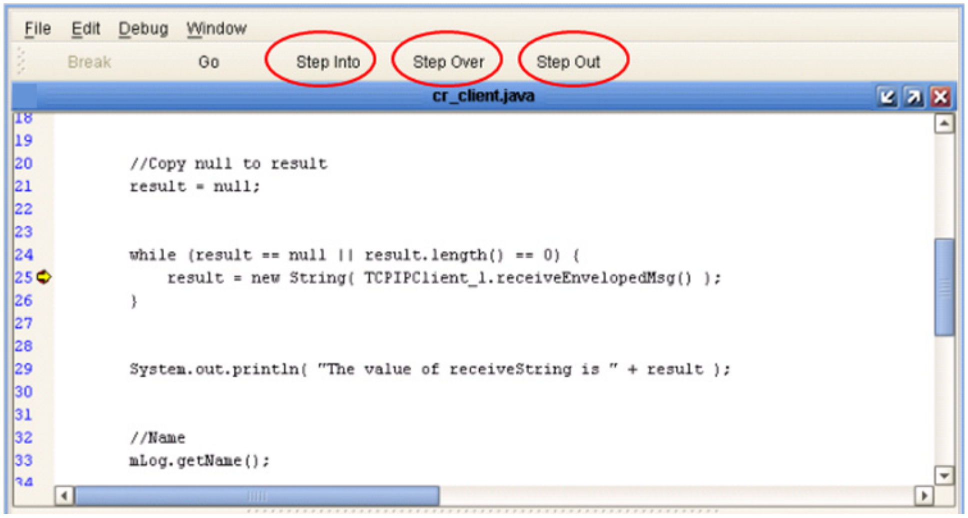
```
% result
No such object: result
% result
null
%
```

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 259 Stepping Into, Over, and Out Commands



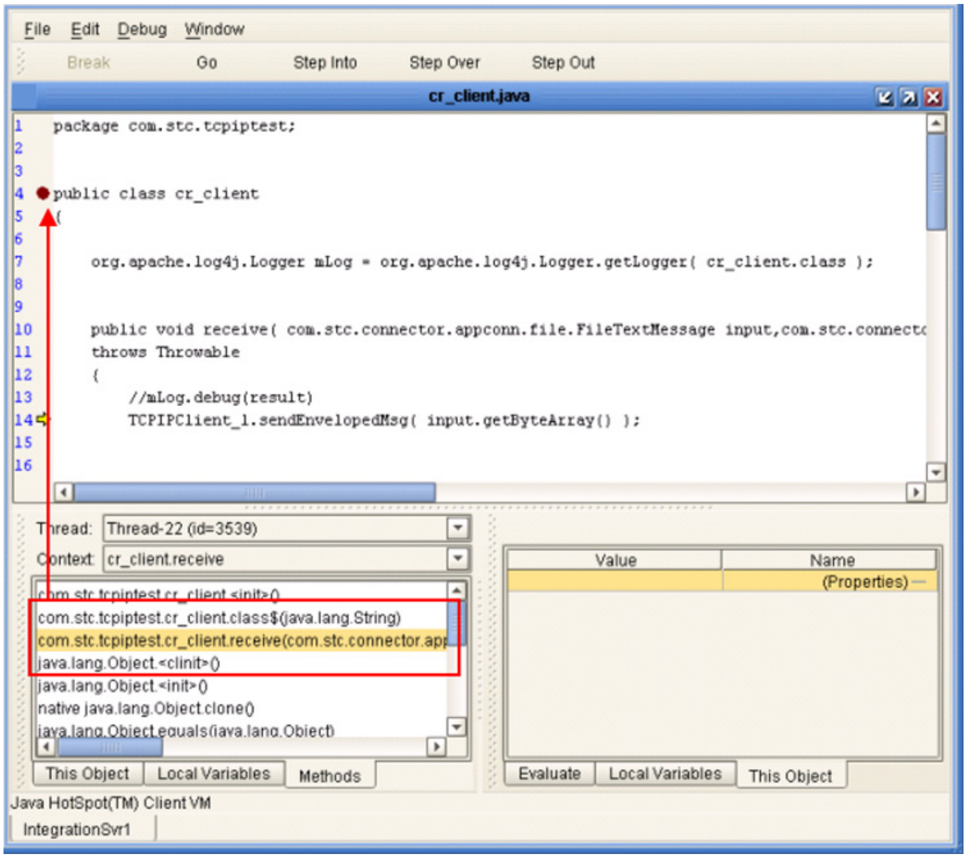
```
//Copy null to result
result = null;


while (result == null || result.length() == 0) {
    result = new String( TCPIPClient_1.receiveEnvelopedMsg() );
}


System.out.println( "The value of receiveString is " + result );


//Name
mLog.getName();
```

- By selecting the **Step Into** option (see Figure 259), 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 260).

Figure 260 Java Thread and Method Display



```
package com.stc.tcpiptest;


public class cr_client
{

    org.apache.log4j.Logger mLog = org.apache.log4j.Logger.getLogger( cr_client.class );


    public void receive( com.stc.connector.appconn.file.FileTextMessage input,com.stc.connectc
    throws Throwable
    {
        //mLog.debug(result)
        TCPIPClient_1.sendEnvelopedMsg( input.getByteArray() );
```

Thread: Thread-22 (id=3539)

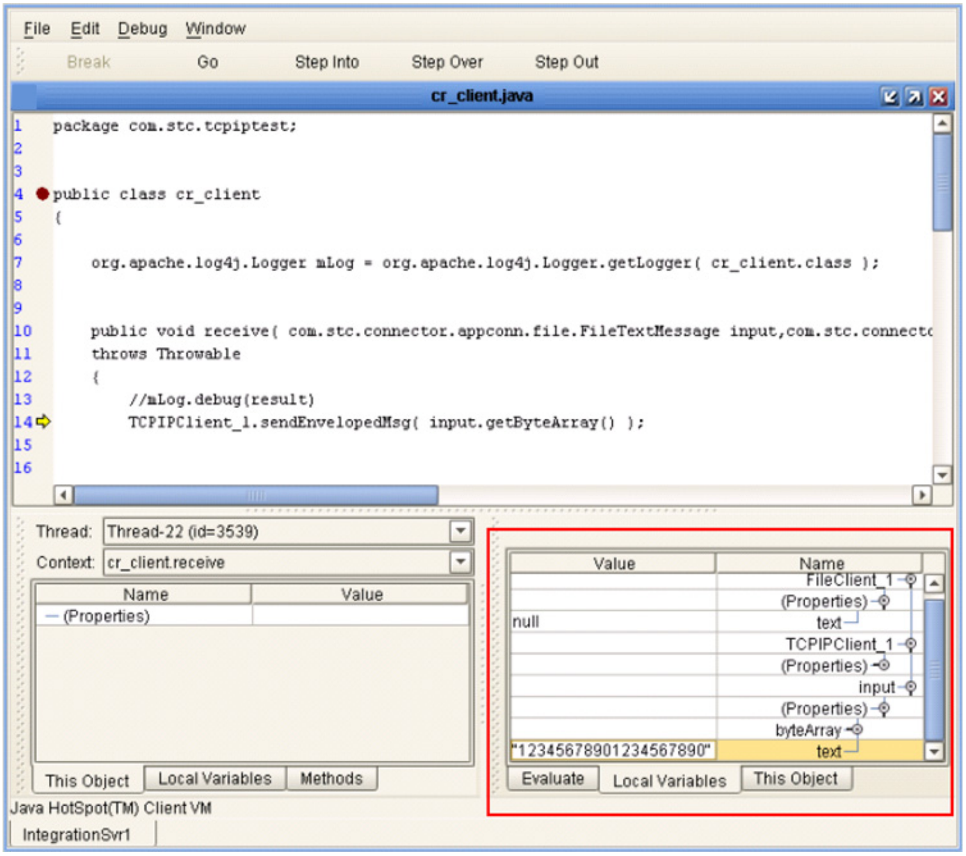
Context: cr_client.receive

```
com.stc.tcpiptest.cr_client.<init>()
com.stc.tcpiptest.cr_client.class$(java.lang.String)
com.stc.tcpiptest.cr_client.receive(com.stc.connector.app
java.lang.Object.<clinit>()
java.lang.Object.<init>()
native java.lang.Object.clone()
java.lang.Object.equals(java.lang.Object)
```

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 261). 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 261 Local Variables Tab



```
package com.stc.tcpiptest;


public class cr_client
{

    org.apache.log4j.Logger mLog = org.apache.log4j.Logger.getLogger( cr_client.class );


    public void receive( com.stc.connector.appconn.file.FileTextMessage input,com.stc.connecto
    throws Throwable
    {
        //mLog.debug(result)
        TCPIPClient_1.sendEnvelopedMsg( input.getByteArray() );
```

Thread: Thread-22 (id=3539)

Context: cr_client.receive

Name	Value
— (Properties)	

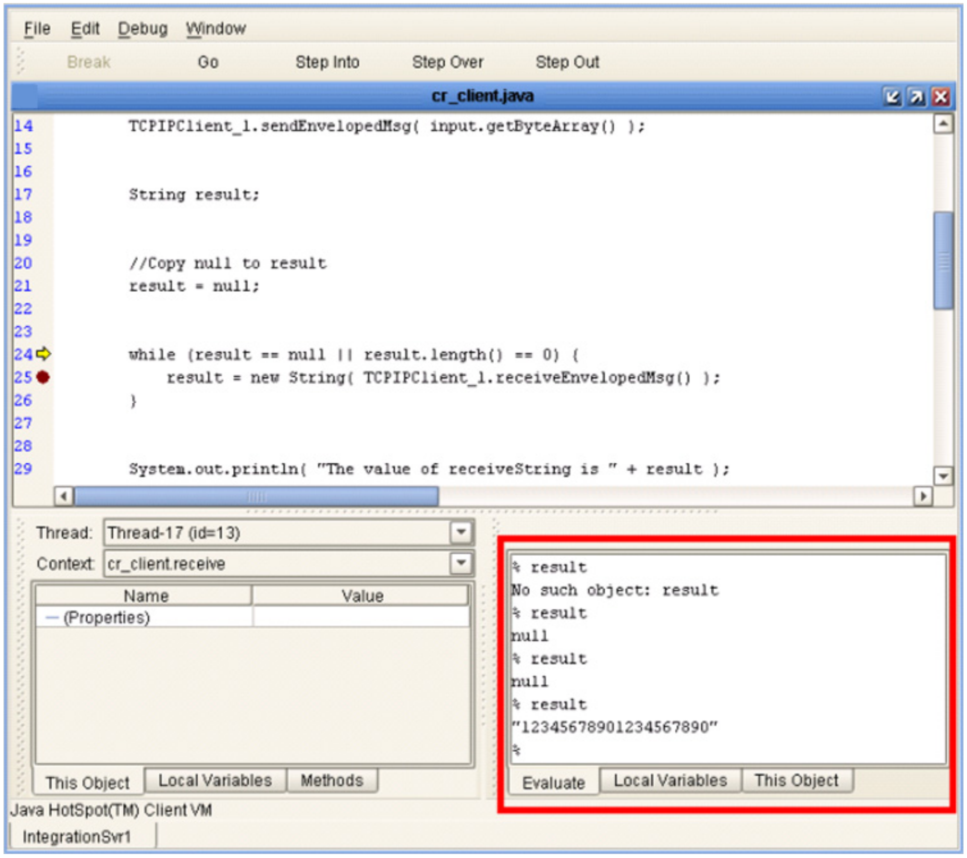
Value	Name
	FileClient_1
	(Properties)
null	text
	TCPIPClient_1
	(Properties)
	input
	(Properties)
	byteArray
"12345678901234567890"	text

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 262).

Figure 262 Evaluate Local Variable

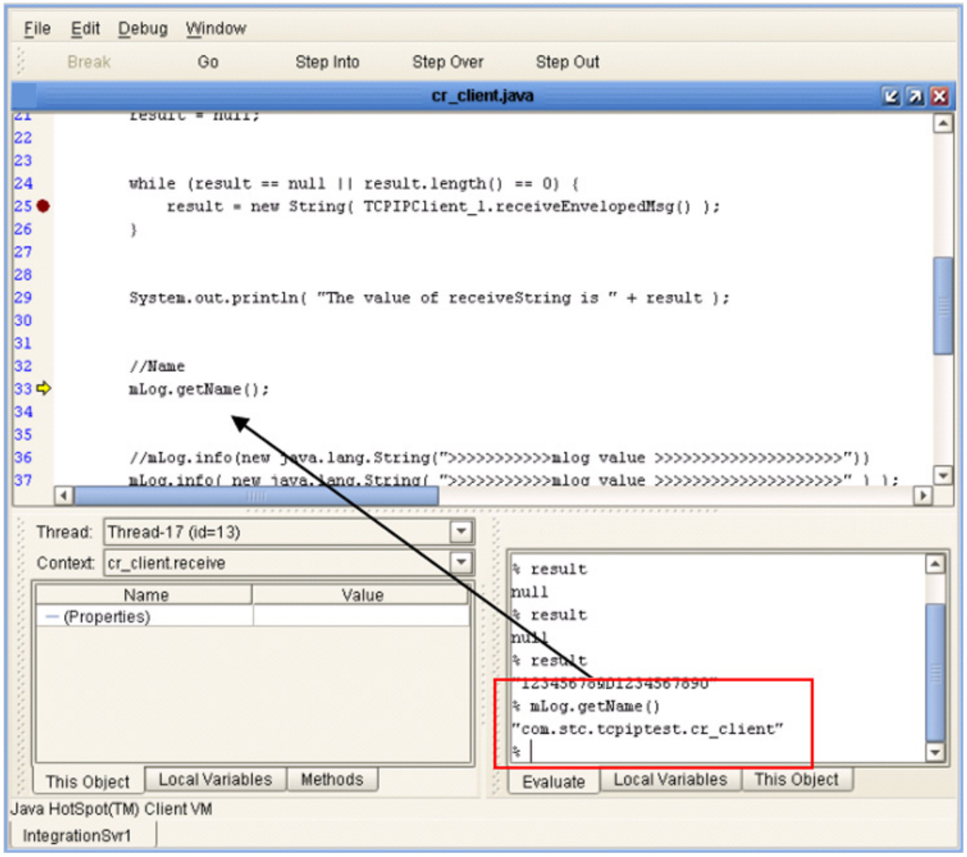


```
        TCPIPClient_1.sendEnvelopedMsg( input.getByteArray() );


        String result;


        //Copy null to result
        result = null;


        while (result == null || result.length() == 0) {
            result = new String( TCPIPClient_1.receiveEnvelopedMsg() );
        }


        System.out.println( "The value of receiveString is " + result );
```

```
% result
No such object: result
% result
null
% result
null
% result
"12345678901234567890"
%
```

You can also inspect the result of a method by entering the method name, following the same syntax format (see Figure 263).

Figure 263 Evaluate Method



```
        result = null;


        while (result == null || result.length() == 0) {
            result = new String( TCPIPClient_1.receiveEnvelopedMsg() );
        }


        System.out.println( "The value of receiveString is " + result );


        //Name
        mLog.getName();


        //mLog.info(new java.lang.String(">>>>>>>>>>>mlog value >>>>>>>>>>>>>>>>>>>>"))
        mLog.info( new java.lang.String( ">>>>>>>>>>>mlog value >>>>>>>>>>>>>>>>>>>>" ) );
```

Thread: Thread-17 (id=13)

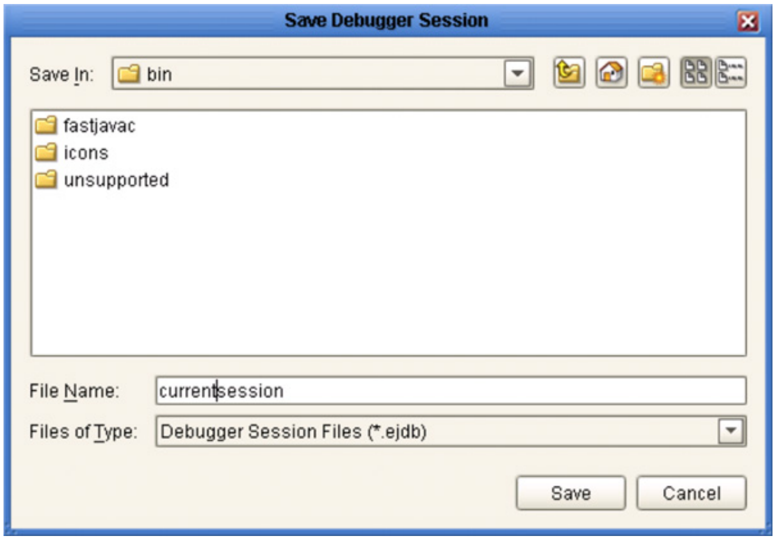
Context: cr_client.receive

```
% result
null
% result
null
% result
"123456789D1234567890"
% mLog.getName()
"com.stc.tcpiptest.cr_client"
% |
```

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 264), in which you specify the file name and location.

Figure 264 Save Debugger Session Dialog



Save Debugger Session

Save In: bin

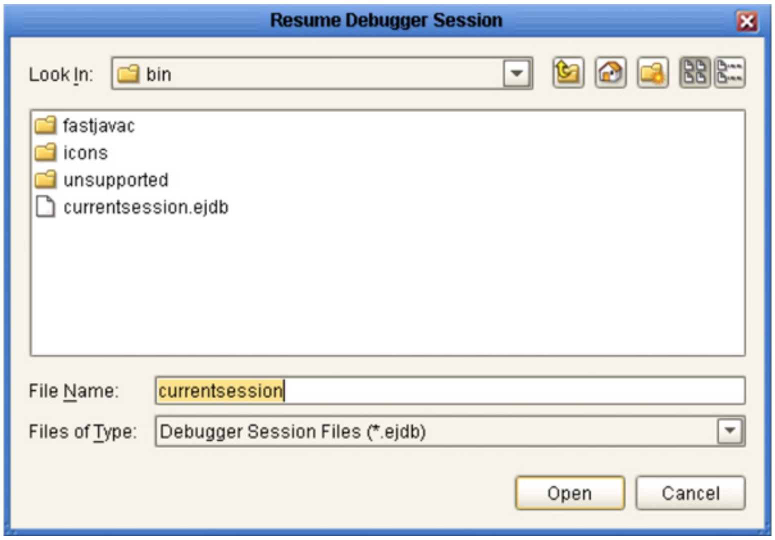
fastjavac
icons
unsupported

File Name: currentsession

Files of Type: Debugger Session Files (*.ejdb)

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 265), in which you specify the file name and location.

Figure 265 Resume Debugger Session Dialog



Resume Debugger Session

Look In: bin

fastjavac
icons
unsupported
currentsession.ejdb

File Name: currentsession

Files of Type: Debugger Session Files (*.ejdb)

Chapter 10

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 326
- **Editing Collaboration Definition Properties** on page 334
- **Creating Classes** on page 337
- **Creating Methods** on page 339
- **Creating Constructors** on page 343
- **Creating Fields** on page 346
- **Creating Arrays** on page 348
- **Creating and Modifying Variables** on page 350
- **Creating and Modifying Literals** on page 354
- **Generating Alerts** on page 355
- **Creating Log Entries** on page 359
- **Importing Third-Party Java Classes** on page 362
- **Adding Java classes to Collaboration Definitions** on page 364
- **Copying OTD Nodes** on page 366
- **Using Try-Catch** on page 368
- **Using Byte-Array Converters** on page 371
- **Validating Java-based Collaboration Definitions** on page 386

10.1 Creating a Java-based Collaboration Definition

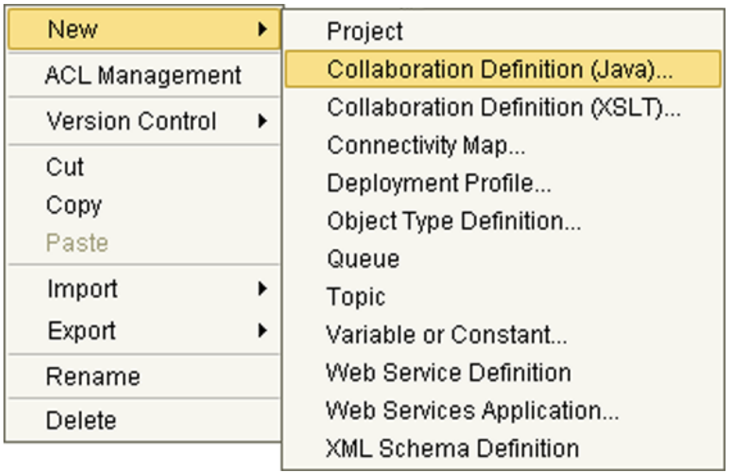
The Collaboration Definition Wizard (Java) guides you through the initial phases of creating a Java-based Collaboration Definition, and then invokes the Collaboration Editor (Java).

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 266).

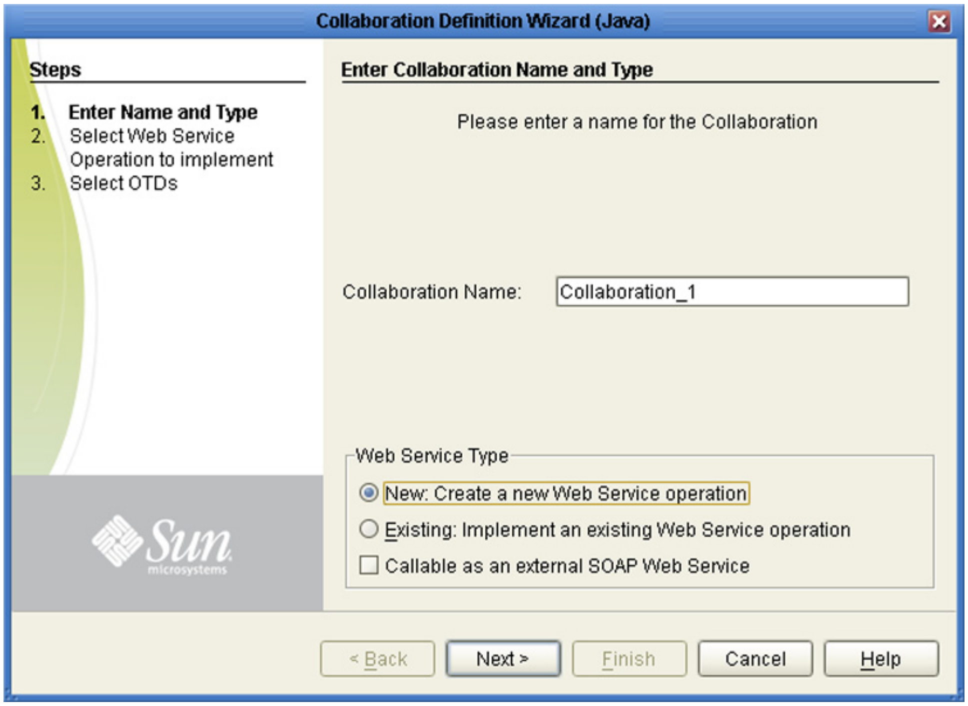
Figure 266 Project Context Menu - New Web Service Definition



2 Select **New > Collaboration Definition (Java)** to invoke the Collaboration Definition Wizard (Java).

3 Enter a **Name** for your Collaboration, as shown in Figure 267.

Figure 267 Initial Wizard Dialog


Collaboration Definition Wizard (Java)
Steps
1. Enter Name and Type
2. Select Web Service Operation to implement
3. Select OTDs
Enter Collaboration Name and Type
Sun microsystems
< Back
Next >
Finish
Cancel
Help

Please enter a name for the Collaboration

Collaboration Name: Collaboration_1

Web Service Type

● New: Create a new Web Service operation

○ Existing: Implement an existing Web Service operation

□ Callable as an external SOAP Web Service

4 Select a Web Service Type, which can be either:

- A **new** web service operation.
- An **existing** web service operation (including an eInsight business process or an OTD).

5 If you selected a **new** web service operation, and you want to expose the Collaboration Definition as an external web service, check the box entitled **Callable as an external SOAP 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.*

Note: *XSD OTDs that are used in SOAP-callable Java Collaboration Definitions must not contain **imported** or **included** XSDs. If you need this functionality, you must create an XSD Object based on the XML Schema Definition(s), and export the needed elements as XSD Nodes. The XSD Nodes can then be used in the Collaboration Definition. See* **Previewing and Exporting XSD Nodes** *on page 433.*

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; see:

- **New Web Service Operation** on page 328
- **Existing Web Service Operation** on page 332

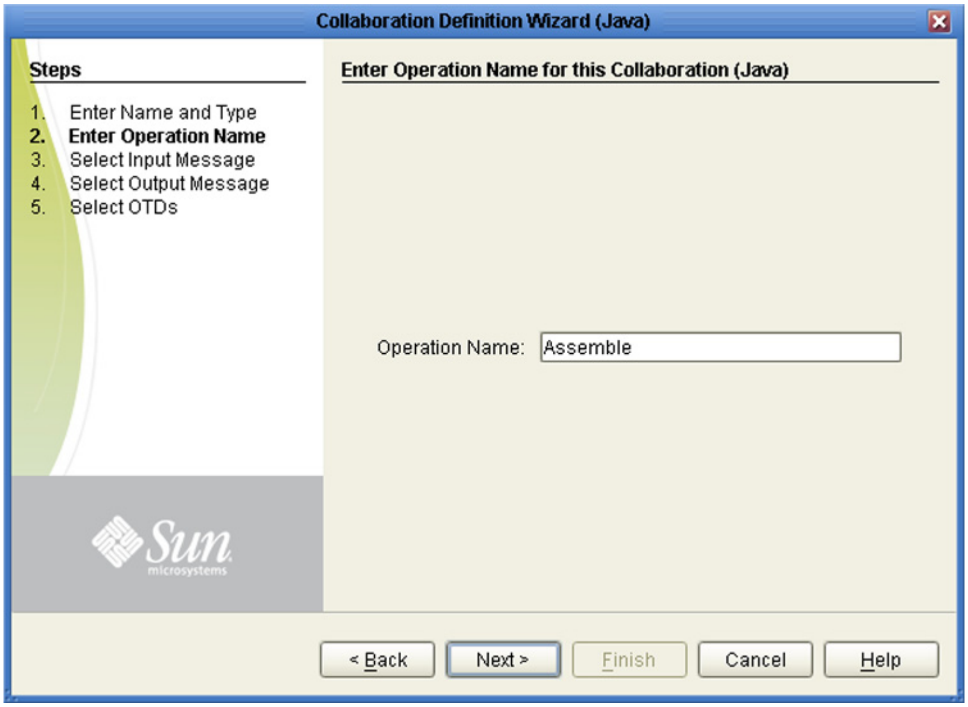
10.1.1 New Web Service Operation

If you selected a new web service operation, you will be presented with the Wizard dialog shown in Figure 268.

To create a new web service operation

1 Enter an operation name, as shown in Figure 268. This will become the *method* that can be used to invoke the Java-based Collaboration as a web service.

Figure 268 New Web Service: Operation Name



Collaboration Definition Wizard (Java)

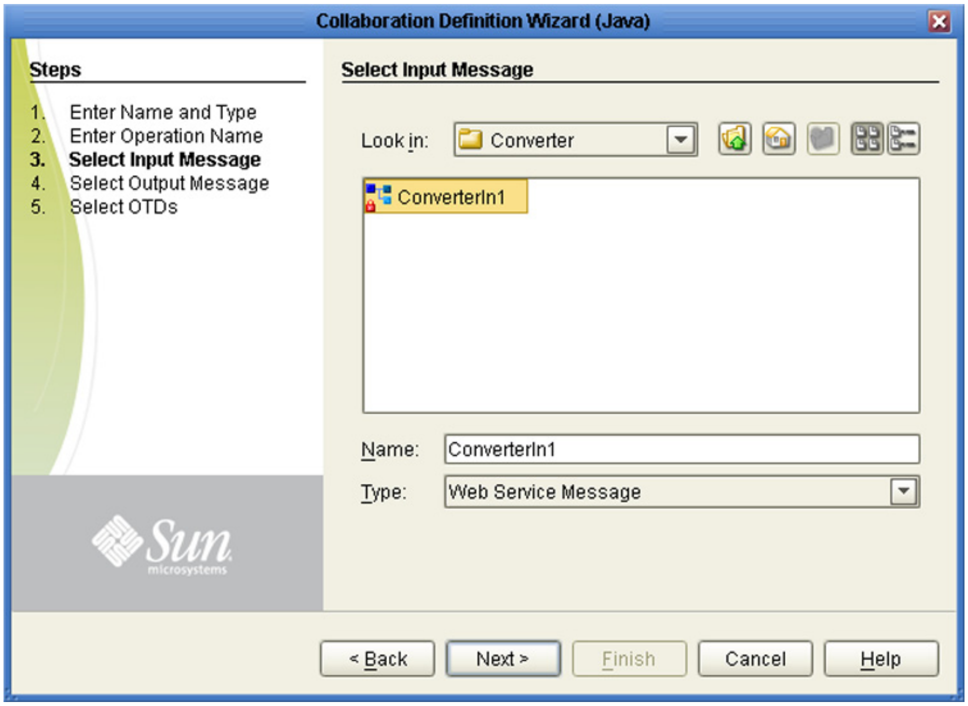
Enter Operation Name for this Collaboration (Java)

Operation Name: Assemble

2 Click **Next** to proceed to the next Wizard dialog.

3 Select the input web service message, (for example, *ConverterIn1* - shown in Figure 269).

Figure 269 New Web Service: Input Message



Look in: Converter

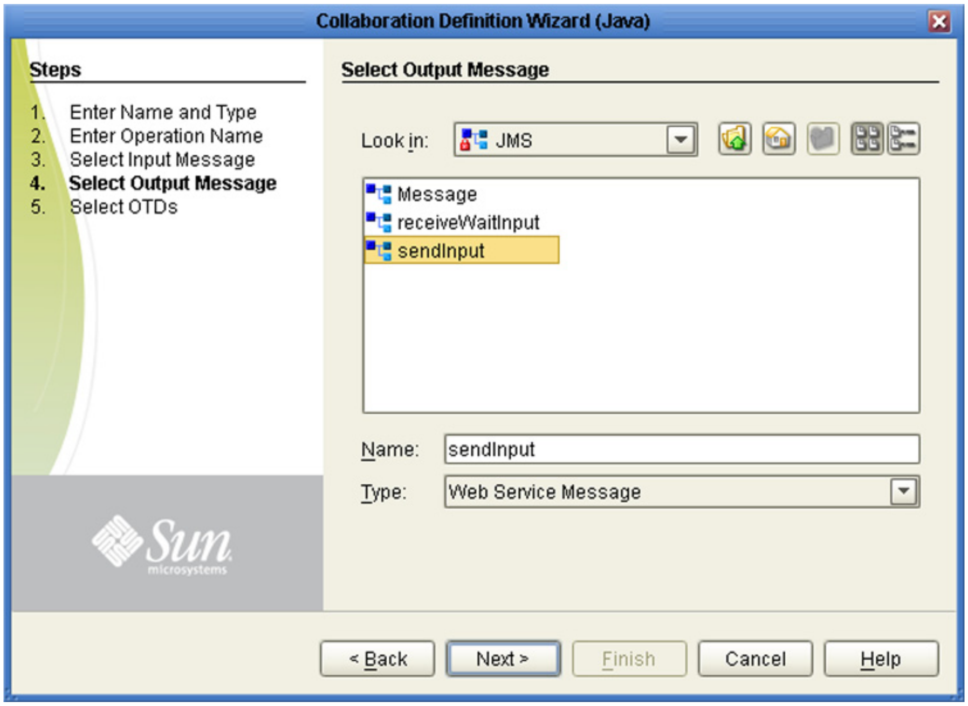
Name: ConverterIn1

Type: Web Service 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 270).

Figure 270 New Web Service: Output Message



Collaboration Definition Wizard (Java)

Steps

1. Enter Name and Type
2. Enter Operation Name
3. Select Input Message
4. **Select Output Message**
5. Select OTDs

Select Output Message

Look in: JMS

- Message
- receiveWaitInput
- sendInput

Name: sendInput

Type: Web Service Message

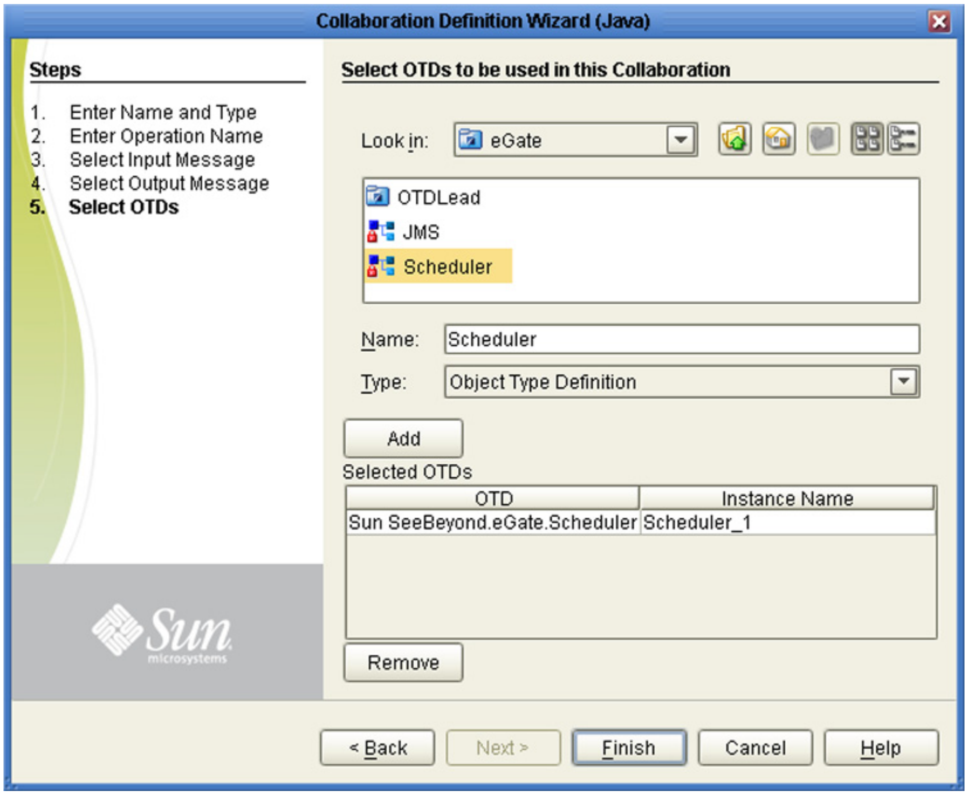
6 Click **Next** to proceed to the next Wizard dialog.

7 Select an auxiliary OTD, if necessary (see Figure 271). This step is optional, and is intended to support additional functionality. For example:

- For Collaborations that read from or write to a topic or queue, you need to add the **JMS OTD**.
- If the Collaboration is triggered by a Scheduler adapter, you need to add the **Scheduler OTD** (shown as an example in Figure 271).

Note: *You may already have selected OTDs in the preceding steps — if so, you should not repeat them here.*

Figure 271 New Web Service: Auxiliary OTD



Collaboration Definition Wizard (Java)

Steps

1. Enter Name and Type
2. Enter Operation Name
3. Select Input Message
4. Select Output Message
5. **Select OTDs**

Select OTDs to be used in this Collaboration

Look in: eGate

OTDLead
JMS
Scheduler

Name: Scheduler

Type: Object Type Definition

Selected OTDs

OTD	Instance Name
Sun SeeBeyond.eGate.Scheduler	Scheduler_1

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 Operation

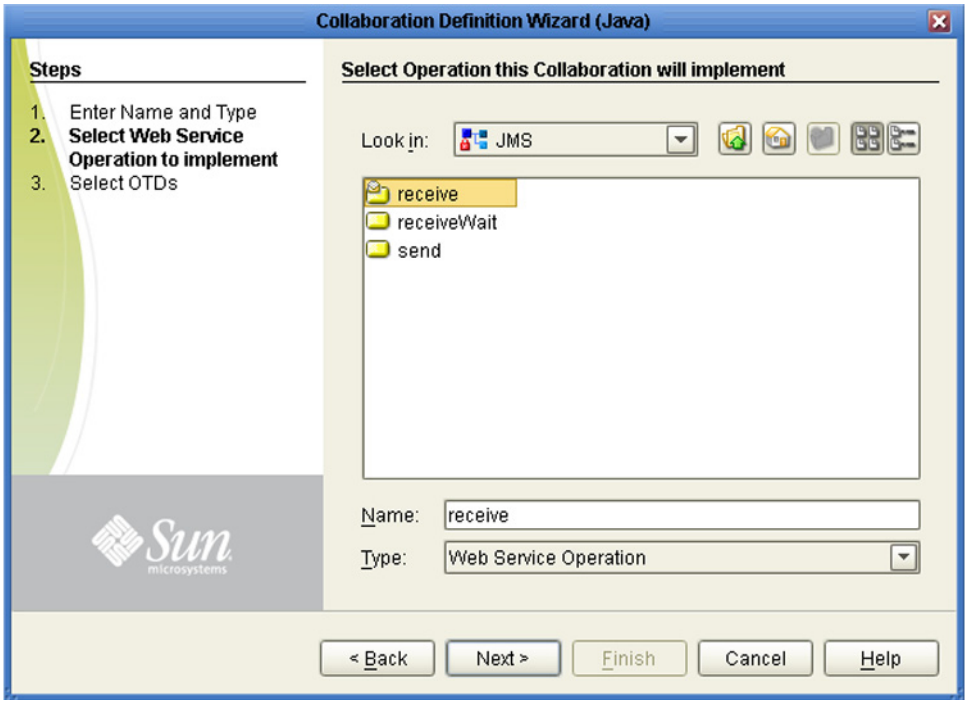
If you selected an existing web service operation, you will be presented with the Wizard dialog shown in Figure 272.

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 operation

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 272).

Figure 272 Existing Web Service: Select Operation


Collaboration Definition Wizard (Java)
Steps
1. Enter Name and Type
2. Select Web Service Operation to implement
3. Select OTDs
Select Operation this Collaboration will implement
receive
receiveWait
send
< Back
Next >
Finish
Cancel
Help

Look in: JMS

Name: receive

Type: Web Service Operation

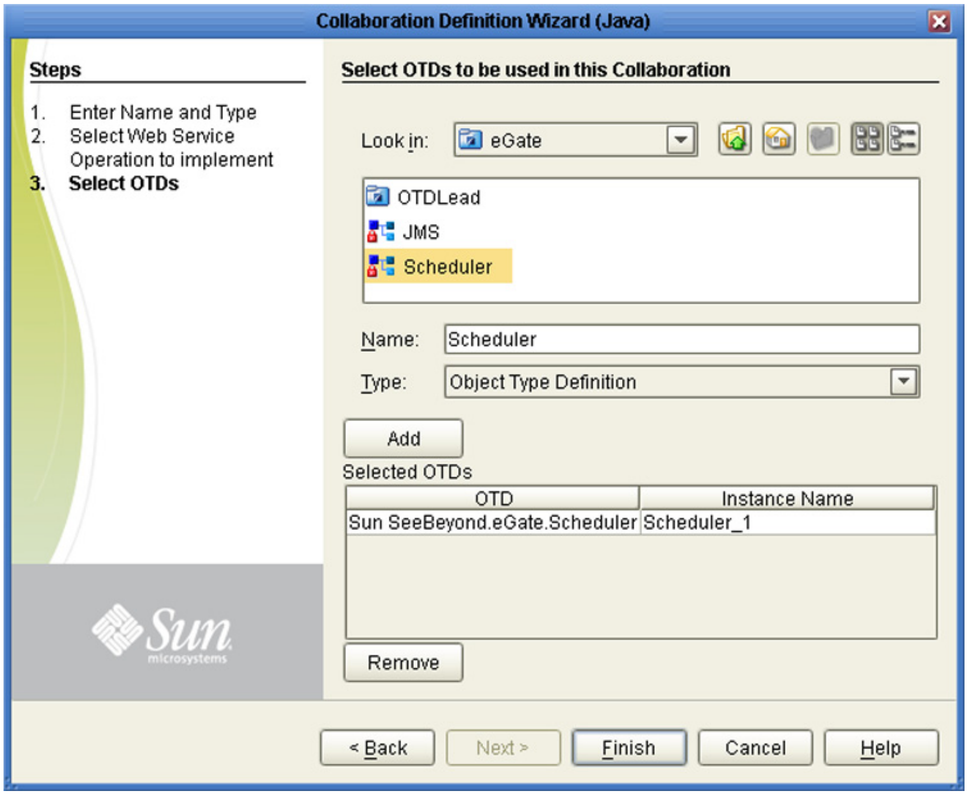
2 Click **Next** to proceed to the next Wizard dialog.

3 Select an auxiliary OTD, if necessary (see Figure 273). This step is optional, and is intended to support additional functionality. For example:

- For Collaborations that read from or write to a topic or queue, you need to add the **JMS OTD**.
- If the Collaboration is triggered by a Scheduler adapter, you need to add the **Scheduler OTD** (shown as an example in Figure 273).
- For Collaborations that invoke a web service, you must add the appropriate **WSD Object** (see **Invoking Web Services From Collaboration Definitions** on page 513).

Note: *You may already have selected OTDs in the preceding steps — if so, you should not repeat them here.*

Figure 273 Existing Web Service: Select OTD



Collaboration Definition Wizard (Java)

Steps

1. Enter Name and Type
2. Select Web Service Operation to implement
3. **Select OTDs**

Select OTDs to be used in this Collaboration

Look in: eGate

OTDLead
JMS
Scheduler

Name: Scheduler

Type: Object Type Definition

Selected OTDs

OTD	Instance Name
Sun SeeBeyond.eGate.Scheduler	Scheduler_1

4 Click **Add** to add the OTD to the Collaboration Definition.

5 Click **Next** to proceed to the Collaboration Editor (Java).

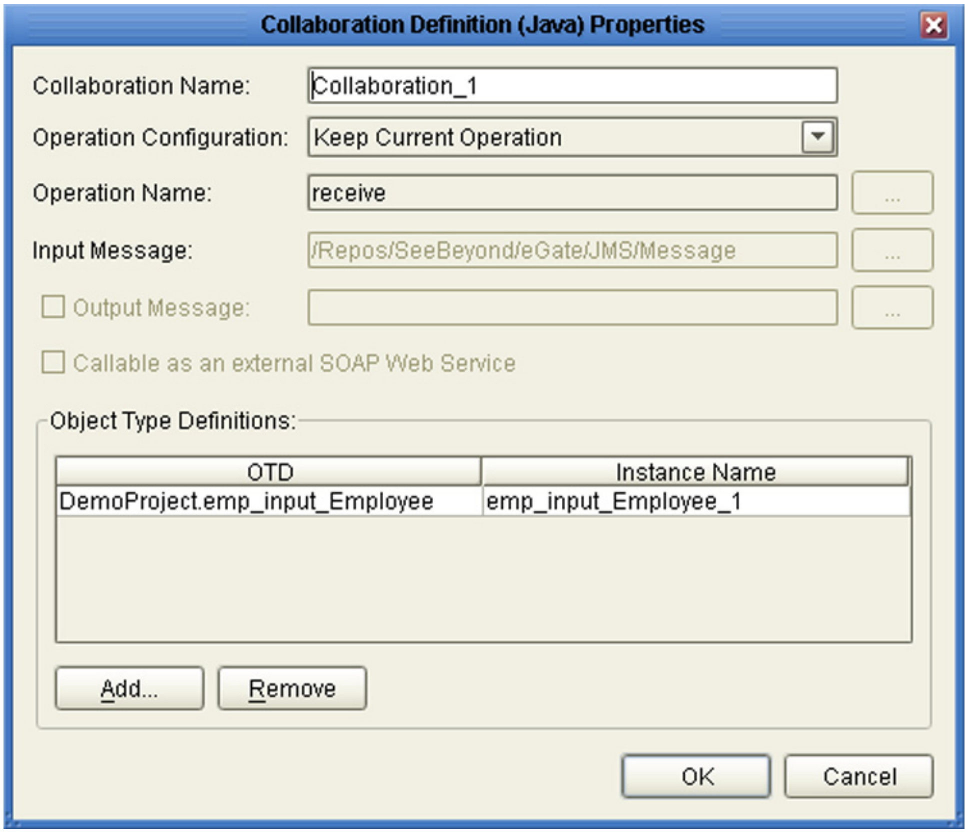
10.2 Editing Collaboration Definition Properties

Right-clicking a Collaboration Definition (Java) in the Project Explorer displays the context menu shown in **Using the Collaboration Definition Context Menu** on page 254. Selecting **Properties** from the menu presents the *Collaboration Definition (Java) Properties* dialog for the selected Collaboration Definition (see Figure 274).

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 274 Collaboration Definition (Java) Properties



Collaboration Definition (Java) Properties

Collaboration Name: Collaboration_1

Operation Configuration: Keep Current Operation

Operation Name: receive

Input Message: /Repos/SeeBeyond/eGate/JMS/Message

□ Output Message:

□ Callable as an external SOAP Web Service

Object Type Definitions:

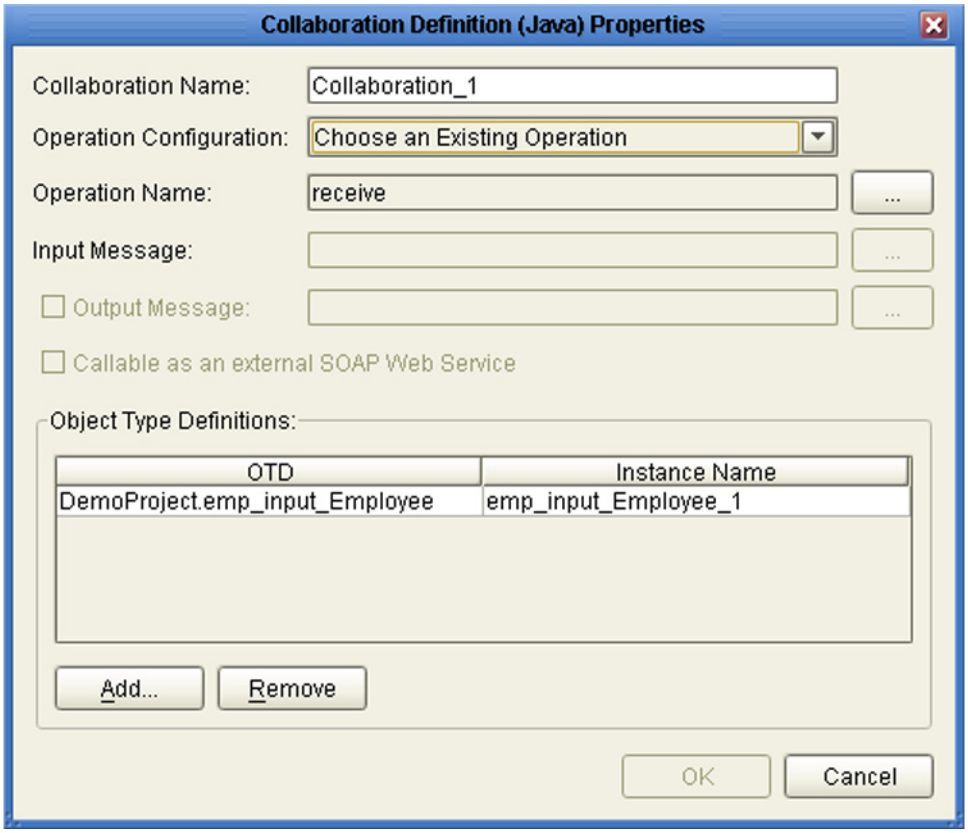
OTD	Instance Name
DemoProject.emp_input_Employee	emp_input_Employee_1

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 254).
3. Select **Choose an Existing Operation** from the Operation Configuration list (see Figure 275).

Figure 275 Collaboration Definition (Java) Properties



Collaboration Definition (Java) Properties

Collaboration Name: Collaboration_1

Operation Configuration: Choose an Existing Operation

Operation Name: receive

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. Select the same web service operation as before in **Operation Name** (click the ellipsis [**...**] button, as shown in Figure 275, 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.

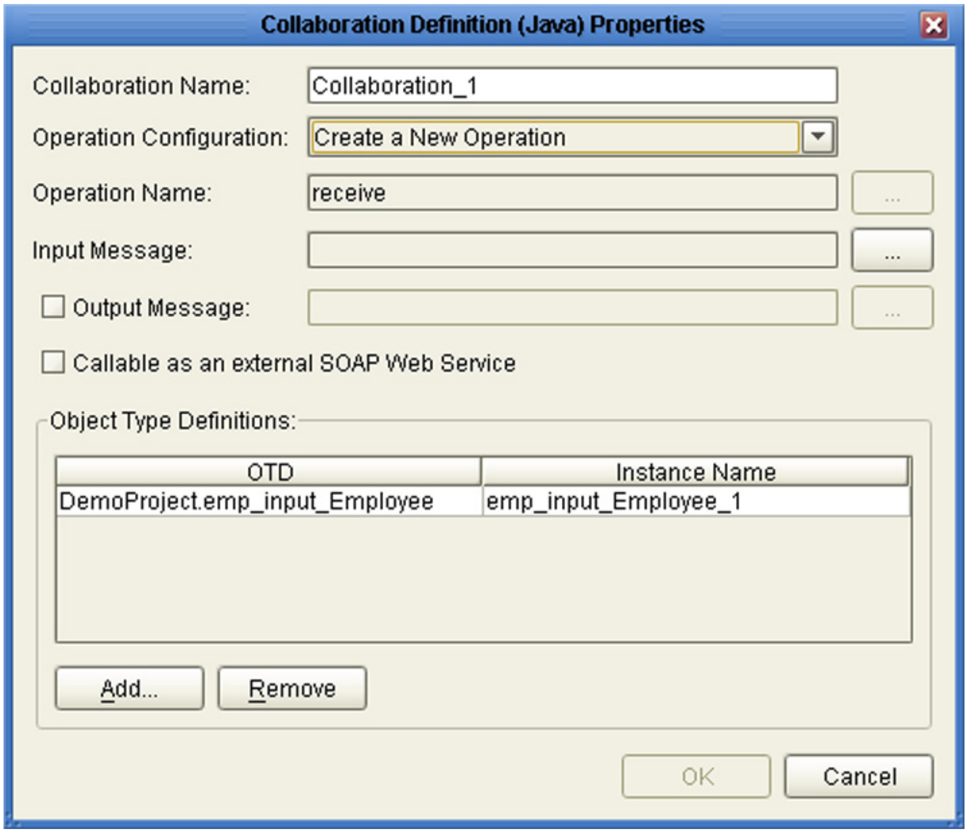
To set the configuration properties for a new operation

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 254).

3 Select **Create a New Operation** from the Operation Configuration list (see Figure 276).

Figure 276 Collaboration Definition (Java) Properties



Collaboration Definition (Java) Properties

Collaboration Name: Collaboration_1

Operation Configuration: Create a New Operation

Operation Name: receive

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 276, 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.

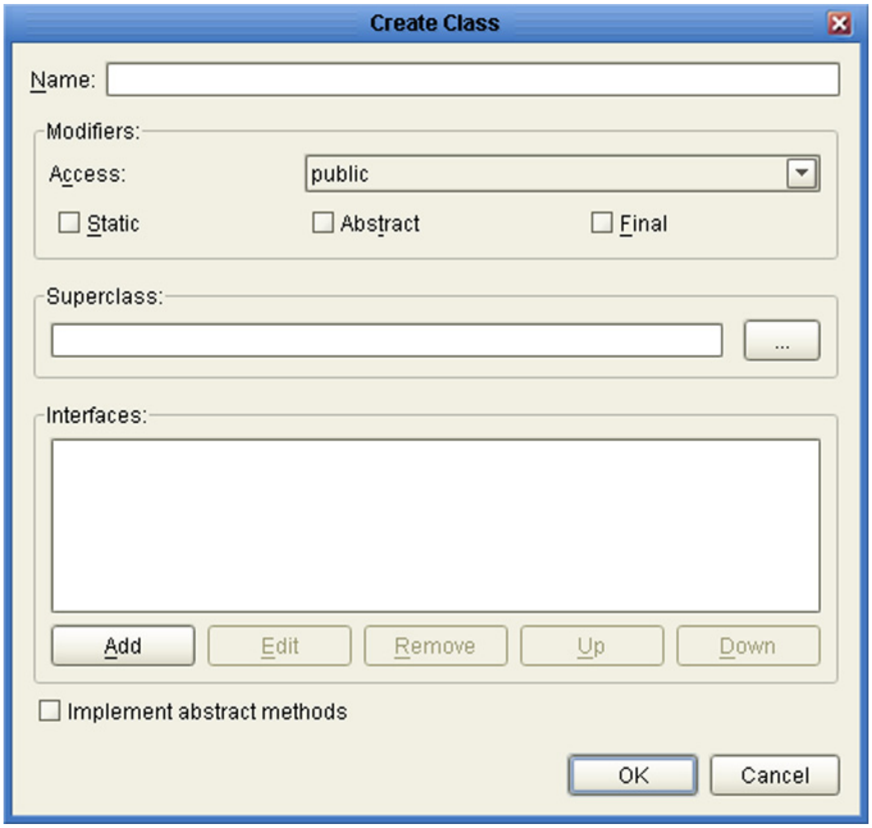
8 Click **OK** to finish.

10.3 Creating Classes

Clicking the **Class** icon in the Business Rules Editor toolbar (see Figure 277) presents the *Create Class* dialog (see Figure 286).

Figure 277 Class Icon

Figure 278 Create Class Dialog



Create Class

Name:

Modifiers:

Access: public

□ Static □ Abstract □ Final

Superclass:

Interfaces:

□ Implement abstract methods

The various options are summarized in Table 76.

Table 76 Class Options

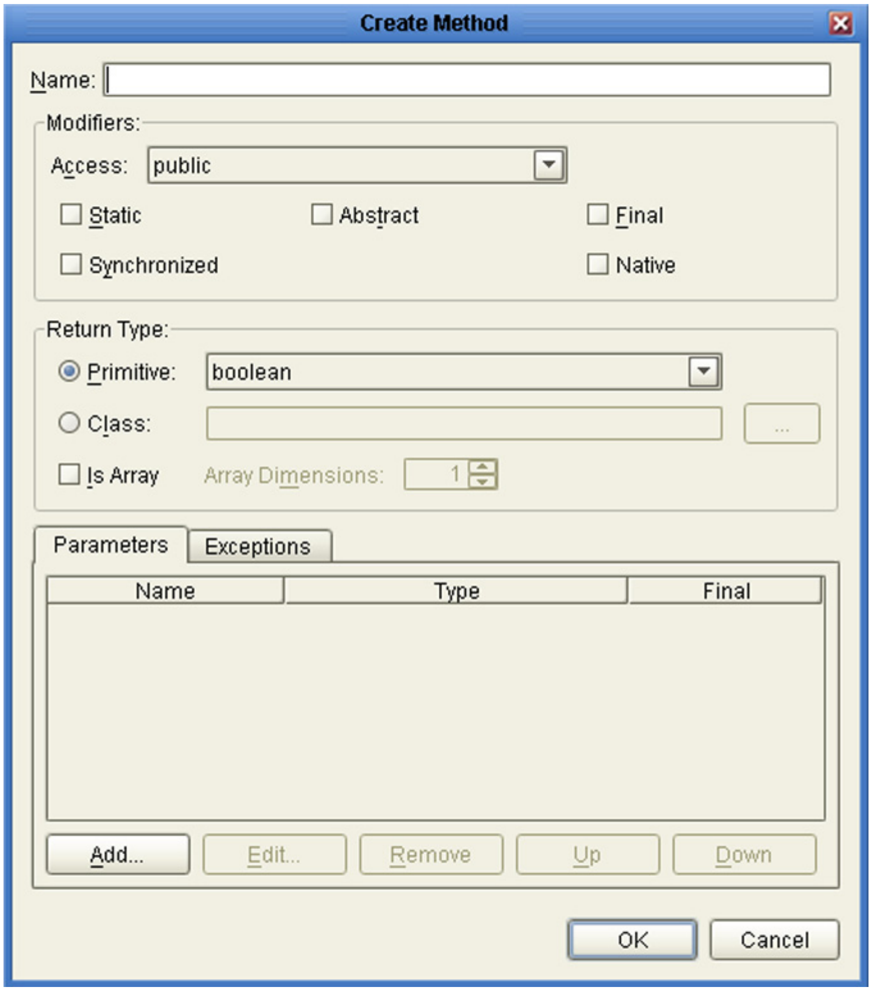
Heading	Option/Command	Description
Name		Your name for the class.
Modifiers	Access	One of the following: - public - protected - private
	Static	Specifies that the class is static.
	Abstract	Specifies that the class is abstract.
	Final	Specifies that the class is final (no subclasses can be created).
Superclass		Assigns the class to another Java class. Clicking the ellipsis (**...**) button launches the *Class Browser* (see **Using the Class Browser** on page 292).
Interfaces		Displays a list of interfaces for the method.
	Add	Invokes the *Class Browser* with which you can locate an interface to add (see **Using the Class Browser** on page 292).
	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 279) presents the *Create Method* dialog (see Figure 280).

Figure 279 Method Icon

Figure 280 Create Method Dialog



Create Method

Name:

Modifiers:

Access: public

☐ Static ☐ Abstract ☐ Final

☐ Synchronized ☐ Native

Return Type:

● Primitive: boolean

○ Class:

☐ Is Array Array Dimensions: 1

Name	Type	Final

The various options are summarized in Table 77.

Table 77 Method Options

Heading	Option/Command	Description
Name		Your name for the method.
Modifiers	Access	One of the following: - public - protected - private
	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: - boolean - byte - char - short - int - long - float - double
	Class	Assigns the method to a Java class. Clicking the ellipsis (**...**) button launches the *Class Browser* (see **Using the Class Browser** on page 292).
	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 *Create Parameter* dialog, in which you can define a parameter (see **Creating Parameters** on page 341).
	Edit	Presents the *Parameter* dialog (identical to the *Create Parameter* 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.

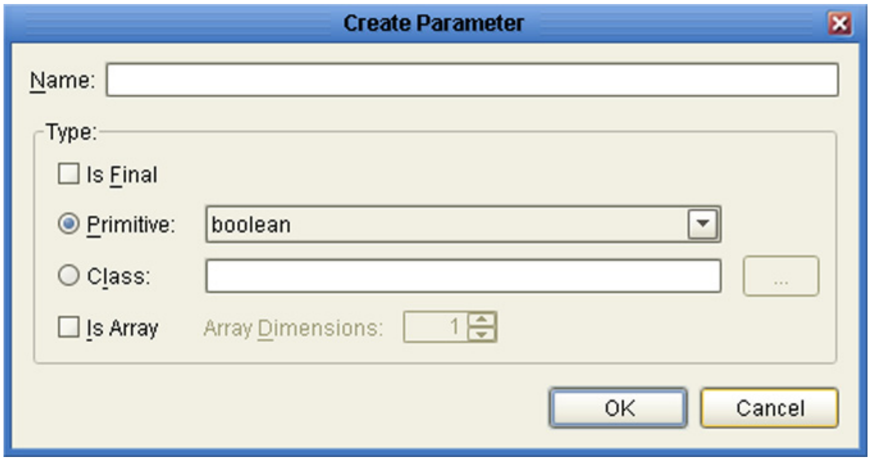
Table 77 Method Options

Heading	Option/Command	Description
Exceptions		Displays a list of exception classes thrown by the method.
	Add	Invokes the *Class Browser,* with which you can locate an exception class (see **Using the Class Browser** on page 292).
	Edit	Invokes the *Class Browser,* 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 281.

Figure 281 Create Parameter Dialog



Create Parameter

Name:

Type:

□ Is Final

● Primitive: boolean

○ Class:

□ Is Array   Array Dimensions: 1

The various options are summarized in Table 78.

Table 78 Parameter Options

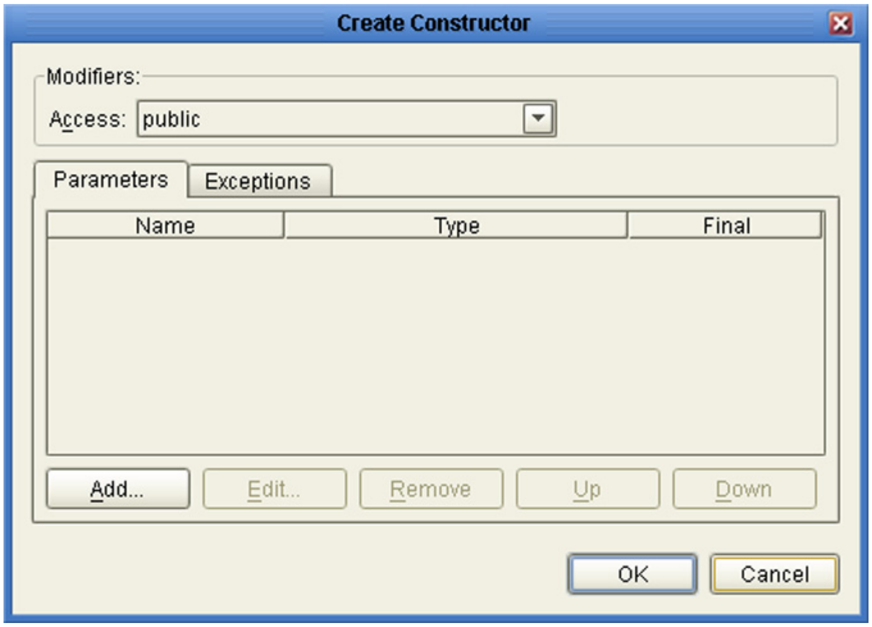
Heading	Option	Description
Name		Your name for the parameter.
Type	Is Final	Specifies that the parameter is final.
	Primitive	One of the following: ▪ boolean ▪ byte ▪ char ▪ short ▪ int ▪ long ▪ float ▪ double
	Class	Assigns the parameter to a Java class. Clicking the ellipsis (**...**) button launches the *Class Browser* (see **Using the Class Browser** on page 292).
	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 282) presents the *Create Constructor* dialog (see Figure 286).

Figure 282 Constructor Icon

Figure 283 Create Constructor Dialog



Access: public

The various options are summarized in Table 79.

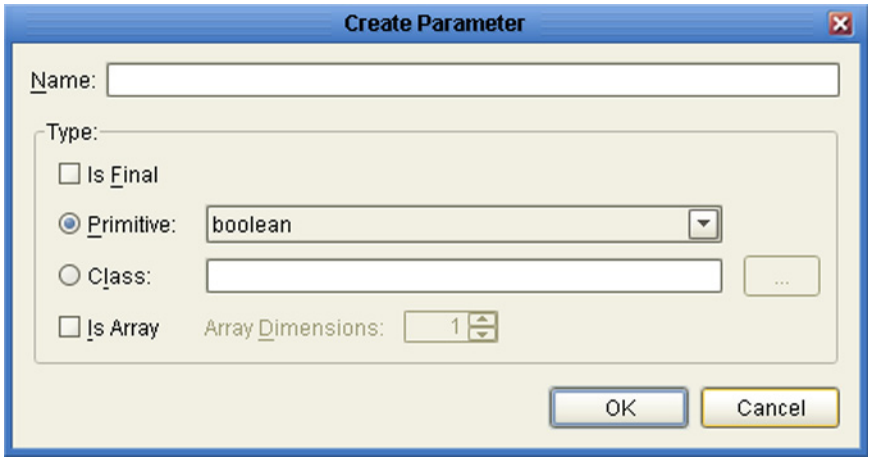
Table 79 Constructor Options

Heading	Option/Command	Description
Modifiers	Access	One of the following: ▪ public ▪ protected ▪ private
Parameters		Displays a list of parameters for the method.
	Add	Presents the *Create Parameter* dialog, in which you can define a parameter (see **Creating Parameters** on page 345).
	Edit	Presents the *Parameter* dialog (identical to the *Create Parameter* 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 *Class Browser*, with which you can locate an exception class (see **Using the Class Browser** on page 292).
	Edit	Invokes the *Class Browser*, 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 281.

Figure 284 Create Parameter Dialog



□ Is Final

● Primitive: boolean

○ Class:

□ Is Array   Array Dimensions: 1

The various options are summarized in Table 78.

Table 80 Parameter Options

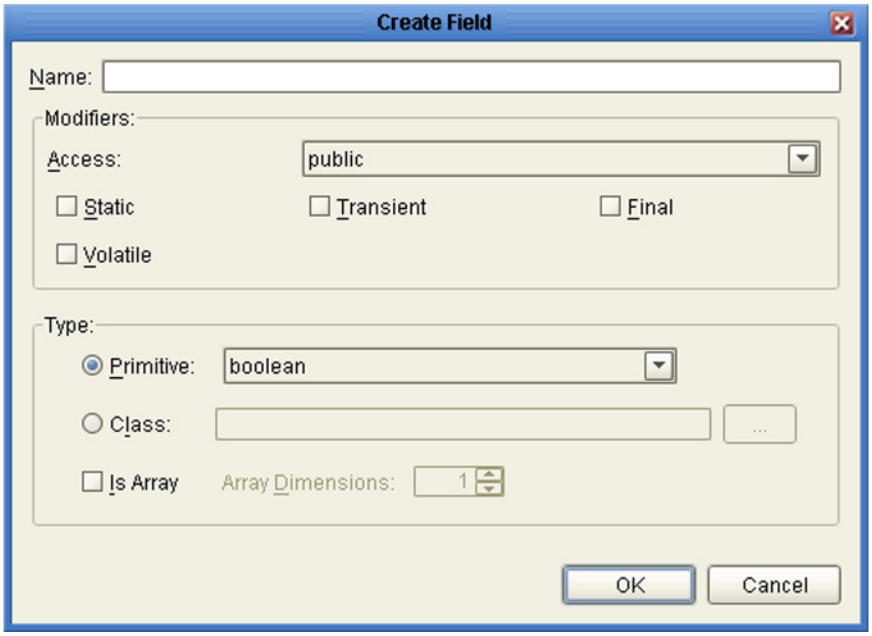
Heading	Option	Description
Name		Your name for the parameter.
Type	Is Final	Specifies that the parameter is final.
	Primitive	One of the following: ▪ boolean ▪ byte ▪ char ▪ short ▪ int ▪ long ▪ float ▪ double
	Class	Assigns the parameter to a Java class. Clicking the ellipsis (**...**) button launches the *Class Browser* (see **Using the Class Browser** on page 292).
	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 285) presents the **Create Field** dialog (see Figure 286).

Figure 285 Field Icon

Figure 286 Create Field Dialog



Create Field

Name:

Modifiers:

Access: public

□ Static □ Transient □ Final

□ Volatile

Type:

● Primitive: boolean

○ Class:

□ Is Array Array Dimensions: 1

The various options are summarized in Table 81.

Table 81 Field Options

Heading	Option	Description
Name		Your name for the field.
Modifiers	Access	One of the following: - public - protected - private
	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: - boolean - byte - char - short - int - long - float - double
	Class	Assigns the field to a Java class. Clicking the ellipsis (**...**) button launches the *Class Browser* (see **Using the Class Browser** on page 292).
	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 287) presents the **New Array** dialog (see Figure 288).

Figure 287 New Array Icon

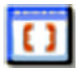
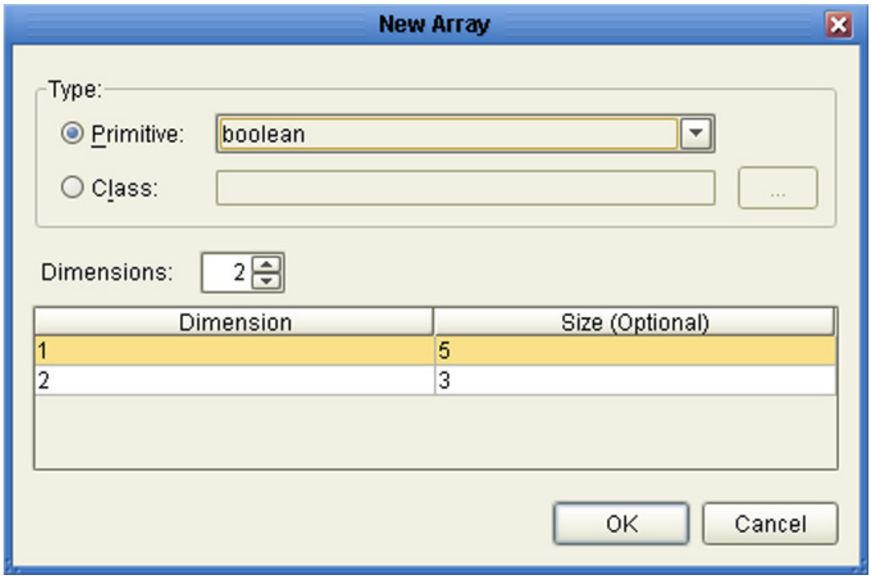


Figure 288 New Array Dialog



New Array

Type:

● Primitive: boolean

○ Class:

Dimensions: 2

Dimension	Size (Optional)
1	5
2	3

The various options are summarized in Table 82.

Table 82 Array Options

Heading	Option	Description
Array Type	Primitive	One of the following: ▪ boolean ▪ byte ▪ char ▪ short ▪ int ▪ long ▪ float ▪ double
	Class	Assigns the field to a Java class. Clicking the ellipsis (**...**) button launches the *Class Browser* (see **Using the Class Browser** on page 292).

Table 82 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.

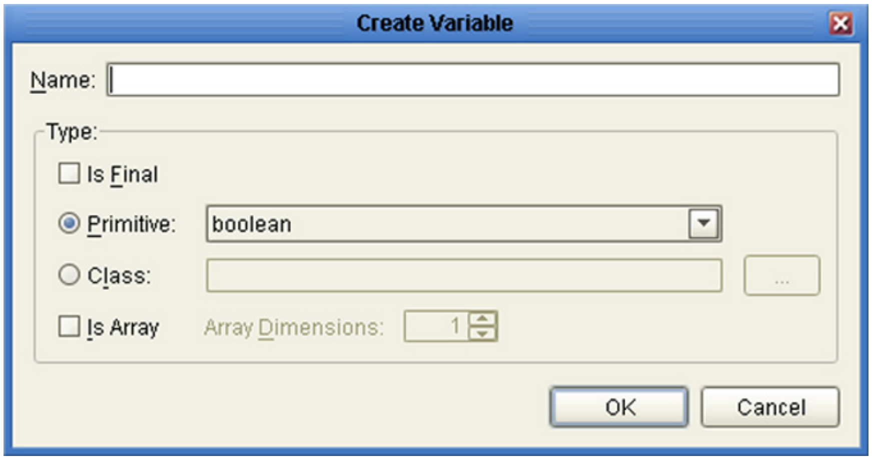
Note: *If you do not specify the size, a placeholder () is entered by default. You can then specify the size at a later time.*

10.8 Creating and Modifying Variables

Clicking the **Local Variable** icon in the Business Rules Editor toolbar (see Figure 289) presents the **Create Variable** dialog (see Figure 290).

Figure 289 Local Variable Icon

Figure 290 Create Variable Dialog



Name:

Type:

☐ Is Final

● Primitive: boolean

○ Class:

☐ Is Array Array Dimensions: 1

The various options are summarized in Table 83.

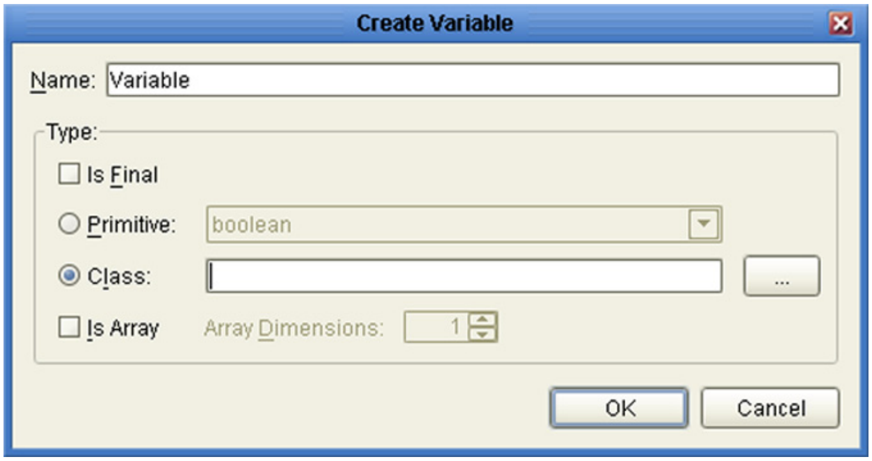
Table 83 Variable Options

Heading	Option	Description
Variable Name		Your name for the variable.
Type	Is Final	Specifies that the variable is final.
	Primitive	One of the following: ▪ boolean ▪ byte ▪ char ▪ short ▪ int ▪ long ▪ float ▪ double
	Class	Assigns the variable to a Java class. Clicking the ellipsis (**...**) button launches the *Class Browser* (see **Using the Class Browser** on page 292).
	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 291.

Figure 291 Create Variable Dialog



Create Variable

Name: Variable

Type:

☐ Is Final

○ Primitive: boolean

● Class:

☐ Is Array

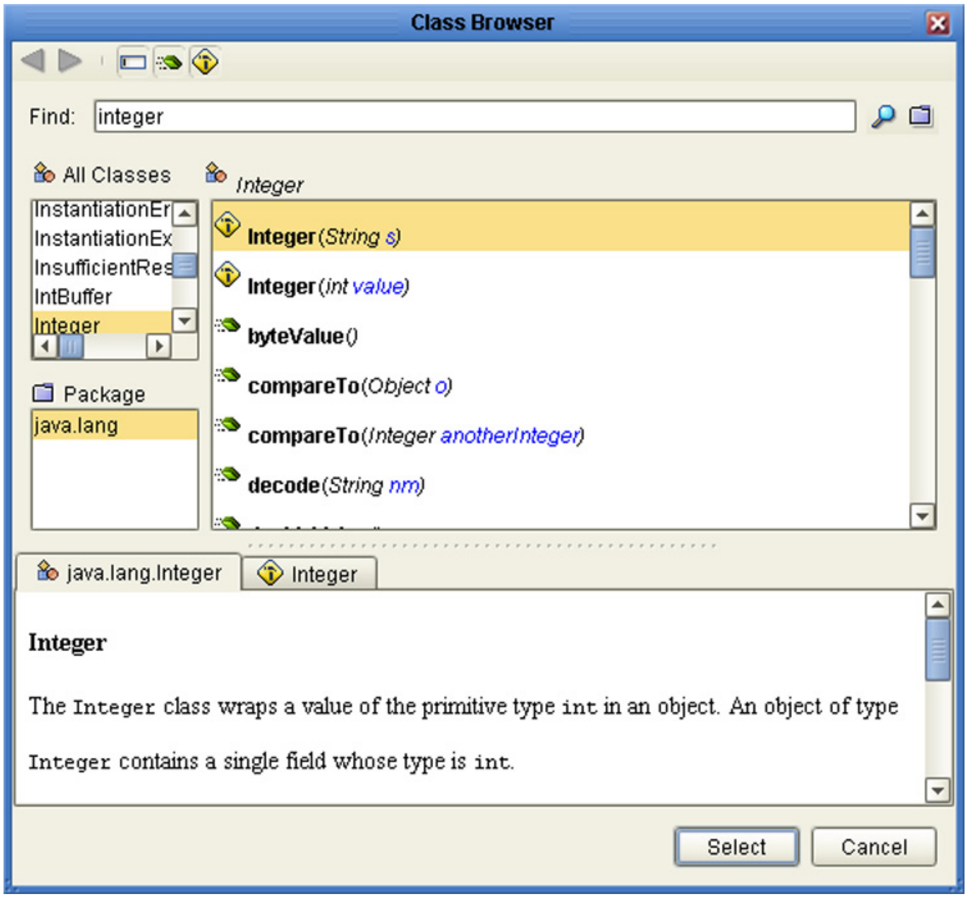
Array Dimensions: 1

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 292.

Figure 292 Class Browser - Select Class

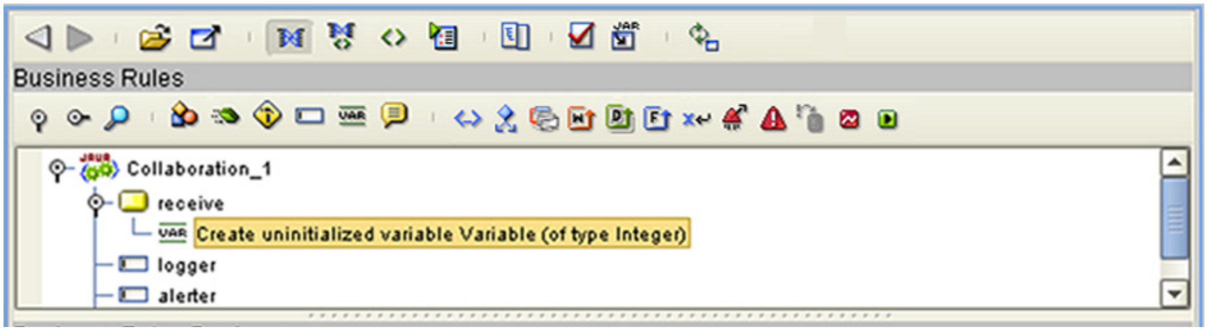


Integer

The `Integer` class wraps a value of the primitive type `int` in an object. An object of type `Integer` contains a single field whose type is `int`.

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 293.

Figure 293 Business Rules with Variable

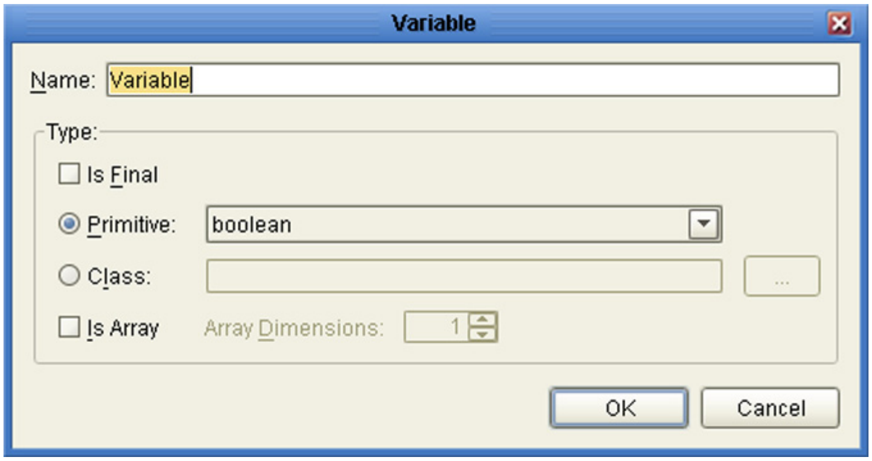


Note: *Having the variable selected when the constructor is created initializes the variable. This also occurs when a literal is created, for those variables that can have literals assigned.*

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 294).

Figure 294 Variable Dialog



Name: Variable

☐ Is Final

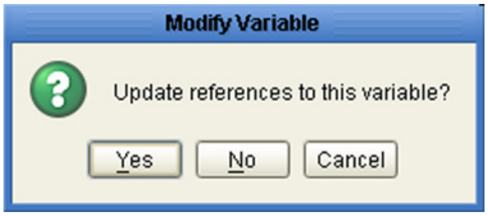
● Primitive: boolean

○ Class:

☐ Is Array Array Dimensions: 1

3. Make the desired changes to the variable properties.
4. Click **OK** to display the dialog shown in Figure 295.

Figure 295 Modify Variable Dialog



Update references to this variable?

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 84, which are described in **Graphical Collaboration Methods (Java)** on page 273. Clicking the menu option places the corresponding method box on the mapping canvas.

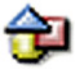
Table 84 Literals in Toolbar Menus

Menu	Literal
Boolean	False
	True
Math	Literal Number
Object	Null
String	Literal Character
	Literal String

Using the Method Dialog

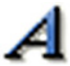
Launch the method dialog as described in **Using the Method Dialog** on page 297 and click the **Create Literal** icon in the toolbar. If you have launched the dialog from an Object, the icon depicts an Object (see Figure 296) and the **Null** method box is placed on the mapping canvas.

Figure 296 Object Icon



If you have launched the icon from a String, the icon depicts a String (seeFigure 297) and the **Literal String** method box is placed on the mapping canvas.

Figure 297 String Icon



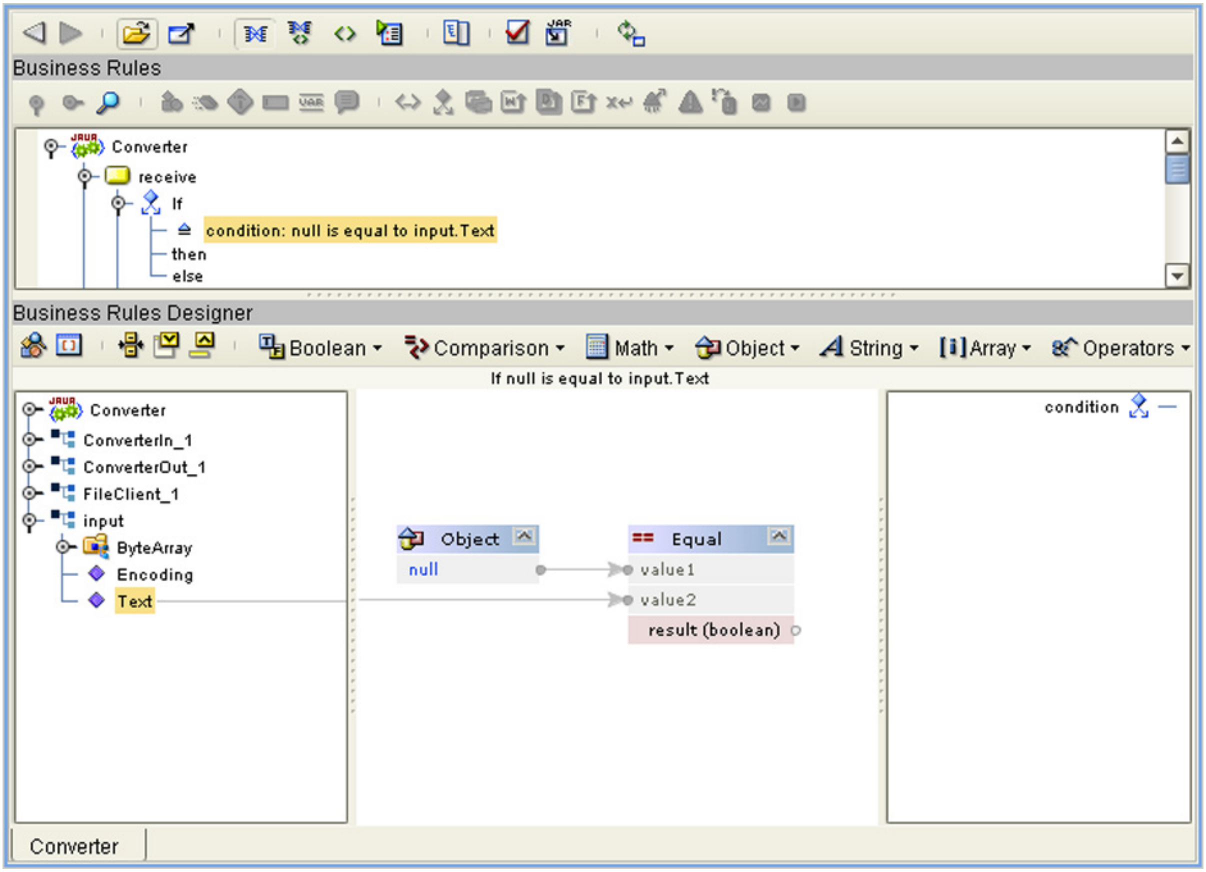
10.10 Generating Alerts

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 298).
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.

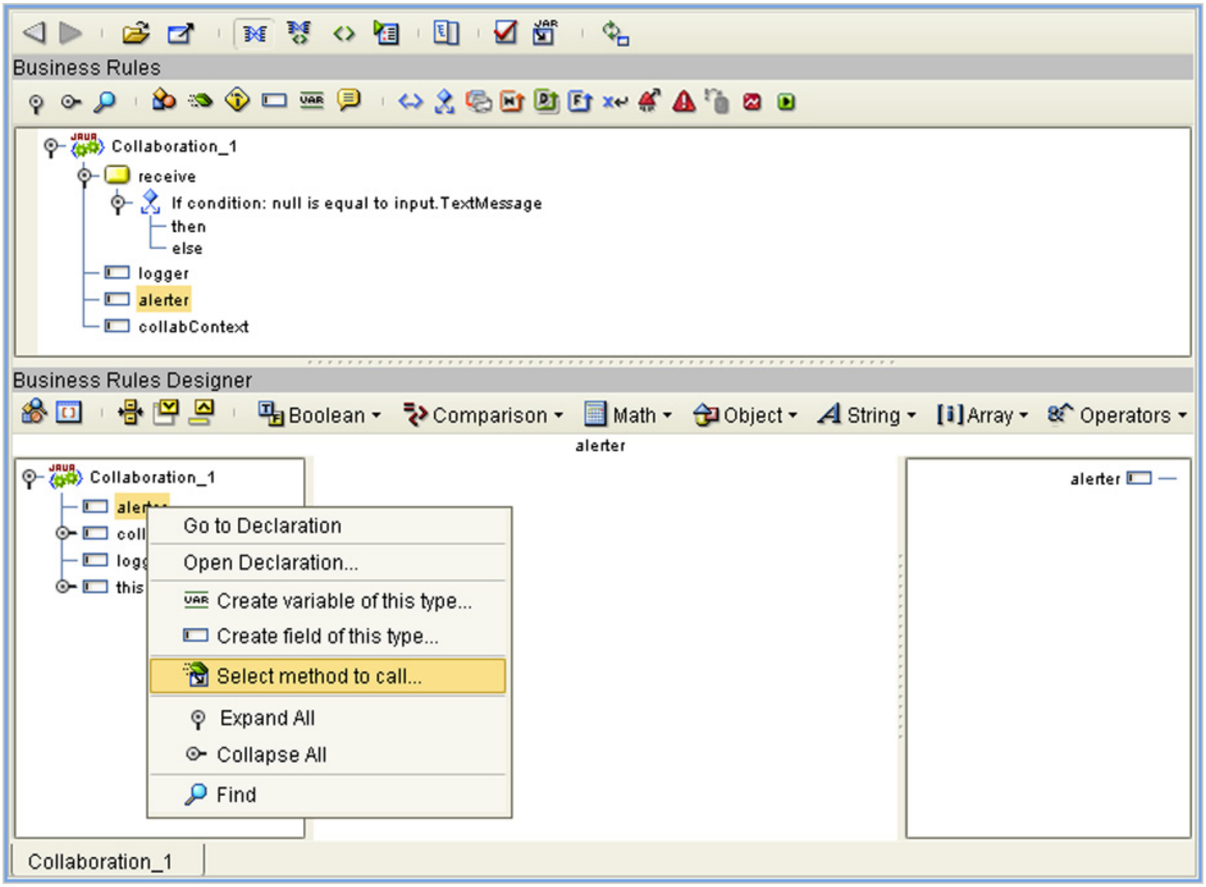
Figure 298 Empty File Test - If Condition



4 Initiate an alert object by clicking the **alerter** node of your Collaboration Definition in the Business Rules editor.

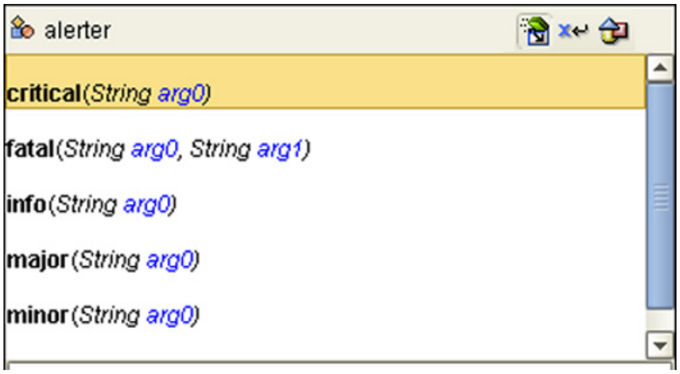
5 Right-click the **alerter** node of the Collaboration Definition in the left pane of the Business Rules Designer, which displays the menu shown in Figure 299.

Figure 299 Empty File Test - Alerter Node Menu



6 Click **Select a method to call**, which displays a method list box containing severity levels (see Figure 300).

Figure 300 Alerter Severity Method List



critical(String arg0)

fatal(String arg0, String arg1)

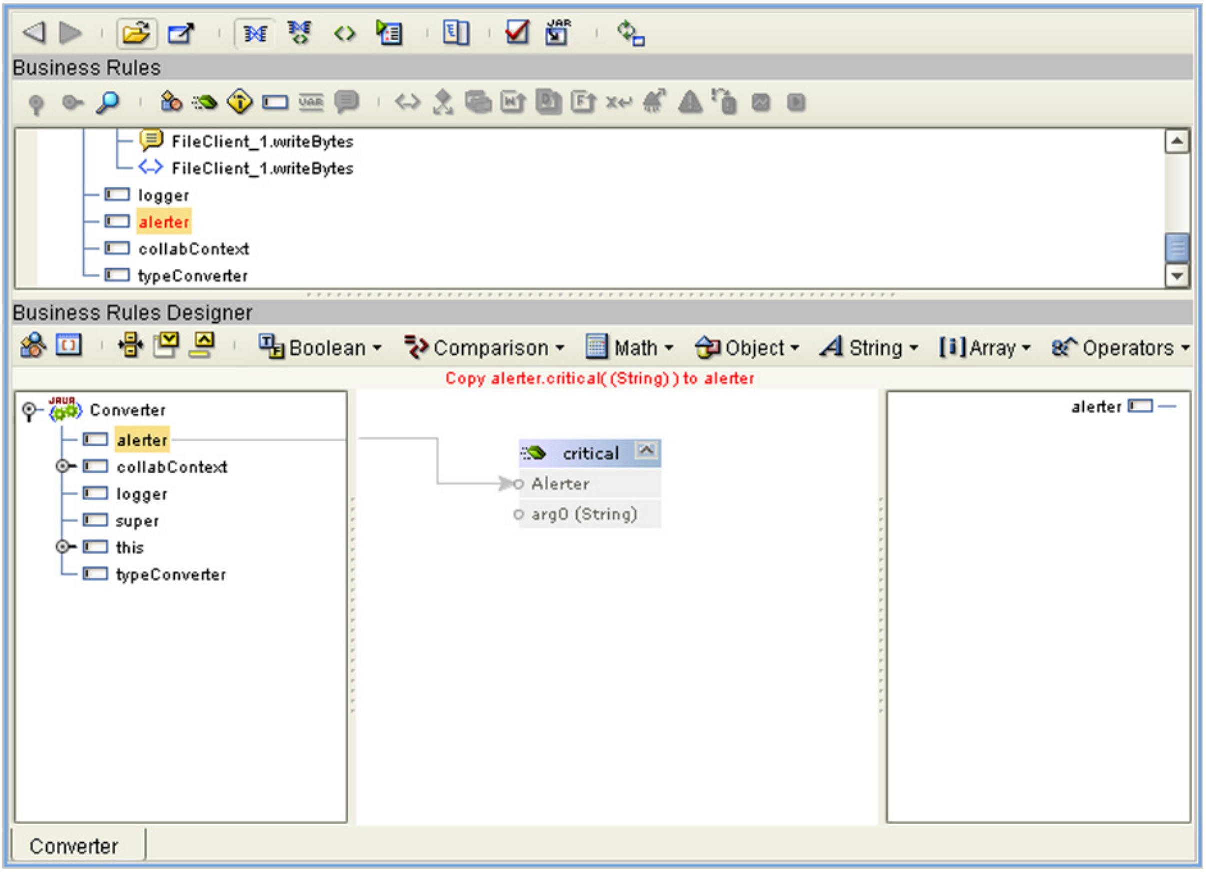
info(String arg0)

major(String arg0)

minor(String arg0)

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 301).

Figure 301 Empty File Test - critical Method

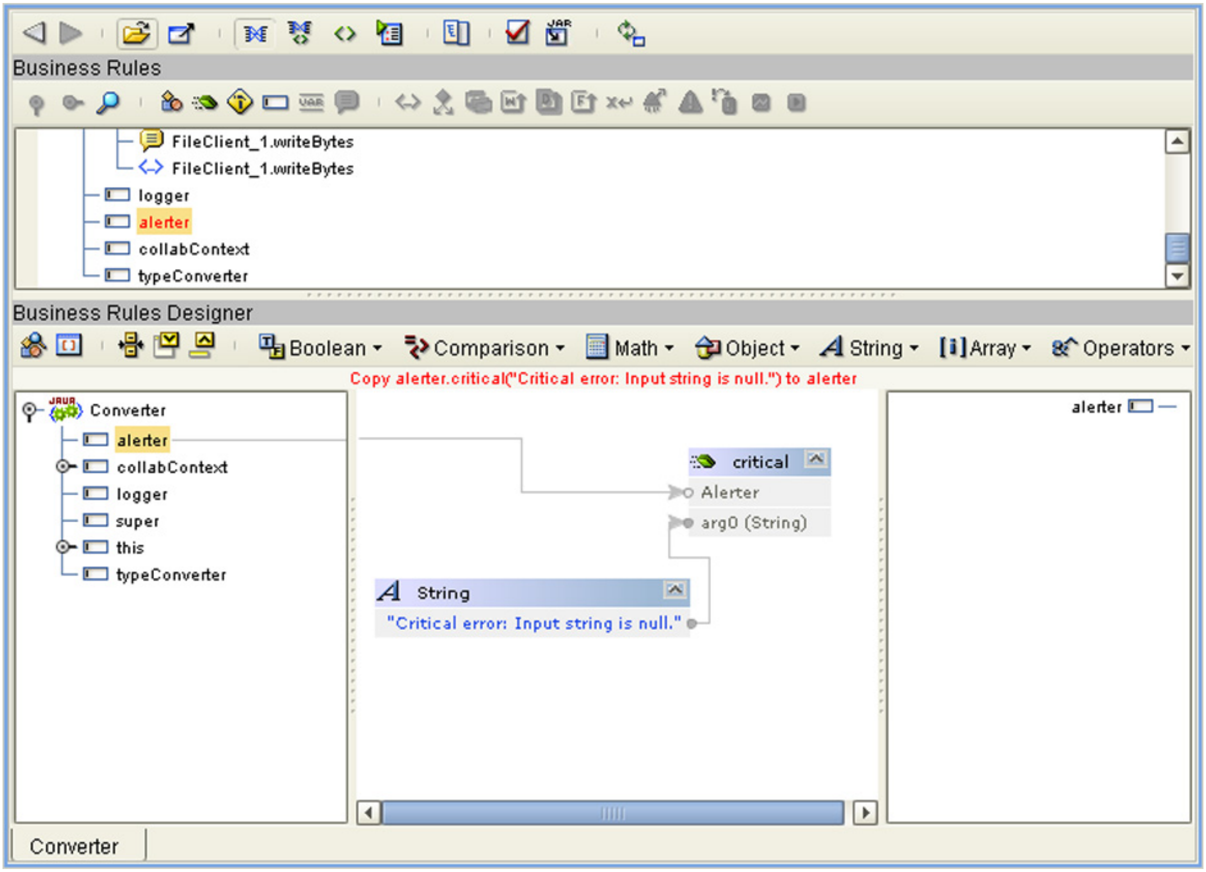


Copy alerter.critical((String)) to alerter

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 302).

Figure 302 Empty File Test - Pass Alert Message



Copy alerter.critical("Critical error: Input string is null.") to alerter

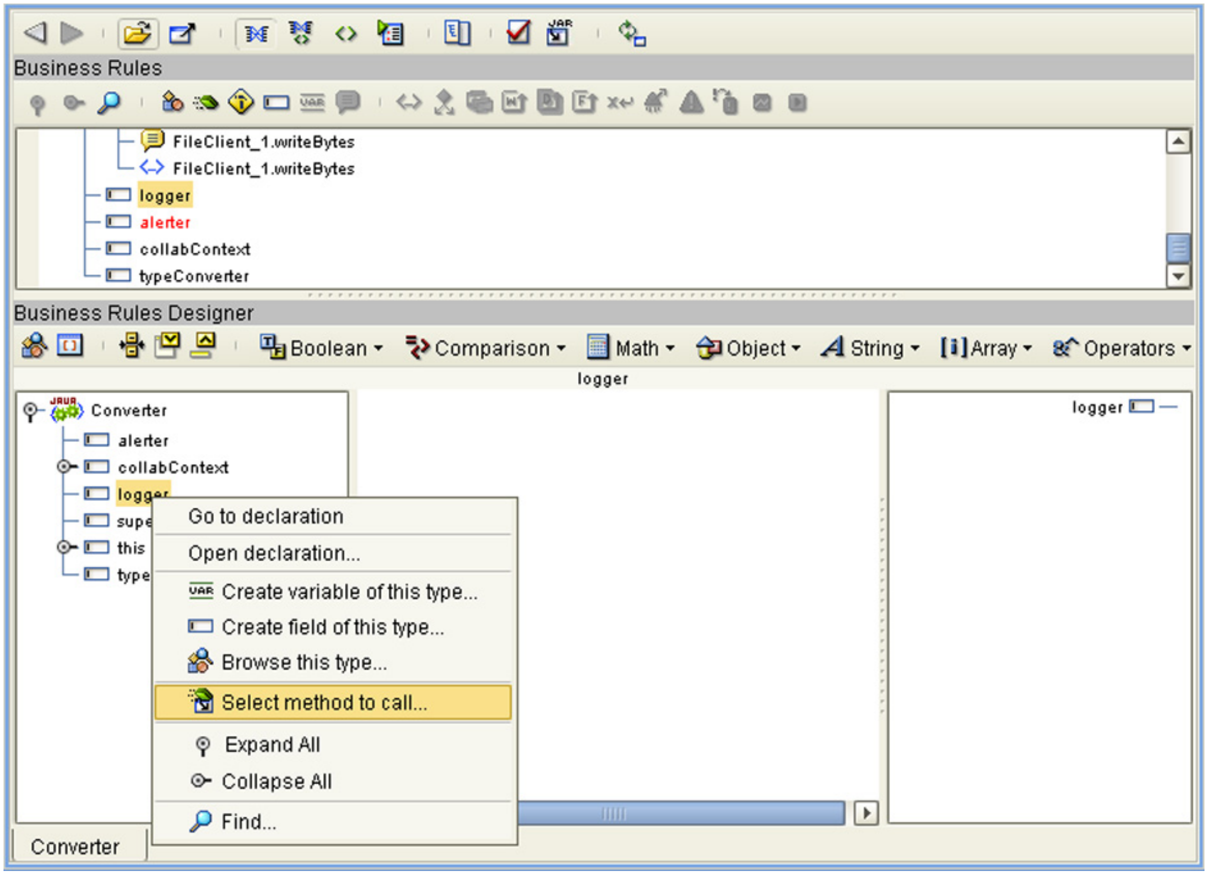
10.11 Creating Log Entries

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)

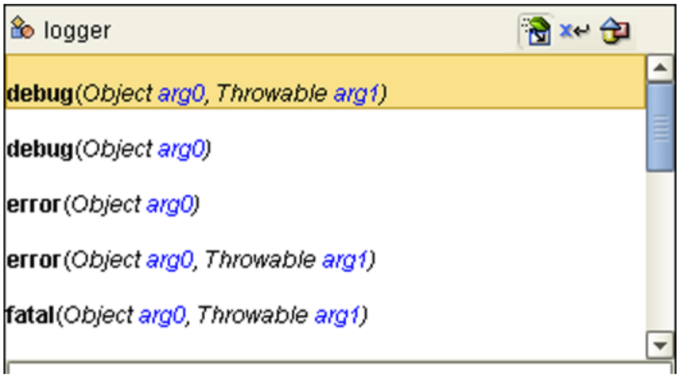
1. Initiate a logging event by clicking the **logger** node of your Collaboration Definition in the Business Rules editor.
2. Right-click the **logger** node for the Collaboration Definition in the left pane of the Business Rules Designer, which displays the menu shown in Figure 303.

Figure 303 Logging Example - Logger Node Menu



3. Click **Select a method to call**, which displays a method list box containing logging levels (see Figure 304).

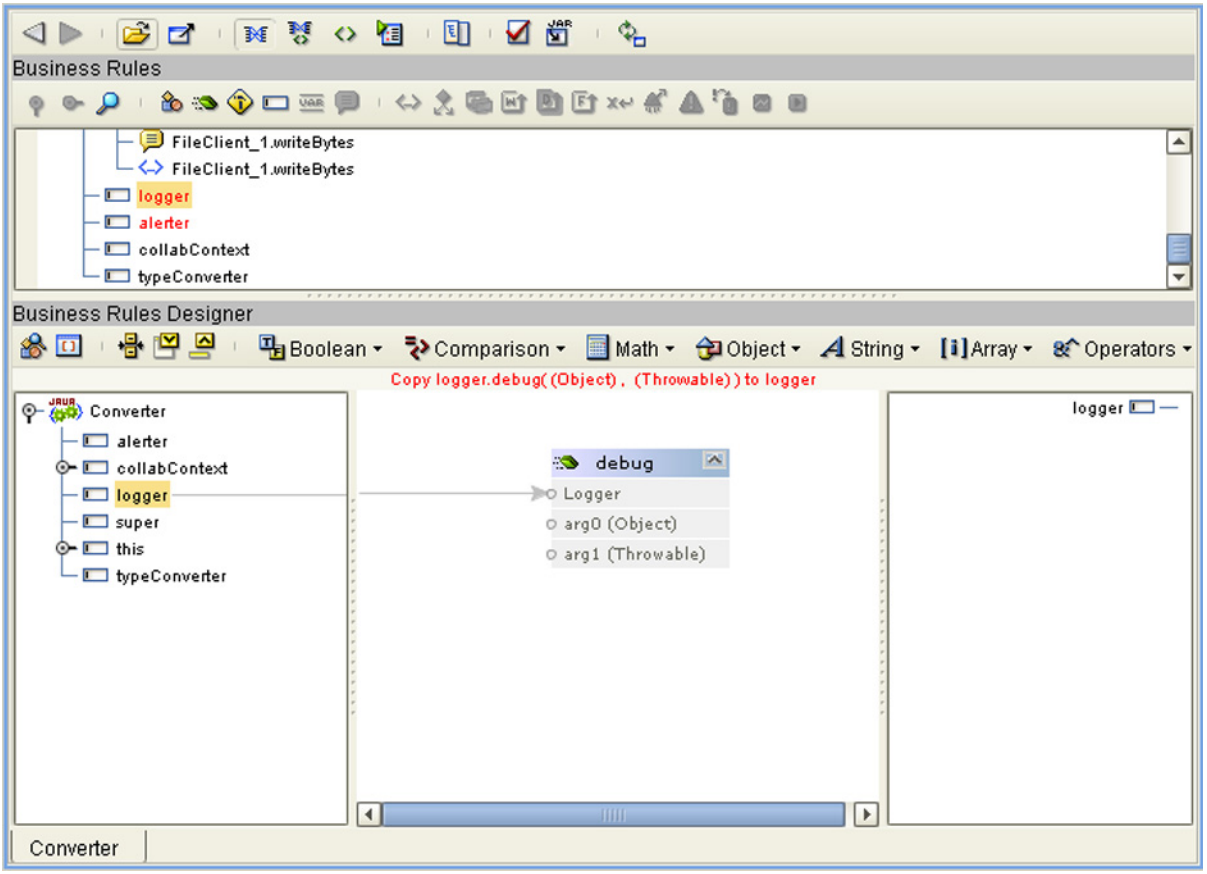
Figure 304 Logging Level Method List



debug(Object arg0, Throwable arg1)
debug(Object arg0)
error(Object arg0)
error(Object arg0, Throwable arg1)
fatal(Object arg0, Throwable arg1)

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 305).

Figure 305 Logging Example - debug Method Box

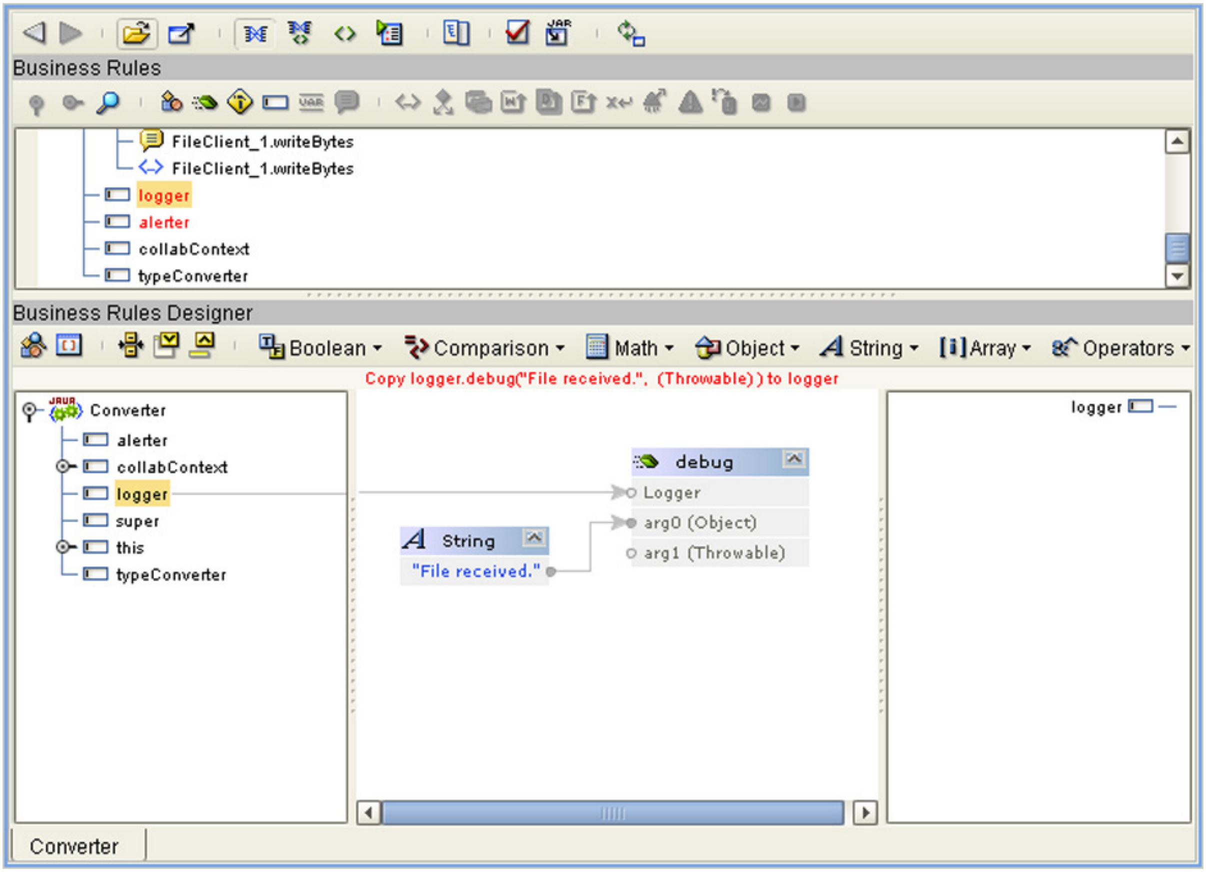


Copy logger.debug((Object) , (Throwable)) to logger

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 306).

Figure 306 Logging Example - Pass Log Message



Copy logger.debug("File received.", (Throwable)) to logger

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 307) to display the file import dialog (see Figure 308).

Figure 307 Project Context Menu: Import File

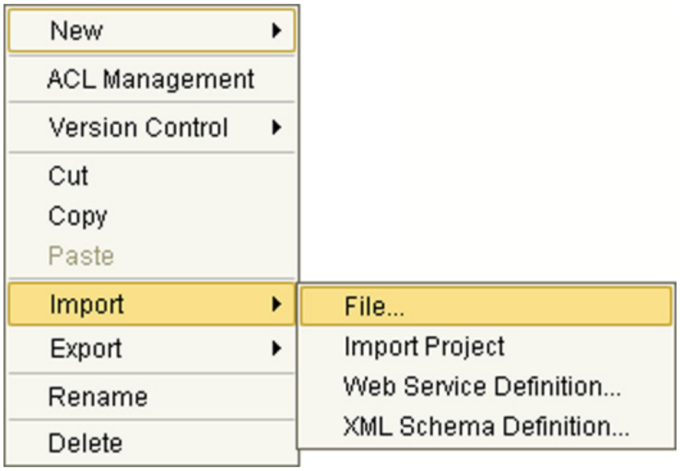
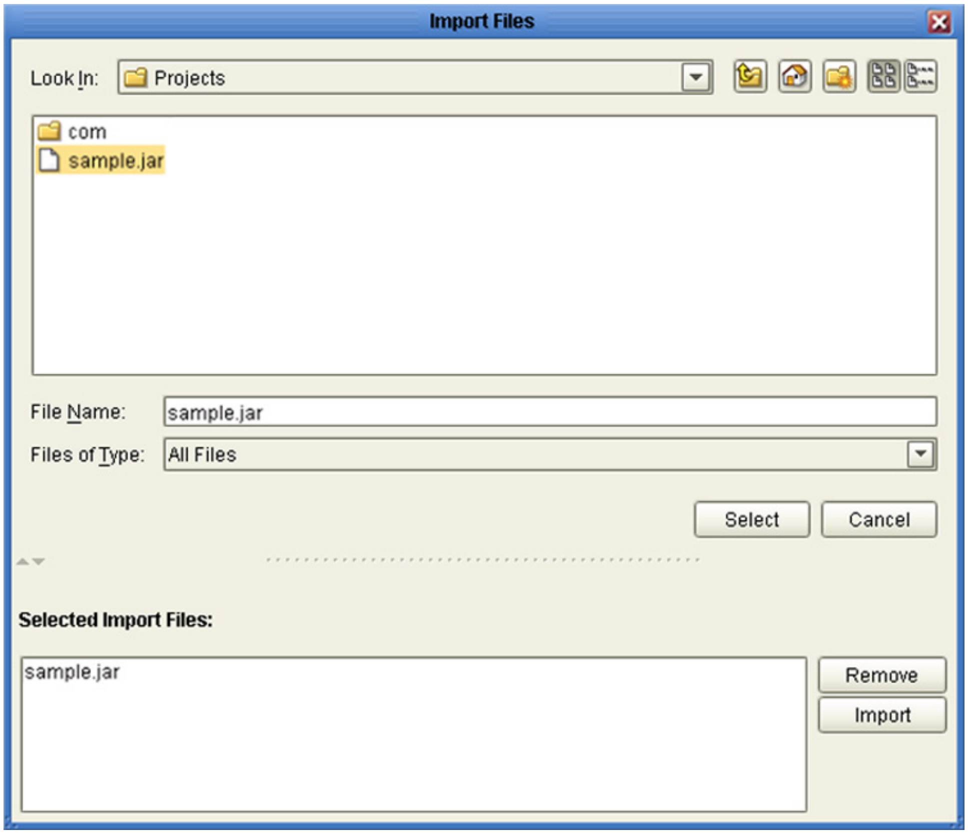


Figure 308 File Import Dialog



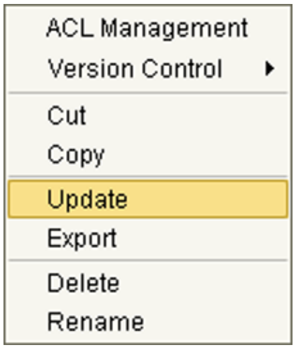
4 Locate the desired **.jar** files and click **Select** for each.

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 364.

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 309).*

Figure 309 File Context Menu - Update



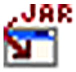
10.13 Adding Java classes to Collaboration Definitions

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:

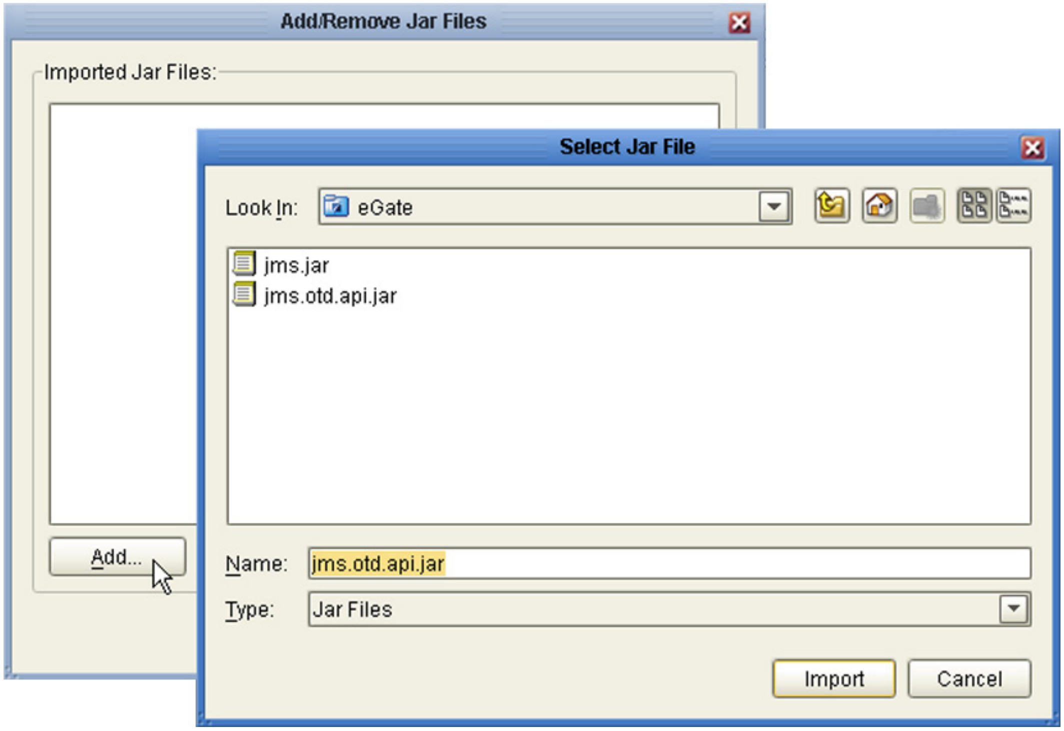
1. Open the Collaboration to which you want to add Java classes.
2. Click the **JAR File** icon in the main Collaboration Editor toolbar (see Figure 310), which presents the *Add/Remove Jar Files* dialog.

Figure 310 JAR File Icon



3. Click **Add** in the dialog to display a file browser (see Figure 311).
4. Locate the desired **.jar** files and click **Import** for each.

Figure 311 Add/Remove Jar File Dialog (1)



Look In: eGate

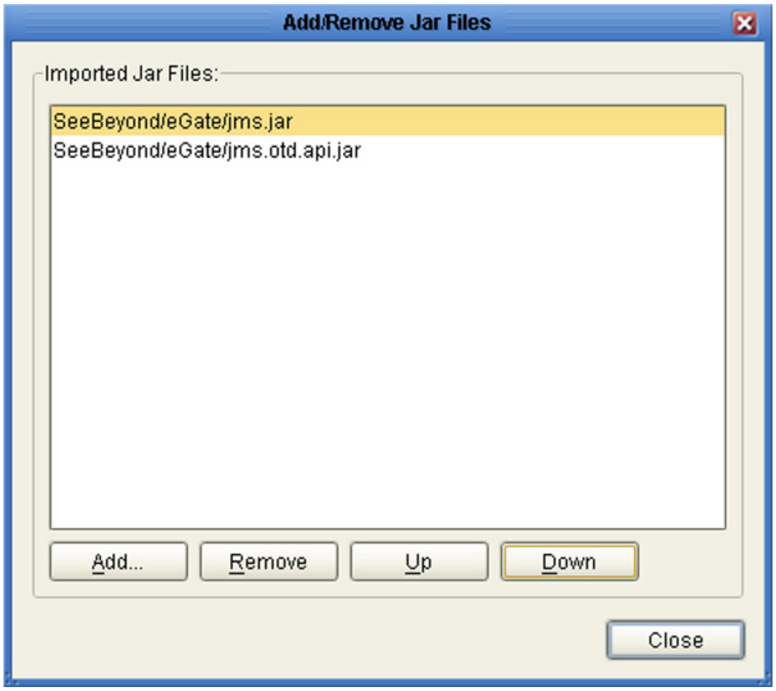
jms.jar
jms.otd.api.jar

Name: jms.otd.api.jar

Type: Jar Files

5 When all the **.jar** files are listed in the *Add/Remove Jar Files* dialog (see Figure 312), click **Close**.

Figure 312 Add/Remove Jar File Dialog (2)



SeeBeyond/eGate/jms.jar
SeeBeyond/eGate/jms.otd.api.jar

- 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 form 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

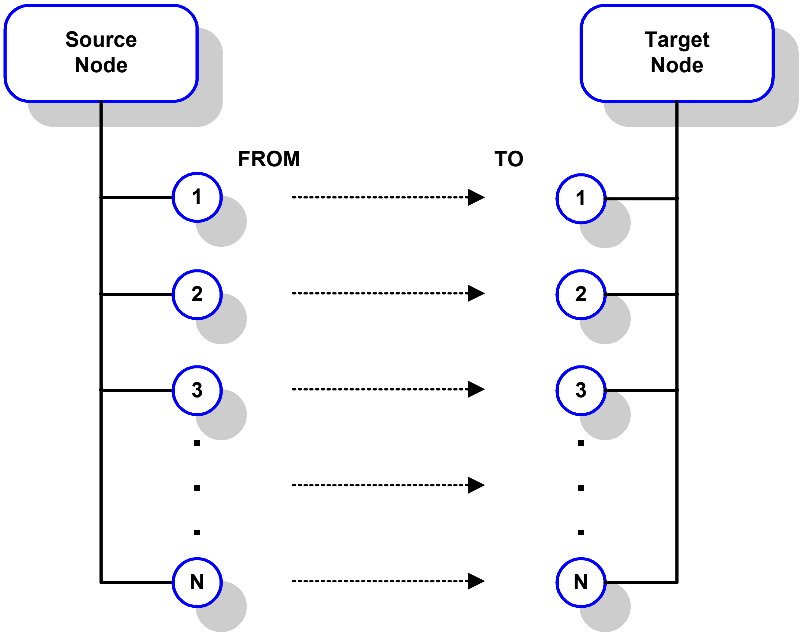
```
public static void copyNode (Object src, Object trg)
```

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 313 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 313 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 JavaType 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 364. 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 314), which adds the **try** statement to the business rules tree (see Figure 315). It also enables the **Catch** icon in the Business Rules editor toolbar.

Figure 314 Try Icon

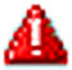
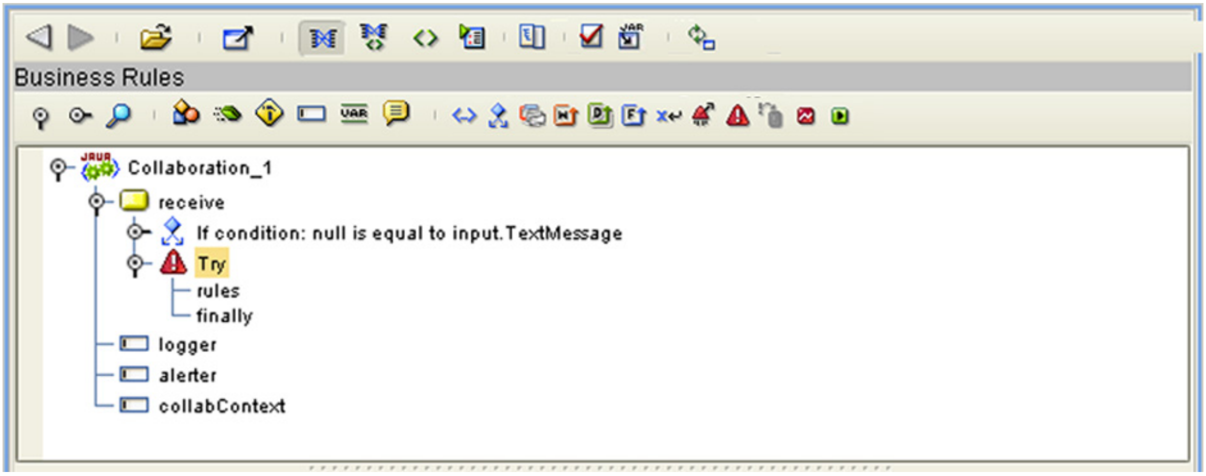


Figure 315 Try Statement



2. Select the **Catch** icon (see Figure 316), which presents the **Create Exception Variable** dialog (see Figure 317).

Figure 316 Catch Icon

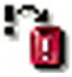
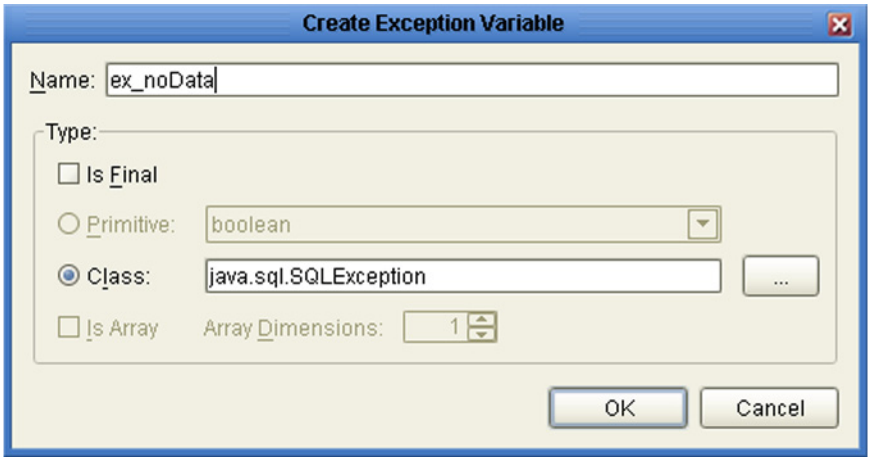


Figure 317 Create Exception Variable Dialog



Create Exception Variable

Name: ex_noData

Type:

☐ Is Final

○ Primitive: boolean

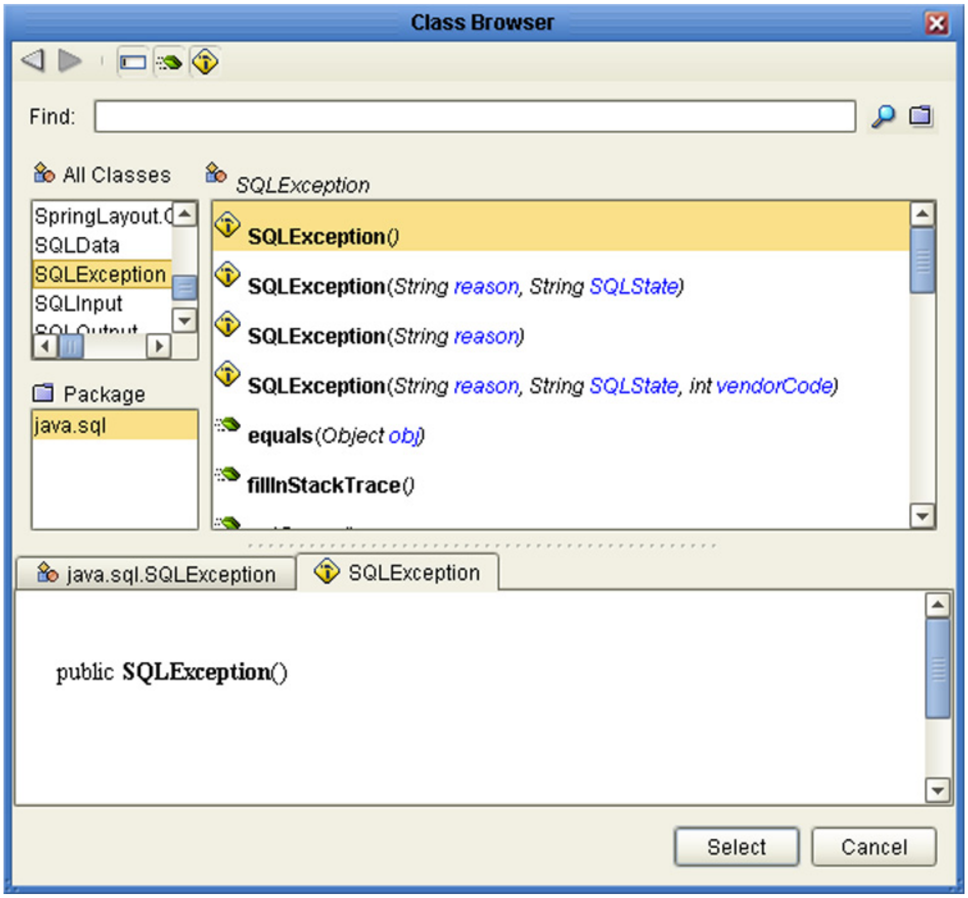
● Class: java.sql.SQLException

☐ Is Array Array Dimensions: 1

3 Enter a name for the exception, for example: **ex_noData**.

4 Click the ellipsis (**…**) button to invoke the *Class Browser* shown in Figure 318.

Figure 318 Class Browser Dialog



Class Browser

Find:

All Classes: SpringLayout.C, SQLData, SQLException, SQLInput, SQLOutput

Package: java.sql

SQLException

- **SQLException**()
- **SQLException**(String reason, String SQLState)
- **SQLException**(String reason)
- **SQLException**(String reason, String SQLState, int vendorCode)
- **equals**(Object obj)
- **fillInStackTrace**()

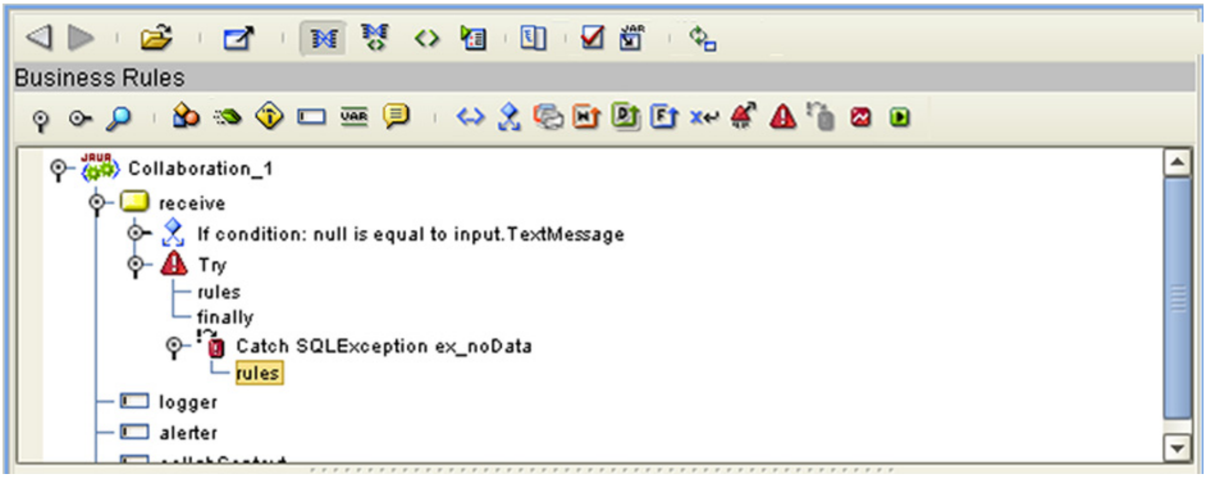
java.sql.SQLException | SQLException

public **SQLException**()

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 319).

Figure 319 Catch Statement


Business Rules
Collaboration_1
receive
If condition: null is equal to input.TextMessage
Try
rules
finally
Catch SQLException ex_noData
rules
logger
alerter

7 The rule shown in Figure 319 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 55.

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 85 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 86 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 320), which displays the dialog shown in Figure 321. The options in the dialog are described in Table 87.

Note: *This icon is displayed only in the Japanese and Korean versions of eGate Integrator.*

Figure 320 Encoding Converter Icon

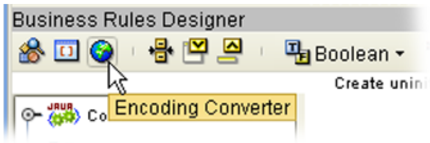
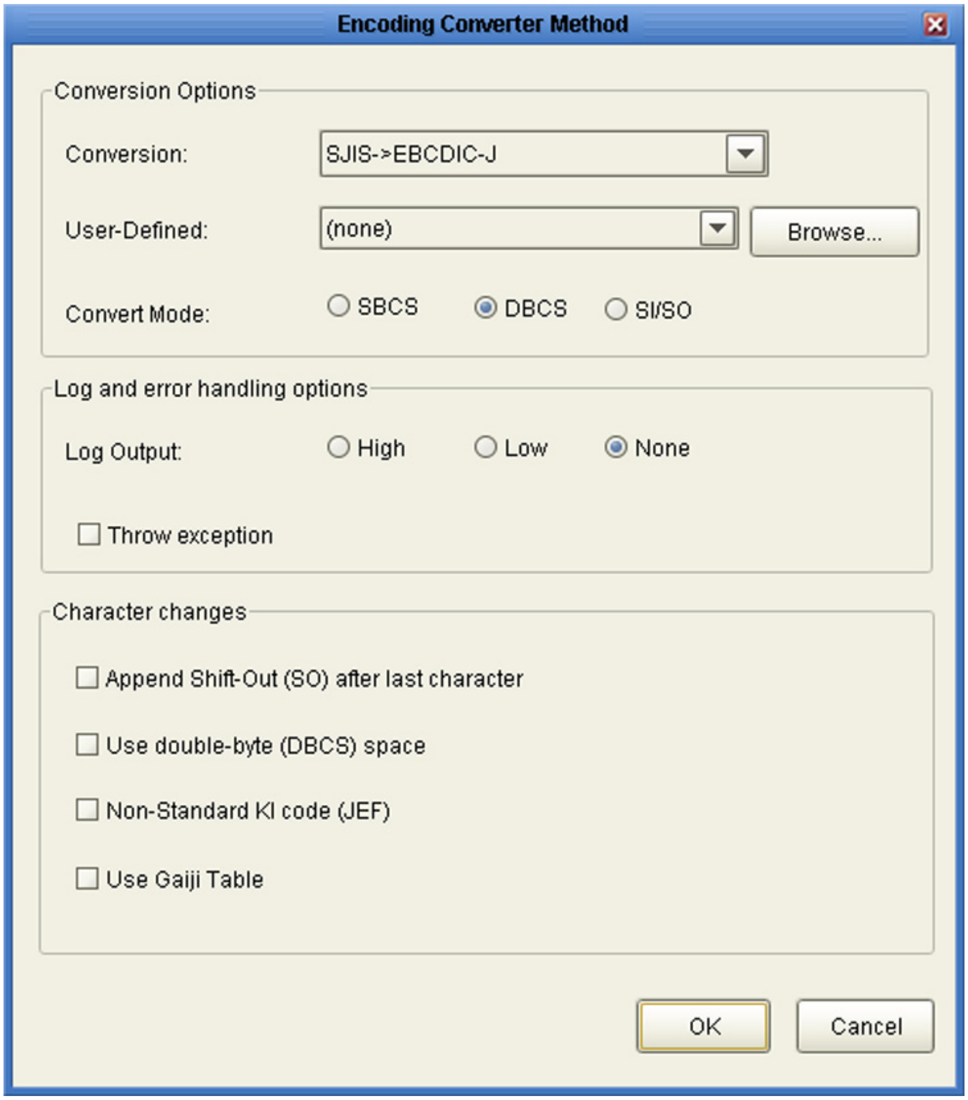


Figure 321 Encoding Converter Methods Dialog



Encoding Converter Method

Conversion Options

Conversion: SJIS->EBCDIC-J

User-Defined: (none) Browse...

Convert Mode: ○ SBCS ● DBCS ○ SI/SO

Log and error handling options

Log Output: ○ High ○ Low ● None

□ Throw exception

Character changes

□ Append Shift-Out (SO) after last character

□ Use double-byte (DBCS) space

□ Non-Standard KI code (JEF)

□ Use Gaiji Table

OK Cancel

Table 87 Encoding Converter Methods Dialog Options

Option	Description
Conversion	Specifies the type of byte-array encoding converter to use (see Table 85 and Table 86 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: - # = comment. - one line = one record. - one record contains a source code point in HEX format, followed by a blank space, followed by a destination code point in HEX format. - HEX format can be with or without **0x**. 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.

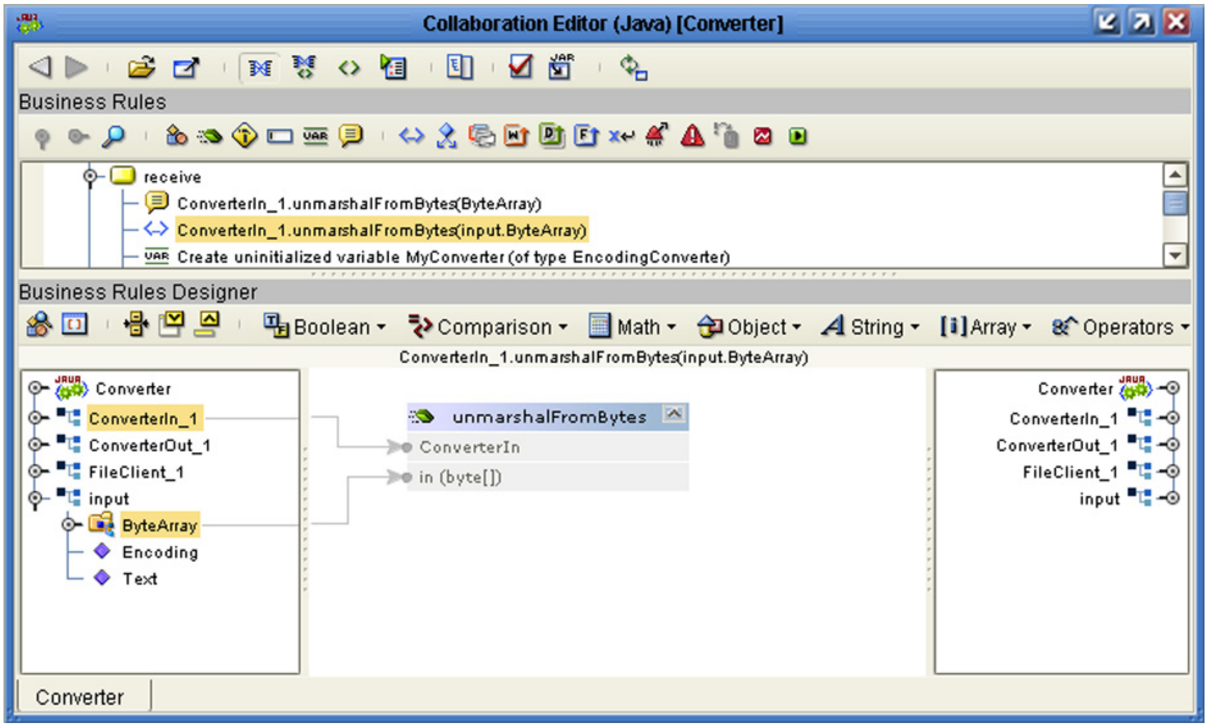
Table 87 Encoding Converter Methods Dialog Options

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 55.
2. Define your business rules to receive the input data as a byte array, using the **unmarshalFromBytes** Java method mapped as shown in Figure 322.

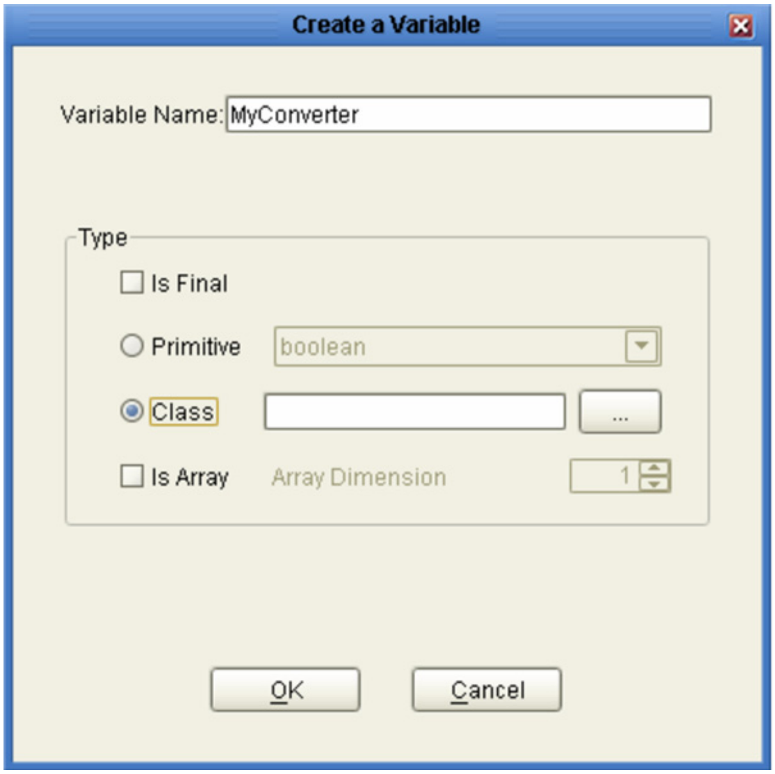
Figure 322 Using the unmarshalFromBytes Method



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 323).

Figure 323 Create a Variable Dialog



Create a Variable

Variable Name: MyConverter

Type

□ Is Final

○ Primitive boolean

● Class

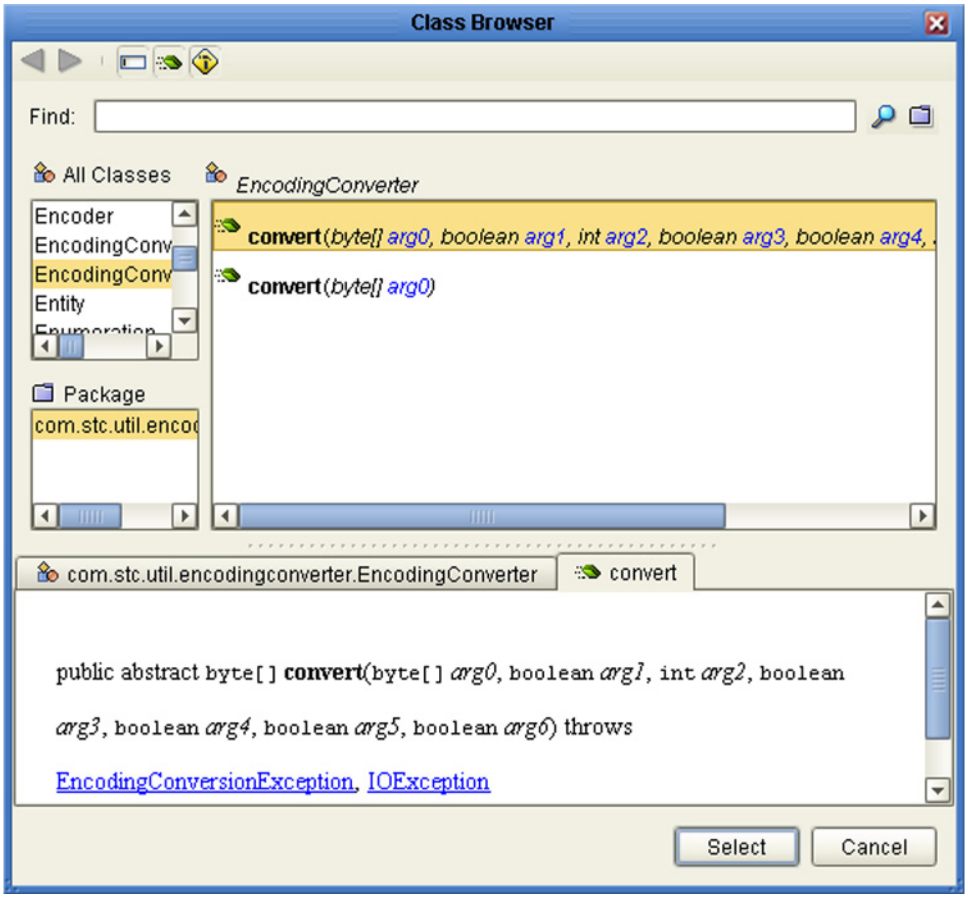
□ Is Array Array Dimension 1

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 324).

Figure 324 Class Browser Dialog



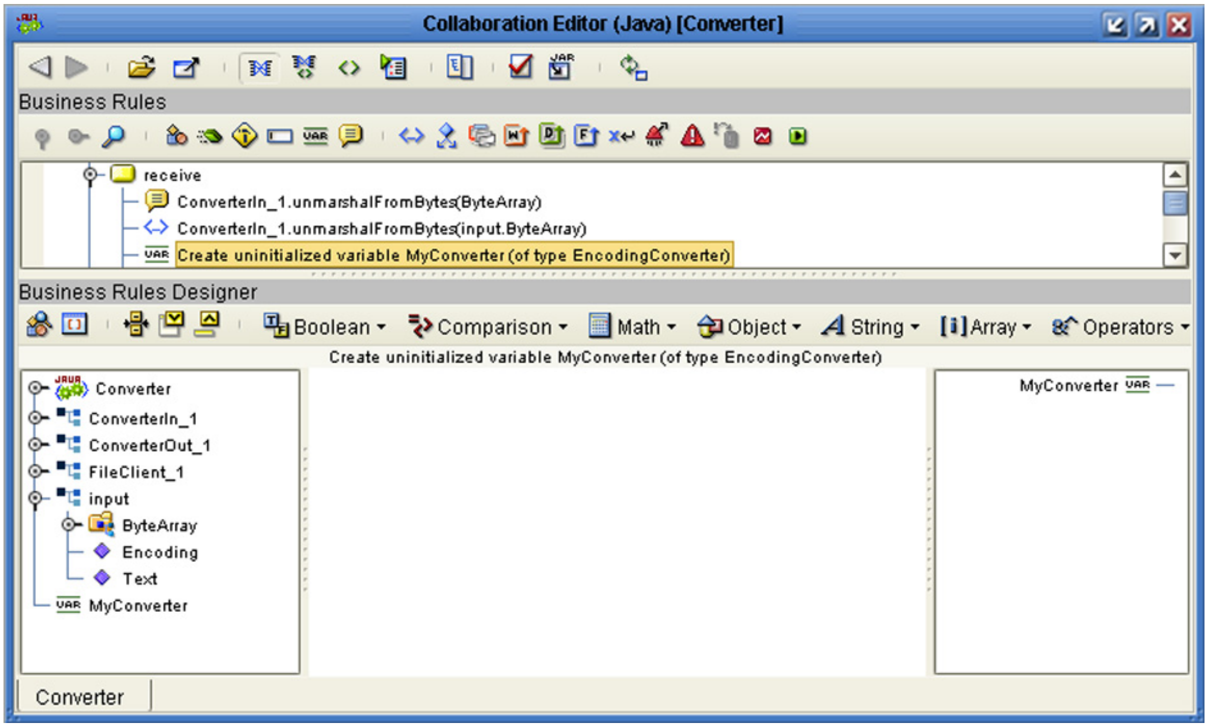
public abstract byte[] convert(byte[] arg0, boolean arg1, int arg2, boolean arg3, boolean arg4, boolean arg5, boolean arg6) throws EncodingConversionException, IOException

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 325).

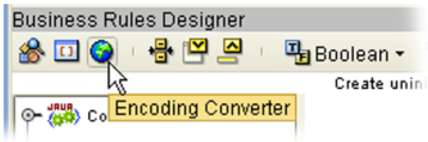
Figure 325 Adding the MyConverter Local Variable



ConverterIn_1.unmarshalFromBytes(ByteArray)
ConverterIn_1.unmarshalFromBytes(input.ByteArray)
Create uninitialized variable MyConverter (of type EncodingConverter)

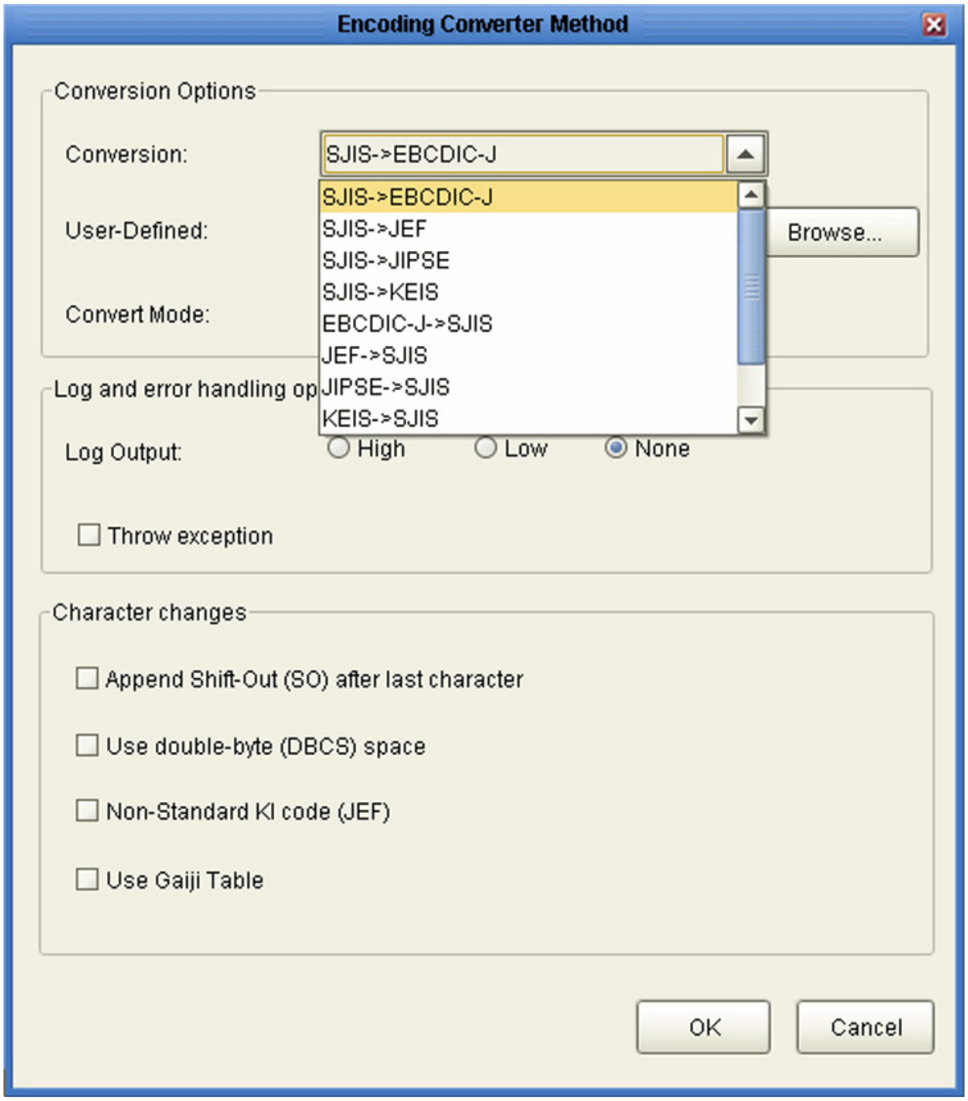
4 Click the **Encoding Converter** icon (see Figure 326) to display the *Encoding Converter Method* dialog shown in Figure 327.

Figure 326 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 327 Encoding Converter Method Dialog



Conversion Options

Conversion: SJIS->EBCDIC-J

User-Defined:

Convert Mode:

Log and error handling op

Log Output: ○ High ○ Low ● None

□ Throw exception

Character changes

□ Append Shift-Out (SO) after last character

□ Use double-byte (DBCS) space

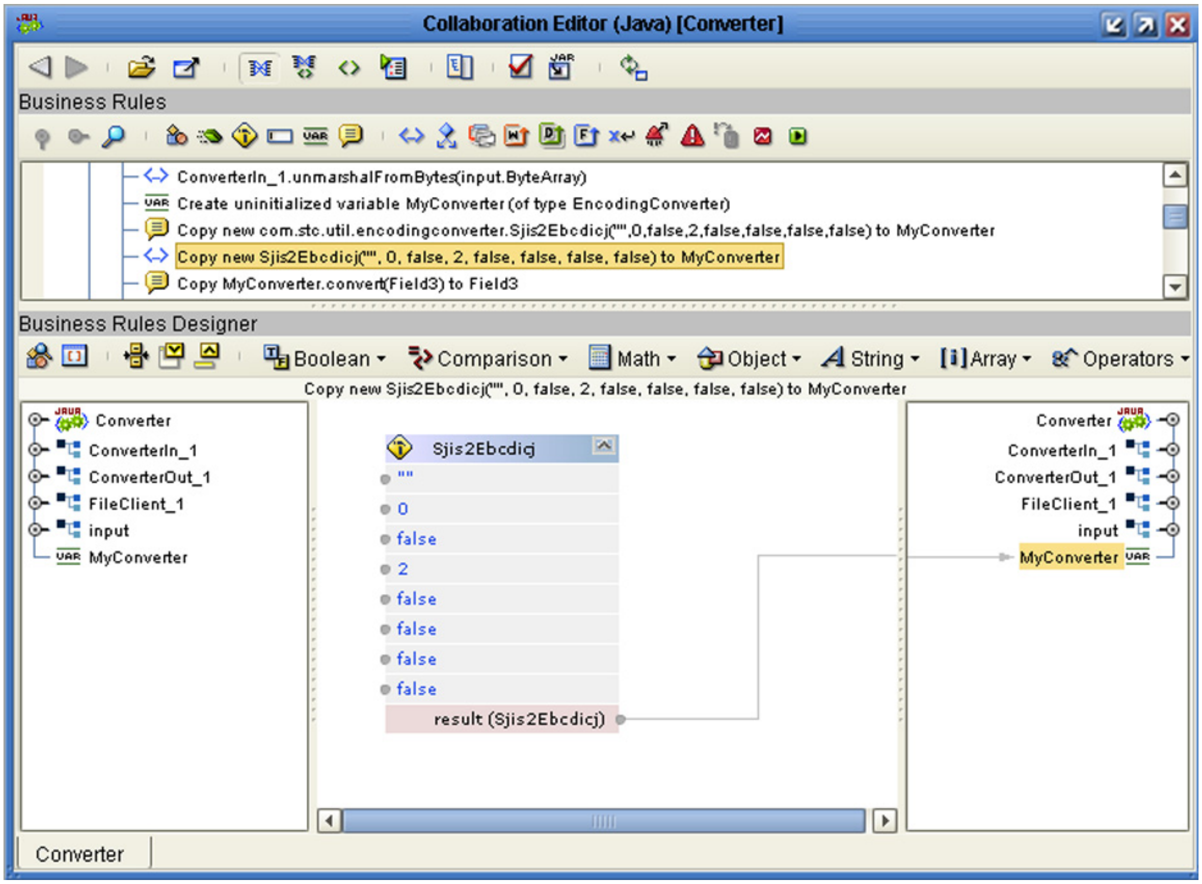
□ Non-Standard KI code (JEF)

□ Use Gaiji Table

5 From the Conversion Options group box, select **SJIS->EBCDIC-J**, as shown in Figure 327, which creates the encoding converter rule shown in Figure 328.

6 Select any desired options from the other group boxes and click **OK**.

Figure 328 Creating the Encoding Converter Rule



ConverterIn_1.unmarshalFromBytes(input.ByteArray)
Create uninitialized variable MyConverter (of type EncodingConverter)
Copy new com.stc.util.encodingconverter.Sjis2Ebcdicj("",0,false,2,false,false,false,false) to MyConverter
Copy new Sjis2Ebcdicj("", 0, false, 2, false, false, false, false) to MyConverter
Copy MyConverter.convert(Field3) to Field3

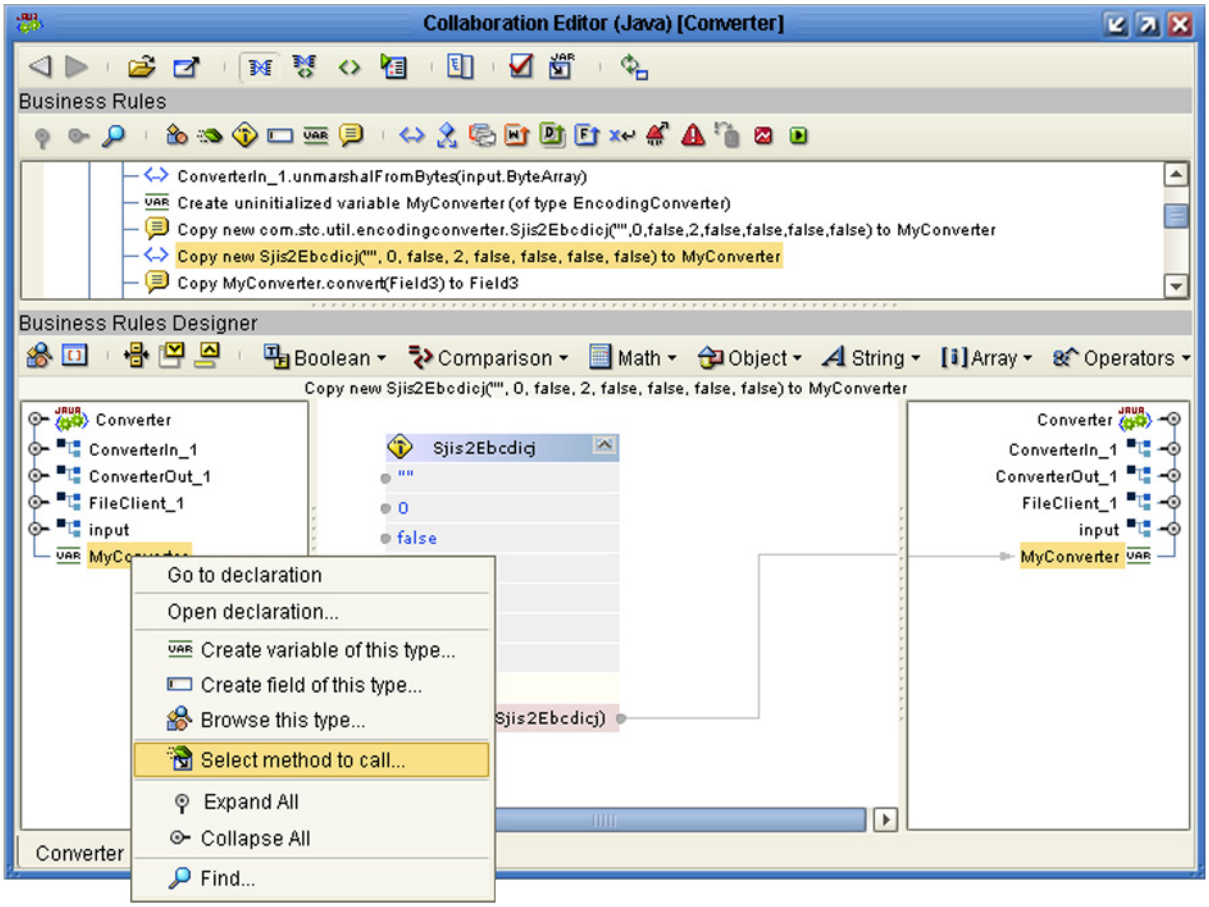
Copy new Sjis2Ebcdicj("", 0, false, 2, false, false, false, false) to MyConverter

Sjis2Ebcdicj: "", 0, false, 2, false, false, false, false; result (Sjis2Ebcdicj) → MyConverter

7 Map the result to the local variable created in step 3 (**MyConverter**) in the right pane as shown in Figure 328.

8 Click on the variable **MyConverter** in the left pane to display its context menu and click **Select a method to call** (see Figure 329). This presents the dialog shown in Figure 330.

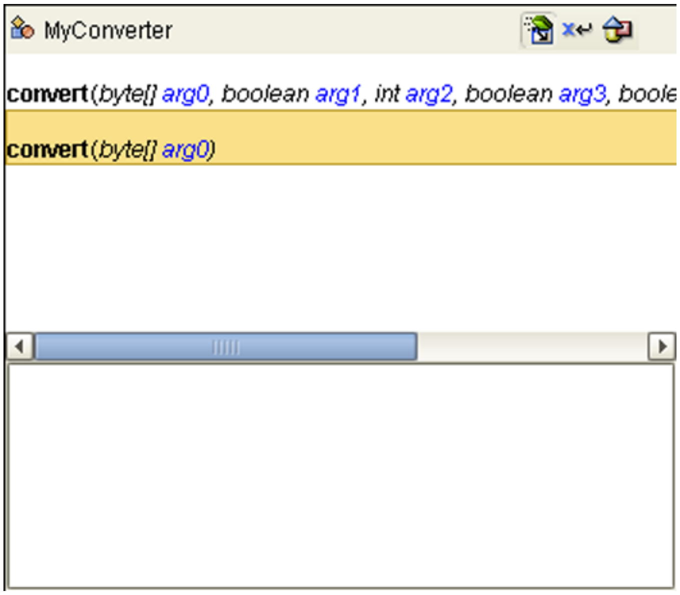
Figure 329 Selecting a Method



ConverterIn_1.unmarshalFromBytes(input.ByteArray)
Create uninitialized variable MyConverter (of type EncodingConverter)
Copy new com.stc.util.encodingconverter.Sjis2Ebcdicj("",0,false,2,false,false,false,false) to MyConverter
Copy new Sjis2Ebcdicj("", 0, false, 2, false, false, false, false) to MyConverter
Copy MyConverter.convert(Field3) to Field3

Copy new Sjis2Ebcdicj("", 0, false, 2, false, false, false, false) to MyConverter

Figure 330 Convert Method List



MyConverter

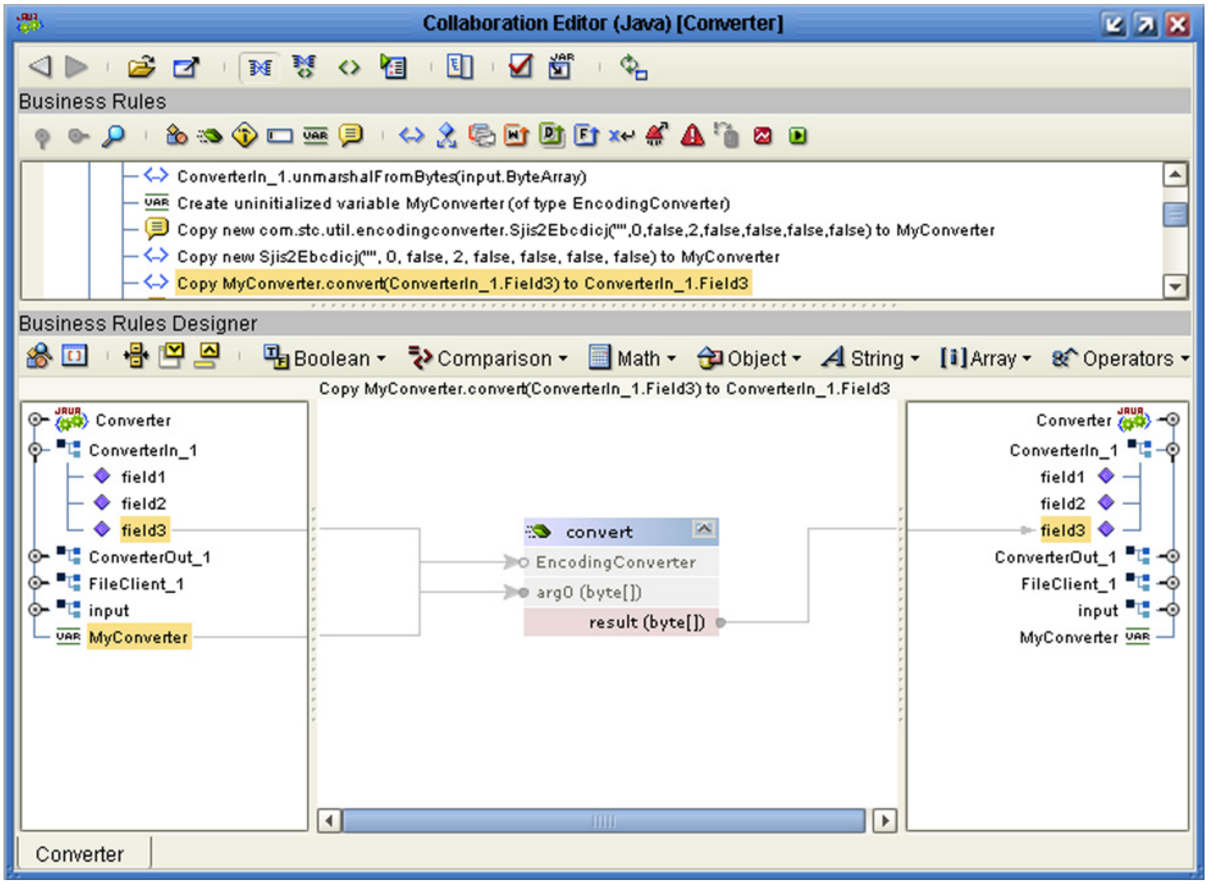
convert(byte[] arg0, boolean arg1, int arg2, boolean arg3, boole

convert(byte[] arg0)

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 331), and map the **result** to the node that gets the converted data (**ConverterOut_1 / field3**).

Figure 331 Data Mapping

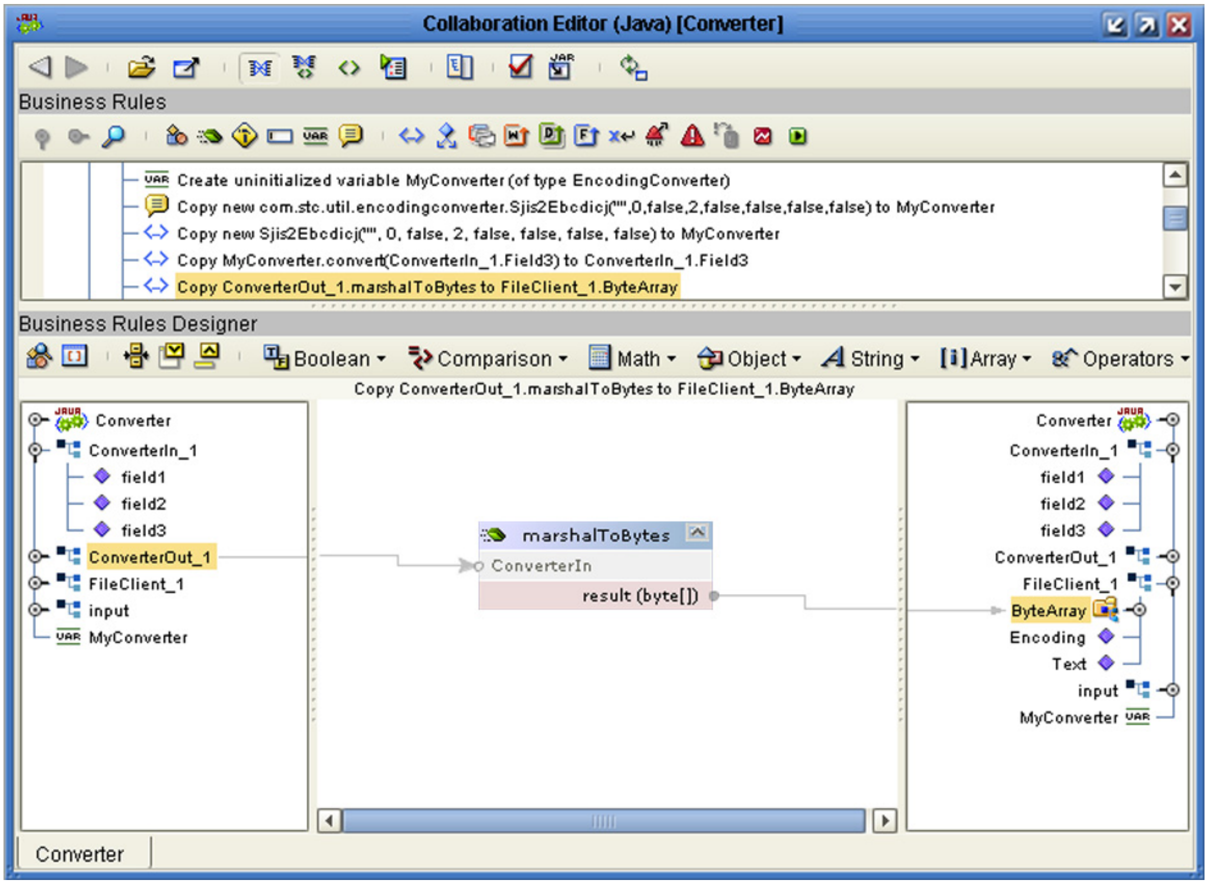


```
ConverterIn_1.unmarshalFromBytes(input.ByteArray)
Create uninitialized variable MyConverter (of type EncodingConverter)
Copy new com.stc.util.encodingconverter.Sjis2Ebcdicj("",0,false,2,false,false,false,false) to MyConverter
Copy new Sjis2Ebcdicj("", 0, false, 2, false, false, false, false) to MyConverter
Copy MyConverter.convert(ConverterIn_1.Field3) to ConverterIn_1.Field3
```

Copy MyConverter.convert(ConverterIn_1.Field3) to ConverterIn_1.Field3

11 When finished with the Collaboration rules, marshal the output data using the **marshalToBytes** Java method as shown in Figure 332.

Figure 332 Marshaling Output Data



Create uninitialized variable MyConverter (of type EncodingConverter)

Copy new com.stc.util.encodingconverter.Sjis2Ebcdicj("",0,false,2,false,false,false,false) to MyConverter

Copy new Sjis2Ebcdicj("", 0, false, 2, false, false, false, false) to MyConverter

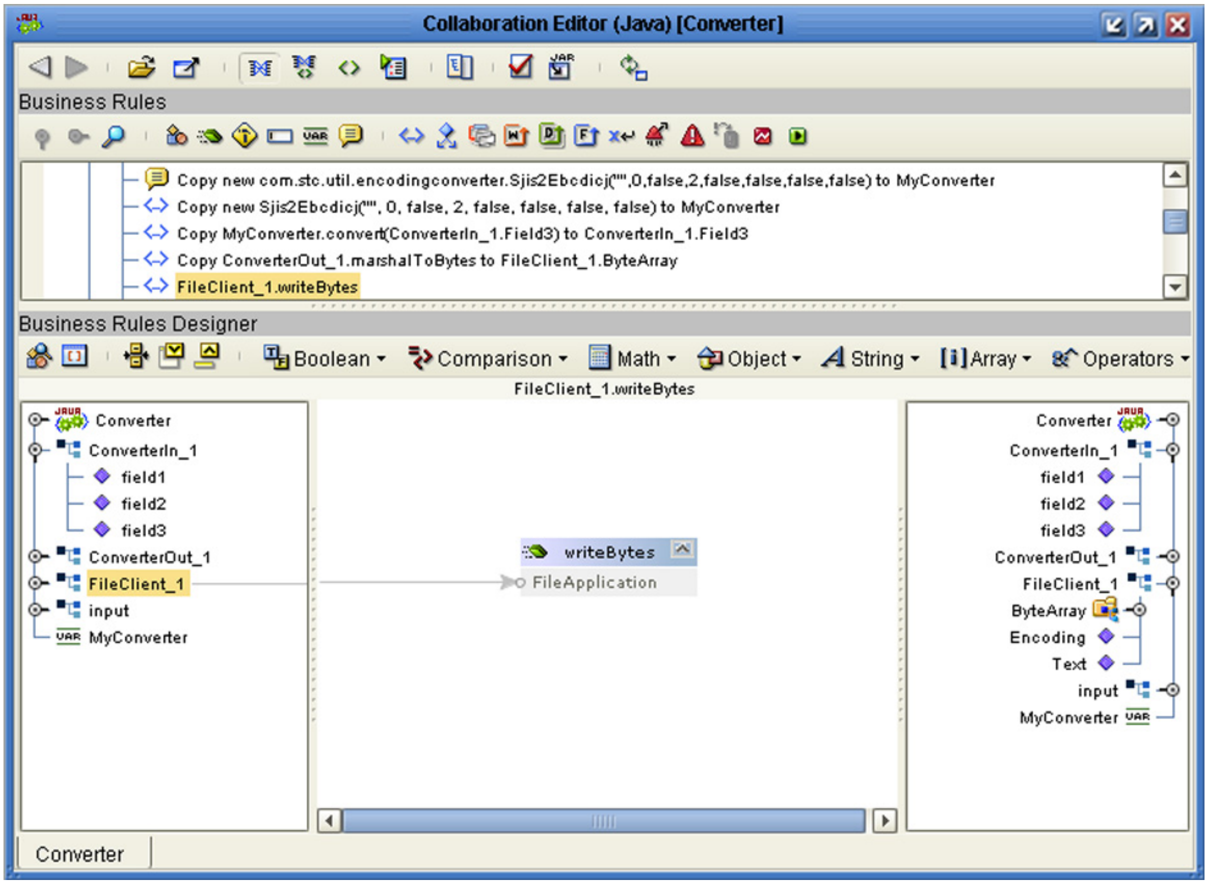
Copy MyConverter.convert(ConverterIn_1.Field3) to ConverterIn_1.Field3

Copy ConverterOut_1.marshalToBytes to FileClient_1.ByteArray

Copy ConverterOut_1.marshalToBytes to FileClient_1.ByteArray

12 Finally, output the data using the **writeBytes** Java method as shown in Figure 333.

Figure 333 Output Data



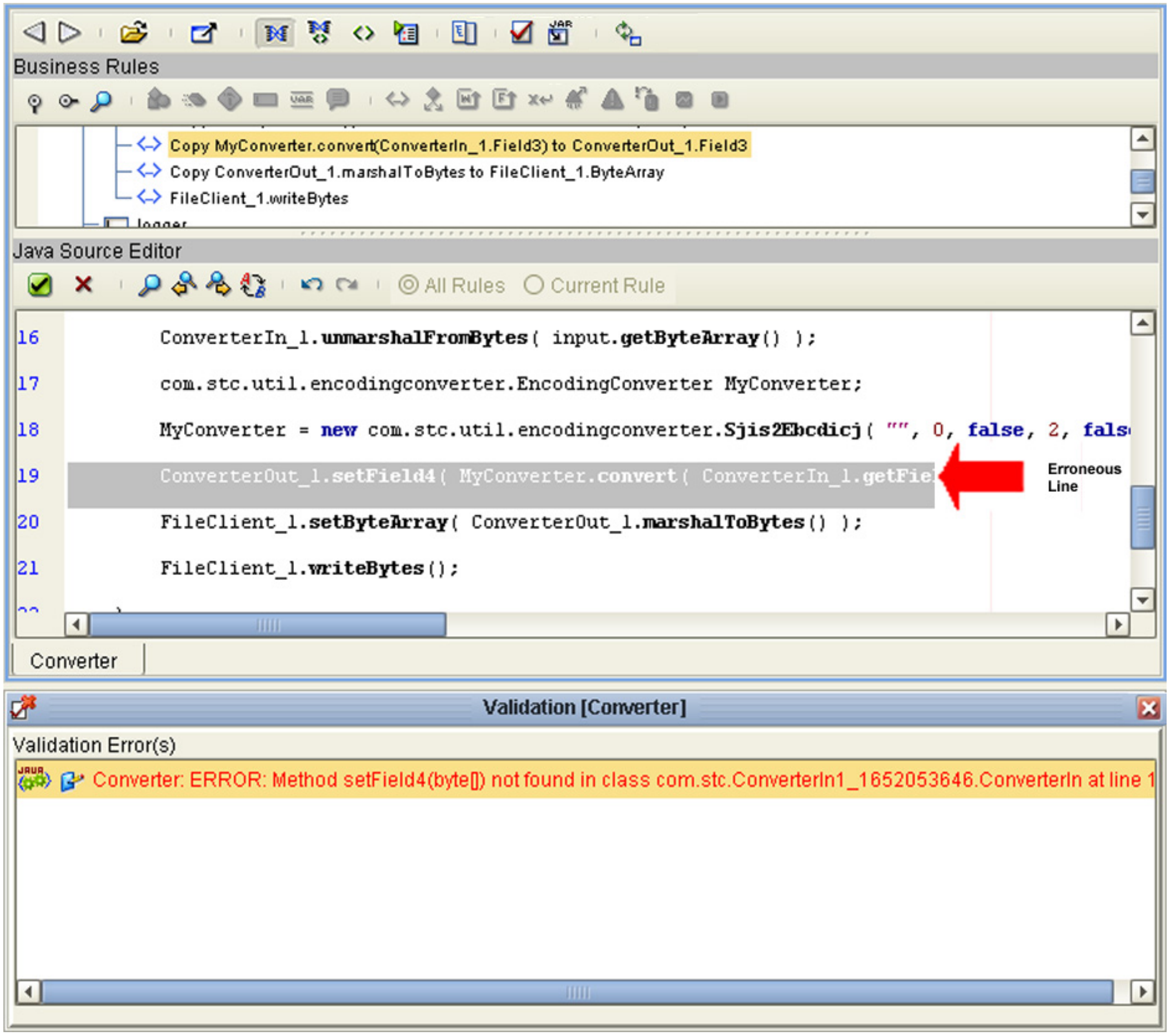
```
Copy new com.stc.util.encodingconverter.Sjis2Ebcdicj("",0,false,2,false,false,false,false) to MyConverter
Copy new Sjis2Ebcdicj("", 0, false, 2, false, false, false, false) to MyConverter
Copy MyConverter.convert(ConverterIn_1.Field3) to ConverterIn_1.Field3
Copy ConverterOut_1.marshalToBytes to FileClient_1.ByteArray
FileClient_1.writeBytes
```

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 335.

Figure 334 Validate Icon

Figure 335 Validating Java Code



```
Copy MyConverter.convert(ConverterIn_1.Field3) to ConverterOut_1.Field3
Copy ConverterOut_1.marshalToBytes to FileClient_1.ByteArray
FileClient_1.writeBytes
logger
```

```
        ConverterIn_1.unmarshalFromBytes( input.getByteArray() );
        com.stc.util.encodingconverter.EncodingConverter MyConverter;
        MyConverter = new com.stc.util.encodingconverter.Sjis2Ebcdicj( "", 0, false, 2, fals
        ConverterOut_1.setField4( MyConverter.convert( ConverterIn_1.getFie
        FileClient_1.setByteArray( ConverterOut_1.marshalToBytes() );
        FileClient_1.writeBytes();
```

```
Converter: ERROR: Method setField4(byte[]) not found in class com.stc.ConverterIn1_1652053646.ConverterIn at line 1
```

Chapter 11

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 387
- **Creating an XSLT-based Collaboration Definition** on page 390
- **Using the Collaboration Editor (XSLT)** on page 396
- **Graphical Collaboration Methods (XSLT)** on page 402

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 though 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 81).*

As with other eGate Integrator components, it is essential to manage versions of Collaboration Definitions carefully. See **Managing Component Versions** on page 81 for descriptions of various version control features applicable to Collaboration Definitions.

11.2 Using the Collaboration Definition Context Menu

Figure 336 Collaboration Definition Context Menu

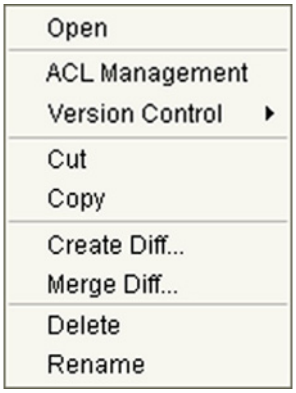


Table 88 Collaboration Definition Context Menu Options

Option	Function
Open	Opens the appropriate Collaboration Editor, showing the selected Collaboration Definition. See **Using the Collaboration Editor (XSLT)** on page 396.
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 *Sun SeeBeyond eGate™ Integrator System Administration Guide*.
Version Control	Accesses the version control features for the selected component. See **Accessing Component Version Control Features** on page 82 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 within the same branch (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 **Moving and Replicating Repository Objects** on page 100 for additional information.
Copy	Copies the selected Collaboration Definition (Java only), after which you can paste it to other Projects within the same branch (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 **Moving and Replicating Repository Objects** on page 100 for additional information.

Table 88 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 **Merging Changes from Another Version** on page 97.
Merge Diff	Presents a dialog with which you can merge modifications into a specific version of the selected Collaboration Definition. See **Merging Changes from Another Version** on page 97.
Delete	Deletes the selected Collaboration Definition, subject to the following conditions: ▪ You have *write* privileges for the component (see ACL Management, above). ▪ You have the component checked out to your workspace. ▪ The component is not checked out by anyone other than yourself. If these conditions are met, a dialog is presented in which you confirm that you want to delete the selected component. Clicking **Yes** then deletes the component.
Rename	Activates the field, allowing you to rename the selected Project.

11.3 Creating an XSLT-based Collaboration Definition

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.

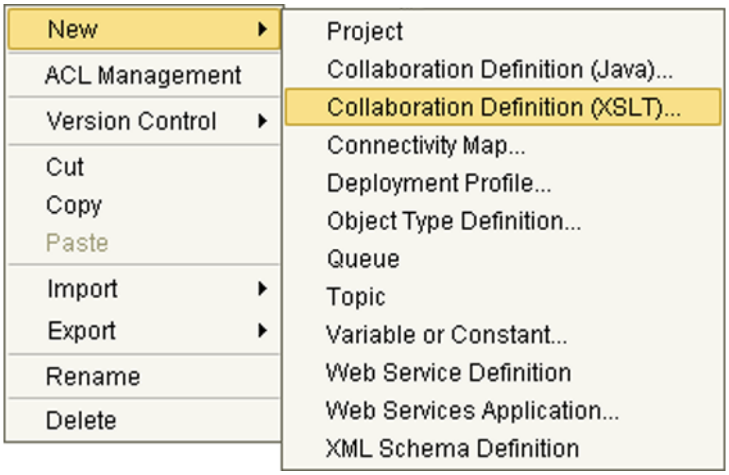
The Collaboration Definition Wizard (XSLT) guides you through the initial phases of creating an XSLT-based Collaboration Definition, and then invokes the Collaboration Editor (XSLT). Creating a Collaboration Definition (XSLT)

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 337).

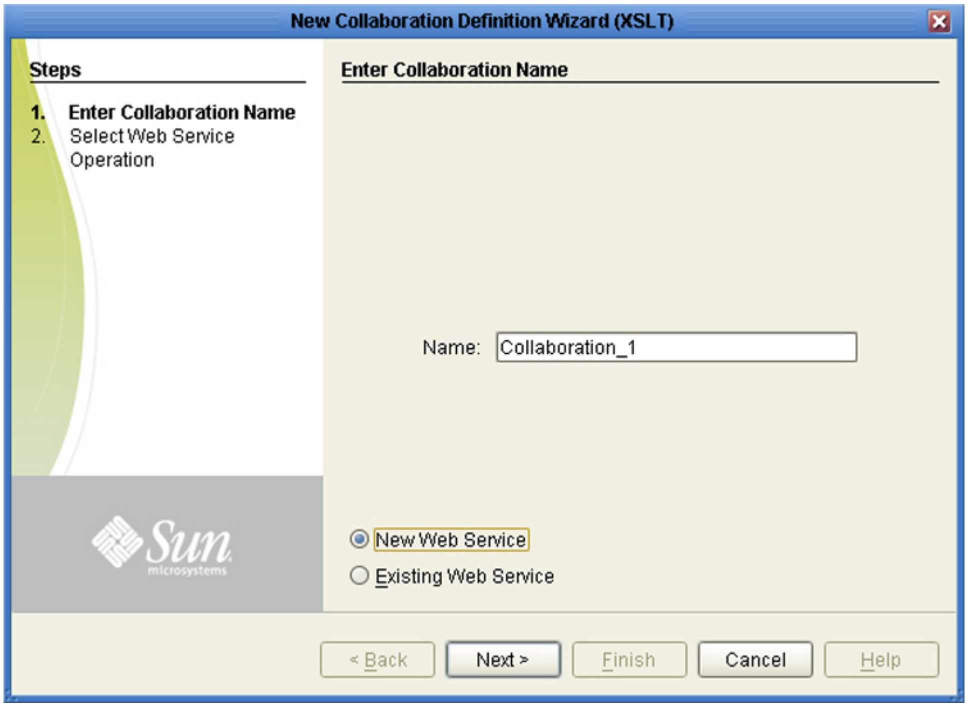
Figure 337 Project Context Menu - New Web Service Definition



2 Select **New > Collaboration Definition (XSLT)** to invoke the Collaboration Definition Wizard (XSLT).

3 Enter a **Name** for your Collaboration, as shown in Figure 338.

Figure 338 Collaboration Wizard (XSLT) Dialog


New Collaboration Definition Wizard (XSLT)
Steps
1. Enter Collaboration Name
2. Select Web Service Operation
Sun microsystems
Enter Collaboration Name
< Back
Next >
Finish
Cancel
Help

Name: Collaboration_1

● New Web Service
○ Existing Web Service

4 Select a web service, which can be either:

- A **new** web service operation.
- An **existing** web service operation (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; see:

- **New Web Service Operation** on page 392
- **Existing Web Service Operation** on page 395

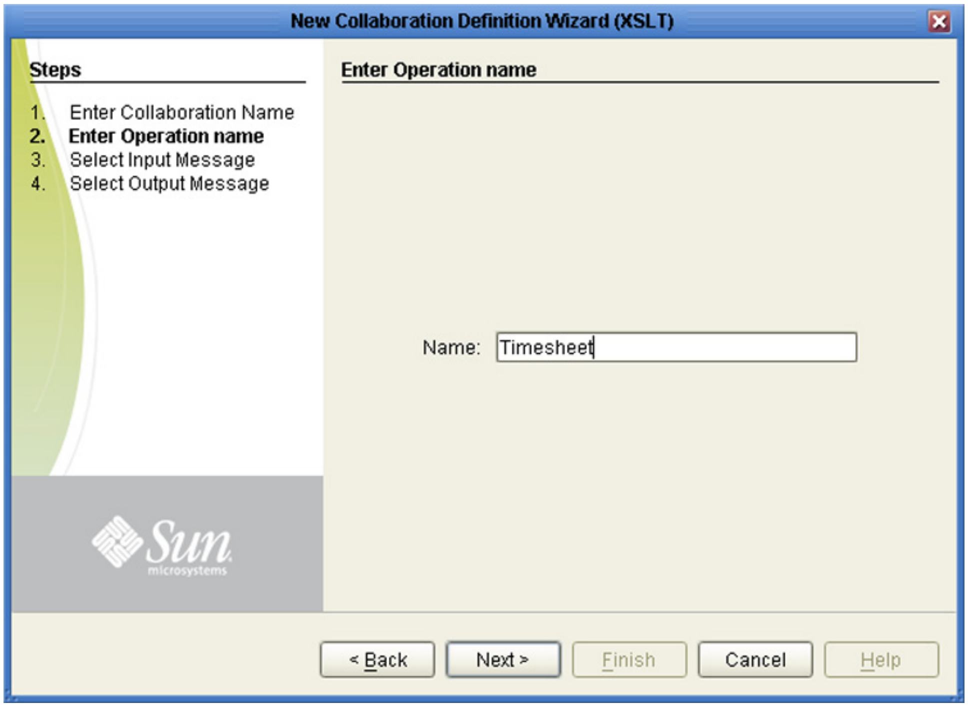
11.3.1 New Web Service Operation

If you selected a new web service, you will be presented with the Wizard dialog shown in Figure 339.

To create a new web service operation

1 Enter an operation name, as shown in Figure 339. This will become the *method* that can be used to invoke the XSLT-based Collaboration as a web service.

Figure 339 New Web Service: Operation Name



New Collaboration Definition Wizard (XSLT)

Steps

1. Enter Collaboration Name
2. **Enter Operation name**
3. Select Input Message
4. Select Output Message

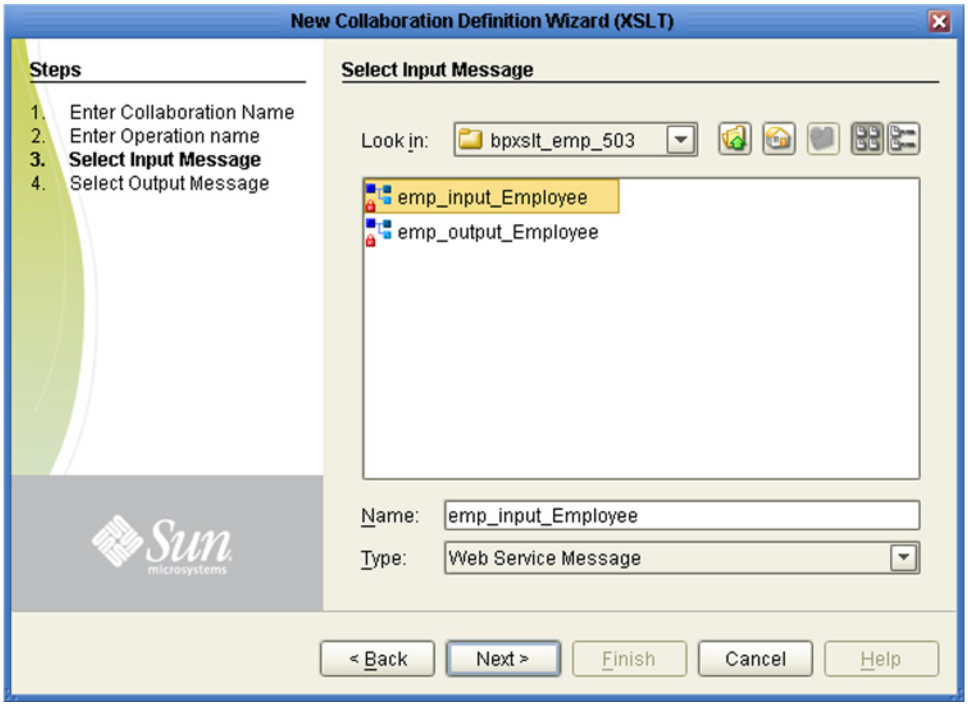
Enter Operation name

Name: Timesheet

2 Click **Next** to proceed to the next Wizard dialog.

3 Select the input web service message, as shown in Figure 340.

Figure 340 New Web Service: Input Message



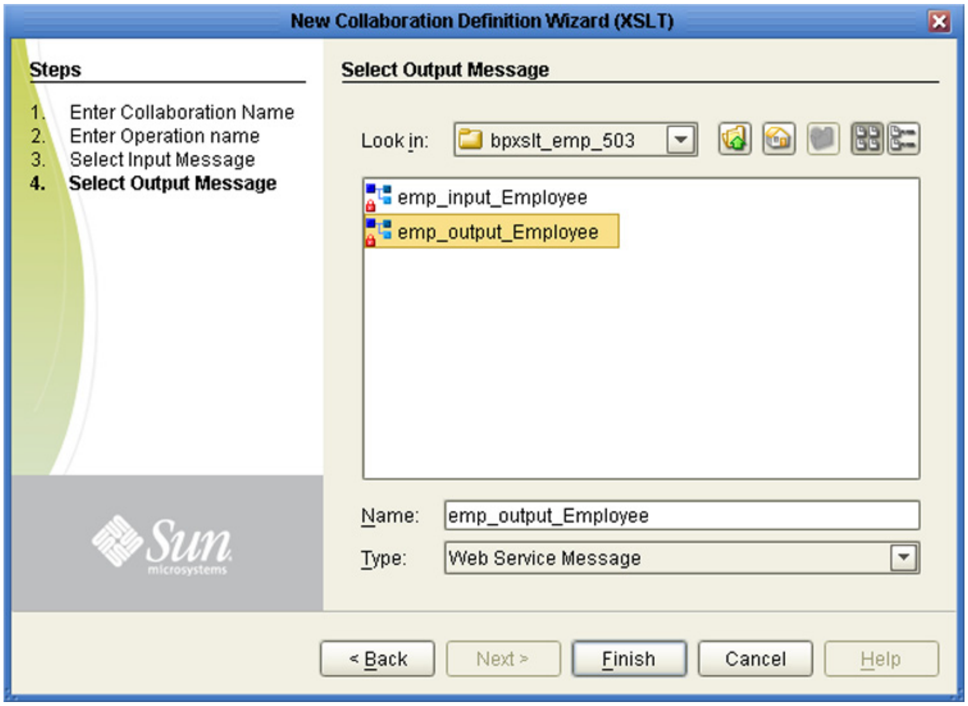
Name: emp_input_Employee

Type: Web Service Message

4 Click **Next** to proceed to the next Wizard dialog.

5 Select the output web service message, as shown in Figure 341.

Figure 341 New Web Service: Output Message



Name: emp_output_Employee

Type: Web Service Message

6 Click **Finish** to proceed to the Collaboration Editor (XSLT).

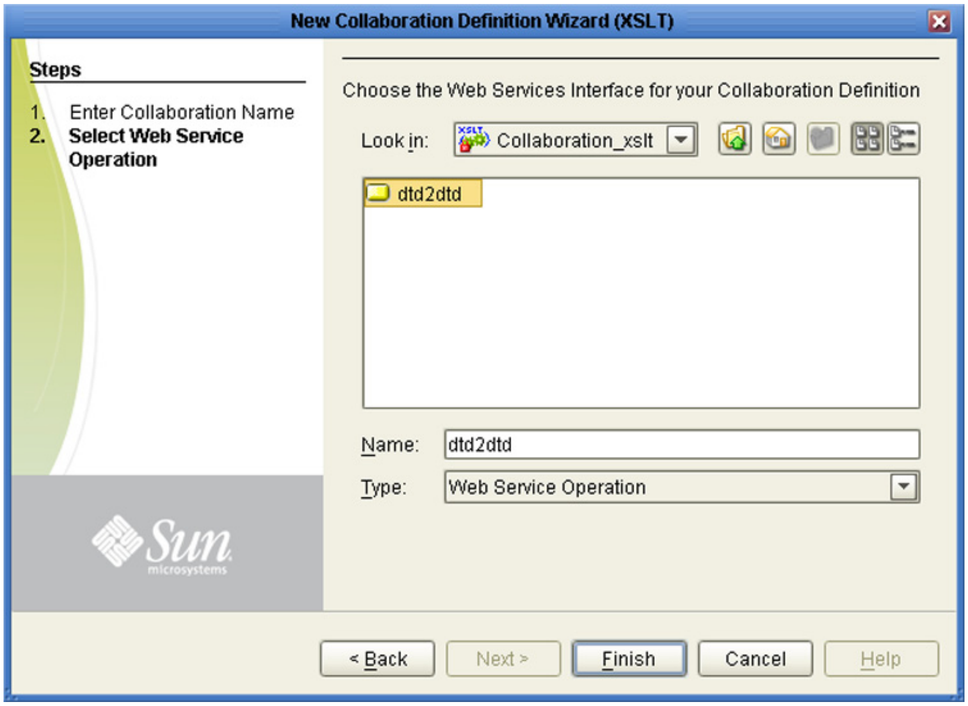
11.3.2 Existing Web Service Operation

If you selected an existing web service operation, you will be presented with the Wizard dialog shown in Figure 342.

To use an existing Web service

1 Select a web service operation, which must be an installed web service.

Figure 342 Existing Web Service: Select Operation



New Collaboration Definition Wizard (XSLT)

Steps

1. Enter Collaboration Name
2. **Select Web Service Operation**

Choose the Web Services Interface for your Collaboration Definition

Look in: Collaboration_xslt

dtd2dtd

Name: dtd2dtd

Type: Web Service Operation

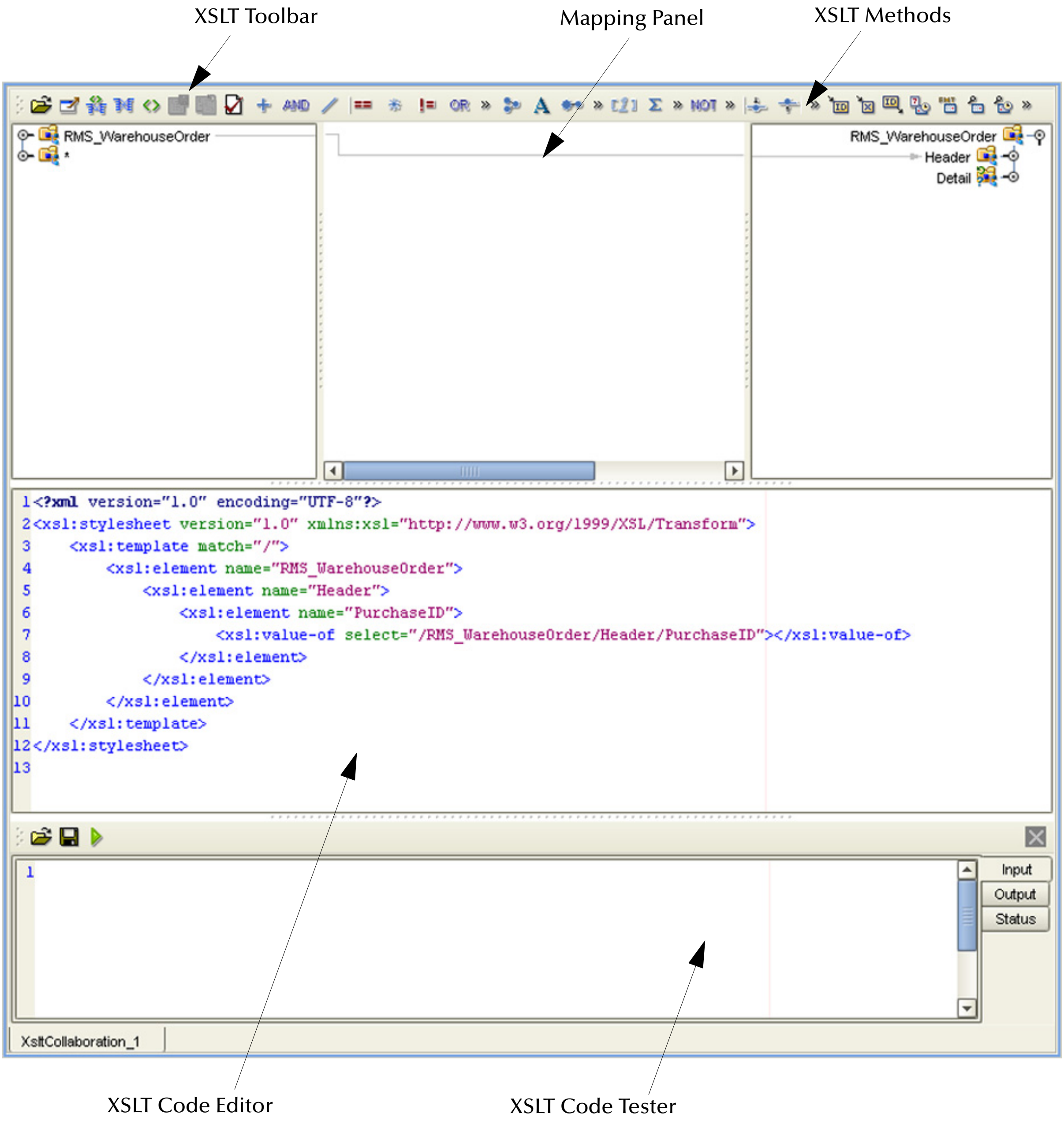
2 Click **Finish** to proceed to the Collaboration Editor (XSLT).

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 343.

Figure 343 Collaboration Editor (XSLT)



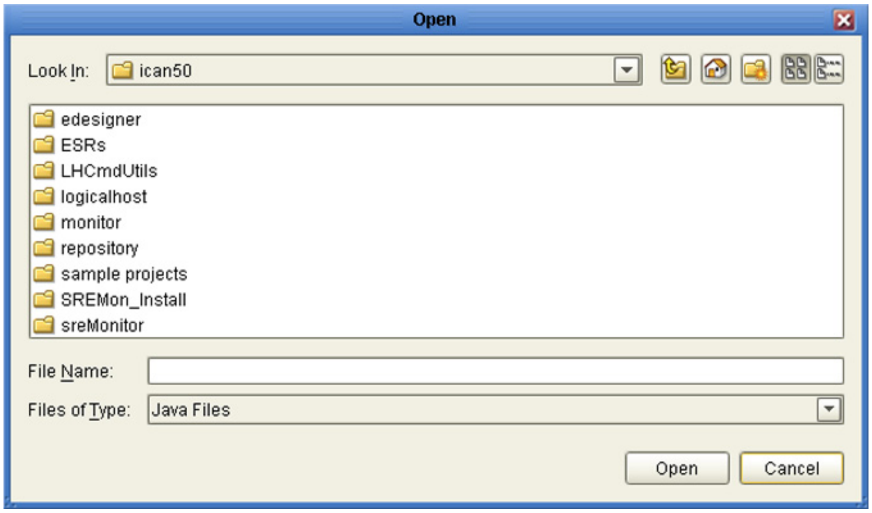
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <xsl:element name="RMS_WarehouseOrder">
            <xsl:element name="Header">
                <xsl:element name="PurchaseID">
                    <xsl:value-of select="/RMS_WarehouseOrder/Header/PurchaseID"></xsl:value-of>
                </xsl:element>
            </xsl:element>
        </xsl:element>
    </xsl:template>
</xsl:stylesheet>
```

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 89. 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.

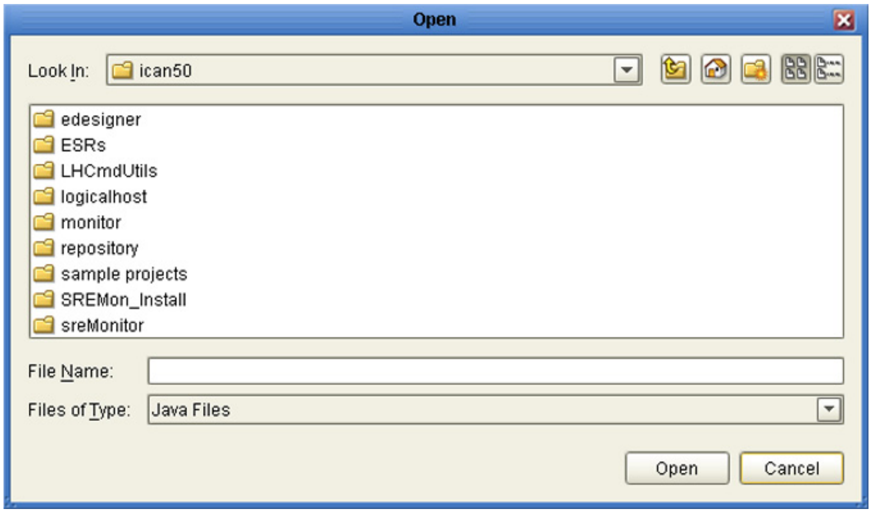
11.4.1 XSLT Toolbar Icons

Table 89 XSLT Toolbar Icons

Icon	Command	Function
	Import File	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.  Look In: ican50 edesigner ESRs LHCmdUtils logicalhost monitor repository sample projects SREMon_Install sreMonitor File Name: Files of Type: Java Files
	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.

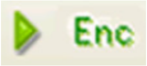
11.4.2 XSLT Tester Icons

Table 90 XSLT Tester Icons

Icon	Command	Function
📂	Import File	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.  Open Look In: ican50 edesigner ESRs LHCmdUtils logicalhost monitor repository sample projects SREMon_Install sreMonitor File Name: Files of Type: Java Files Open Cancel
💾	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 55), an *Encoding* icon (see Figure 344) 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 344 Encoding Icon



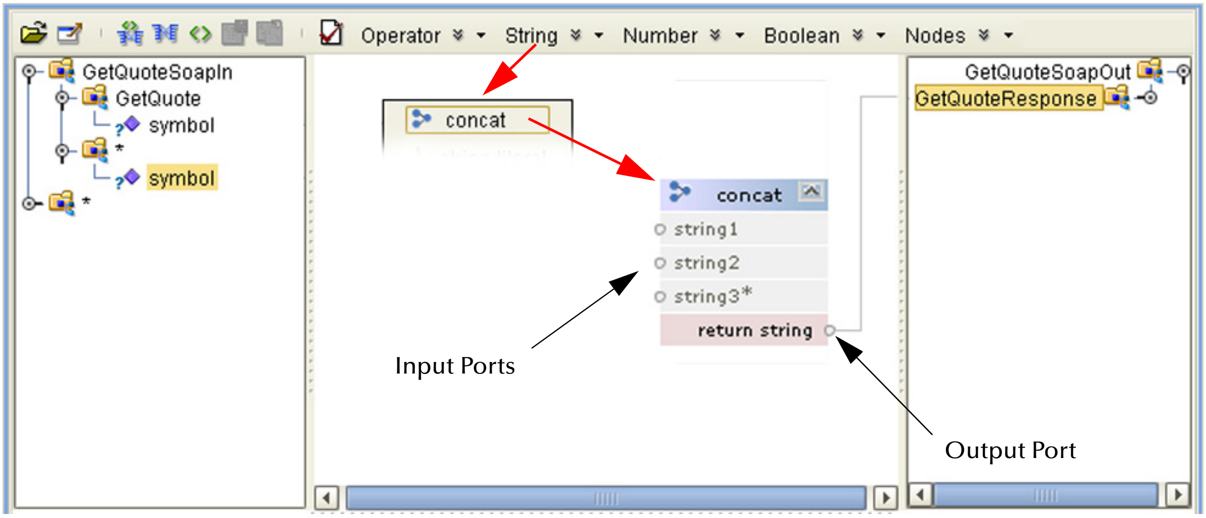
Note: *Extended language options currently apply only to Japanese and Korean localized versions of eGate Integrator.*

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 402). 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.

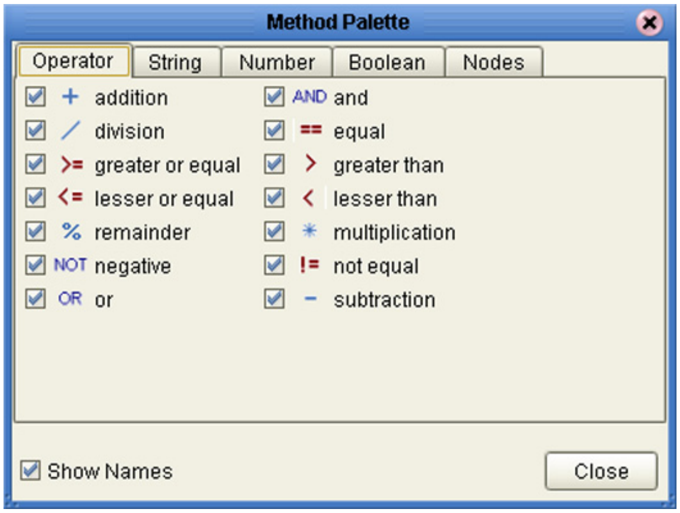
Figure 345 Collaboration Definition Editor (XSLT): Method Menus and Boxes



Clicking the **Settings** option in any menu presents the dialog shown in Figure 346, 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 346 Collaboration Method Palette (XSLT)



Method Palette

Operator | String | Number | Boolean | Nodes

☑ + addition
☑ / division
☑ >= greater or equal
☑ <= lesser or equal
☑ % remainder
☑ NOT negative
☑ OR or
☑ AND and
☑ == equal
☑ > greater than
☑ < lesser than
☑ * multiplication
☑ != not equal
☑ - subtraction

☑ Show Names

Close

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 345 on page 400**, 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 347); you can collapse them (see Figure 348) 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 347 Expanded Method Box

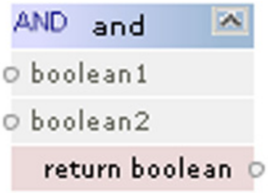
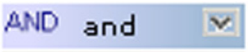


Figure 348 Collapsed Method Box



11.5 Graphical Collaboration Methods (XSLT)

11.5.1 Boolean Methods

Figure 349 Boolean Methods Menu

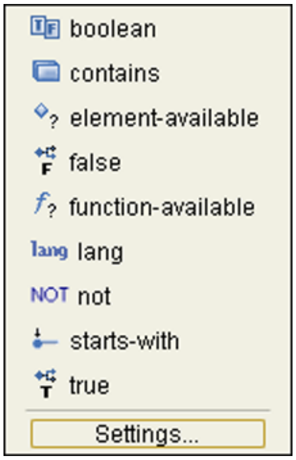


Table 91 Boolean Collaboration Methods (XSLT)

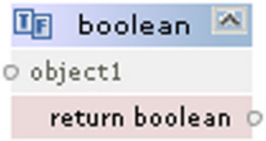
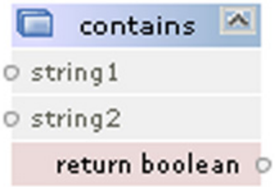
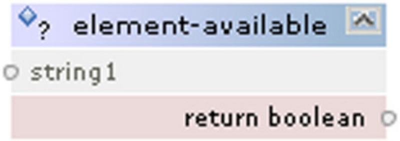
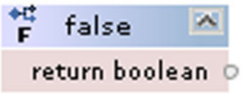
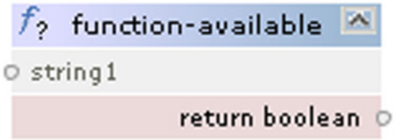
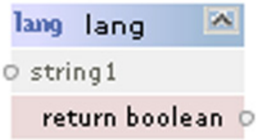
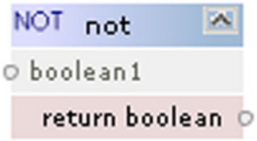
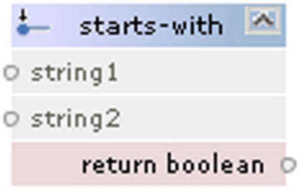
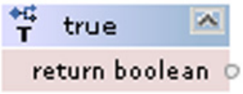
Method Box	Description/Usage
	At run time, the **boolean** method converts the argument *object1* to a Boolean true or false as follows: ▪ A number is true if and only if it is neither ±zero nor NaN (not a number). ▪ A node-set is true if and only if it is non-empty. ▪ A string is true if and only if it is non-zero. ▪ 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.
	At run time, the **contains** method returns Boolean **true** if the first argument (*string1*) contains the second argument (*string2*); if not, returns Boolean **false**.

Table 91 Boolean Collaboration Methods (XSLT)

Method Box	Description/Usage
	The argument *string1* must be a QName, which is expanded into an expanded-name using the namespace declarations in scope for the expression. At run time, the **element-available** 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.
	At run time, the **false** method returns Boolean **false**.
	The argument *string1* must be a QName, which is expanded into an expanded-name using the namespace declarations in scope for the expression. At run time, the **function-available** 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.
	At run time, the **lang** (**language**) method returns Boolean **true** or **false** depending upon whether the language of the context node as specified by *xml:lang* attributes is the same as, or is a sub-language of, the language specified by the argument string (*string1*). Returns Boolean **false** if the attribute *xml:lang* does not exist.
	At run time, the **not** method returns the inverse of *boolean1*.
	At run time, the **starts-with** method returns Boolean **true** if the first argument (*string1*) starts with the second argument (*string2*); if not, returns Boolean **false**.
	At run time, the **true** method returns Boolean **true**.

11.5.2 Nodes Methods

Figure 350 Nodes Methods Menu

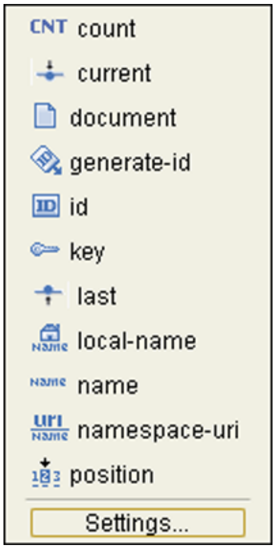


Table 92 Nodes Collaboration Methods (XSLT)

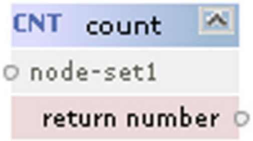
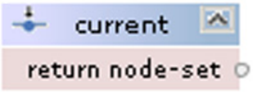
Method Box	Description/Usage
	At run time, the **count** method returns the number of nodes in the argument *node-set1*.
	At run time, the **current** method returns a node-set that has the current node as its only member.

Table 92 Nodes Collaboration Methods (XSLT)

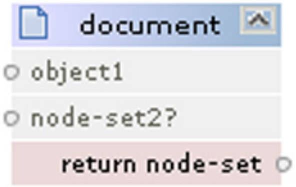
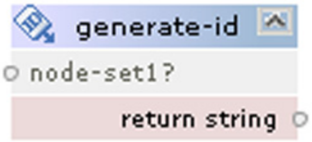
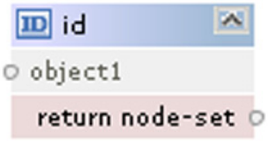
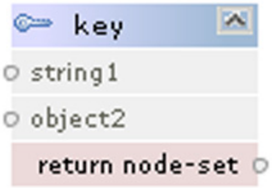
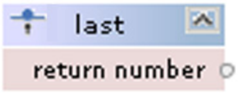
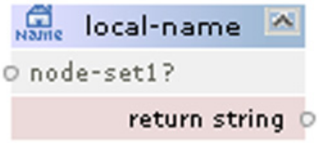
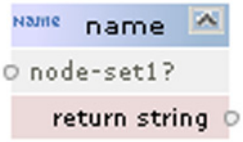
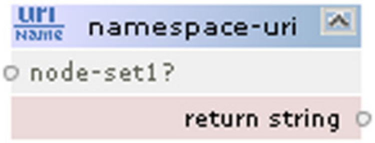
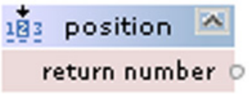
Method Box	Description/Usage
	The **document** method allows access to XML documents other than the main source document. ▪ When the method has exactly one argument and the argument is a node-set, then for each node in the argument node-set, the result is the union of the result of calling the document 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. ▪ 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.
	At run time, the **generate-id** method returns a string that uniquely identifies the node in the argument *node-set1?* 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.
	When the argument *object1* is not of type node-set, the **id** 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.
	The **key** method does for keys what the id function does for IDs. The value of the first argument (*string1*), 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. ▪ 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. ▪ 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.
	At run time, the **last** method returns a number equal to the context size from the expression evaluation context.

Table 92 Nodes Collaboration Methods (XSLT)

Method Box	Description/Usage
	At run time, the **local-name** method returns the local part of the expanded-name of the node in the argument *node-set1?* 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.
	At run time, the **name** method returns a string containing a QName representing the expanded-name of the node in the argument *node-set1?* 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 it omitted, it defaults to a node-set with the context node as its only member.
	At run time, the **namespace-uri** method returns the namespace URI of the expanded-name of the node in the argument *node-set1?* 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.
	At run time, the **position** method returns a number equal to the context position from the expression evaluation context.

11.5.3 Number Methods

Figure 351 Number Methods Menu

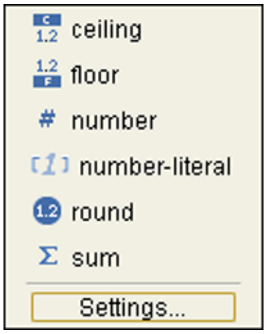


Table 93 Number Collaboration Methods (XSLT)

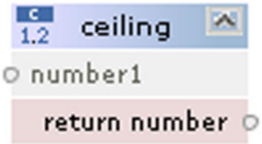
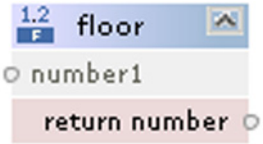
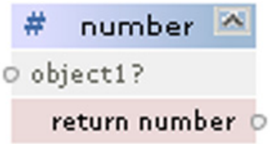
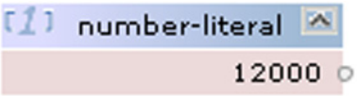
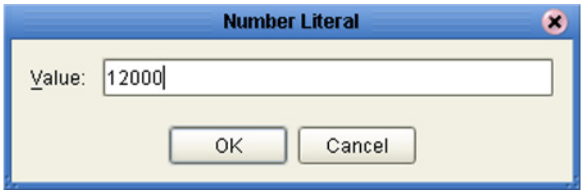
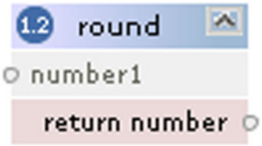
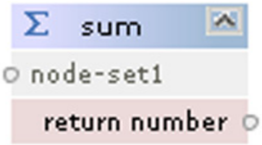
Method Box	Description/Usage
	At run time, the **ceiling** method returns the smallest (closest to negative infinity) number that is not less than the argument and that is an integer.
	At run time, the **floor** method returns the largest (closest to positive infinity) number that is not greater than the argument and that is an integer.
	At run time, the **number** method converts its argument (*object1*) 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.
	Dragging the **number-literal** icon first presents a dialog in which you enter the literal value, such as “12000”:  Value: 12000 At run time, the method returns a number having the specified value.

Table 93 Number Collaboration Methods (XSLT)

Method Box	Description/Usage
	At run time, the **round** method returns the number that is closest to the argument *number1* and that is an integer. - If there are two such numbers, then the one that is closest to positive infinity is returned. - If the argument is not a number (NaN), then NaN is returned. - If the argument is positive infinity, then positive infinity is returned. - If the argument is negative infinity, then negative infinity is returned. - If the argument is positive zero, then positive zero is returned. - If the argument is negative zero, then negative zero is returned. - If the argument is less than zero, but greater than or equal to -0.5, then negative zero is returned.
	At run time, the **sum** method returns the sum, for each node in the argument *node-set1*, of the result of converting the string-values of the node to a number.

11.5.4 Operator Methods

Figure 352 Operator Methods Menu

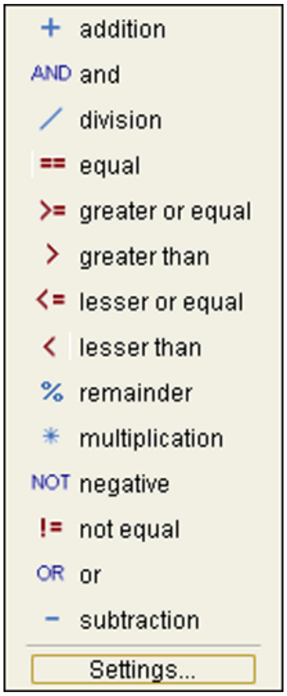


Table 94 Operator Collaboration Methods (XSLT)

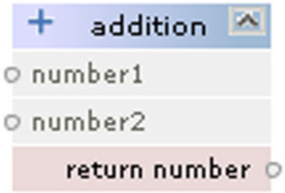
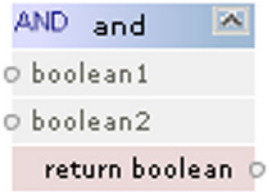
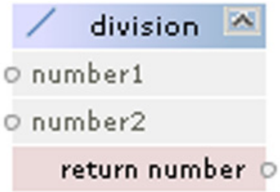
Method Box	Description/Usage
	At run time, the **addition** method adds the value of *number1* to the value of *number2*, returns the sum.
	At run time, the **and** method returns Boolean true if both *boolean1* and *boolean2* are true; otherwise, returns Boolean false.
	At run time, the **division** method divides the value of *number1* by the value of *number2*, returns the quotient.

Table 94 Operator Collaboration Methods (XSLT)

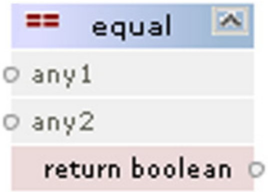
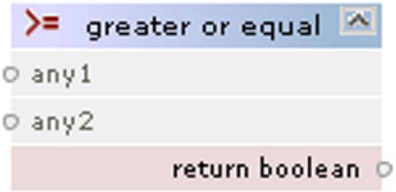
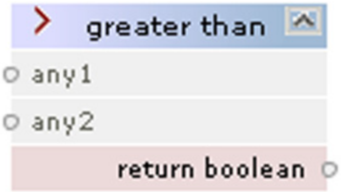
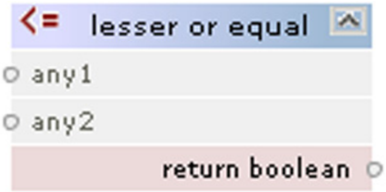
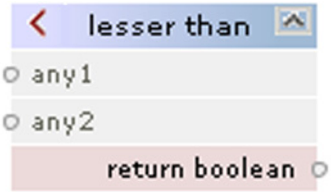
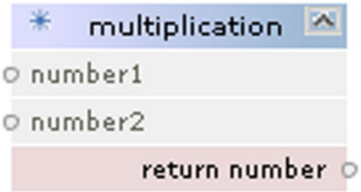
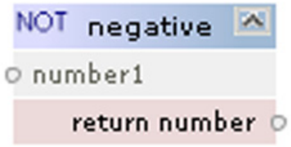
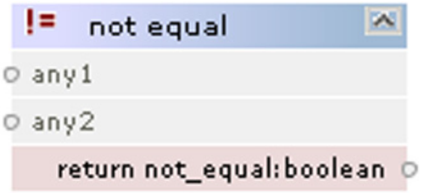
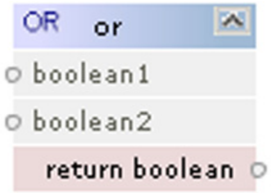
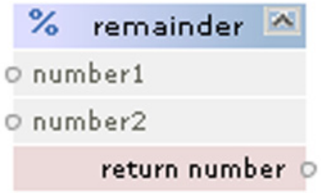
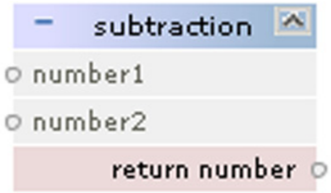
Method Box	Description/Usage
	At run time, the **equal** method returns Boolean true if *any1* is equal to *any2*; otherwise, returns Boolean false.
	At run time, the **greater_or_equal** method returns Boolean true if *any1* is greater than or equal to *any2*; otherwise, returns Boolean false.
	At run time, the **greater_than** method returns Boolean true if *any1* is greater than *any2*; otherwise, returns Boolean false.
	At run time, the **lesser_or_equal** method returns Boolean true if *any1* is less than or equal to *any2*; otherwise, returns Boolean false.
	At run time, the **lesser_than** method returns Boolean true if *any1* is less than *any2*; otherwise, returns Boolean false.
	At run time, the **multiplication** method multiplies the value of *number1* by the value of *number2*, returns the product.

Table 94 Operator Collaboration Methods (XSLT)

Method Box	Description/Usage
	At run time, the **negative** method returns the arithmetic negation of *number1*.
	At run time, the **not_equal** method returns Boolean true if *any1* is not equal to *any2*; otherwise, returns Boolean false.
	At run time, the **OR** method returns Boolean false if both *boolean1* and *boolean2* are false; otherwise, returns Boolean true.
	At run time, the **remainder** method divides the numerical value of *number1* by the numerical value of *number2*, and returns the remainder.
	At run time, the **subtraction** method subtracts the numerical value of *number2* from the numerical value of *number1*, returns the difference.

11.5.5 String Methods

Figure 353 String Methods Menu

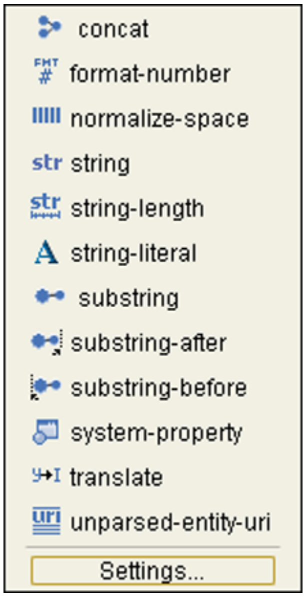


Table 95 String Collaboration Methods (XSLT)

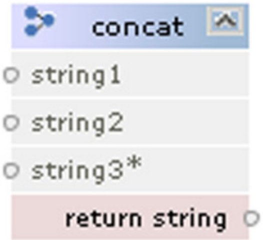
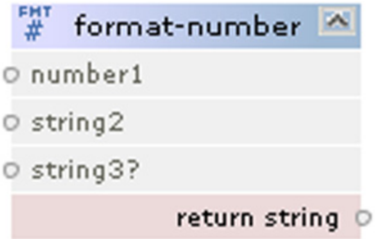
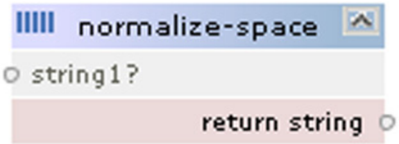
Method Box	Description/Usage
	At run time, the **concat** method returns the string created by concatenating *string2* to the end of *string1*, and *string3** (if present) to the end of *string2*.
 format-number number1 string2 string3? return string	At run time, the **format-number** method converts its first argument (*number1*) to a string using the format pattern string specified by the second argument (*string2*) and the decimal-format named by the third argument (*string3*) or the default decimal-format, if there is no third argument.
	At run time, the **normalize-space** method returns the argument *string1?* 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.

Table 95 String Collaboration Methods (XSLT)

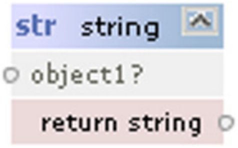
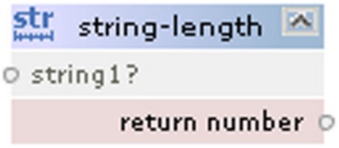
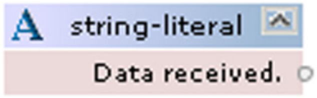
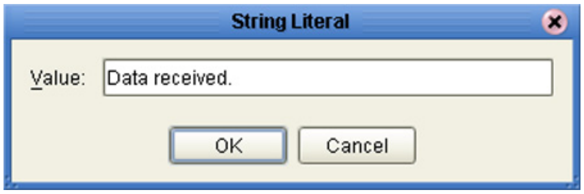
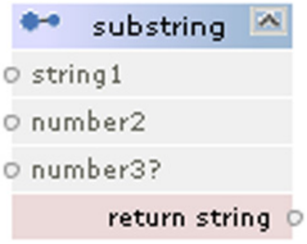
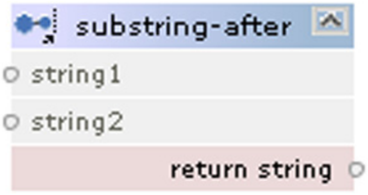
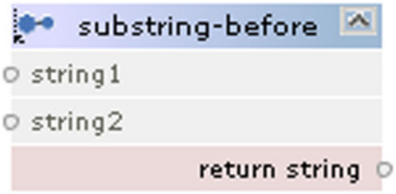
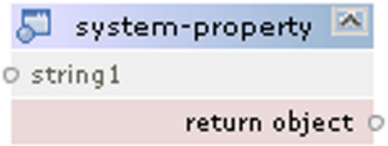
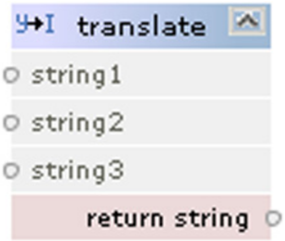
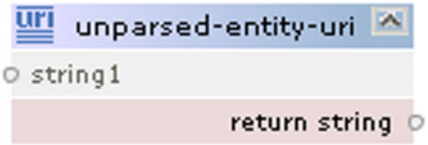
Method Box	Description/Usage
	At run time, the **string** method returns a string representation of the input object.
	At run time, the **string-length** method returns the number of characters in the string.
	Dragging the **string-literal** icon first presents a dialog in which you enter the literal value; for example, "Data received":  Value: Data received. At run time, the method returns a string having the specified value.
 substring string1 number2 number3? return string	At run time, the **substring** method returns the substring of the first argument (*string1*) starting at the position specified in the second argument (*number2*) with length specified in the third argument (*number3?*). 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 **substring-after** method returns the substring of the first argument (*string1*) that follows the first occurrence of the second argument (*string2*) 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 **substring-before** method returns the substring of the first argument (*string1*) that precedes the first occurrence of the second argument (*string2*) in the first argument string; returns an empty string if the first argument string does not contain the second argument string.

Table 95 String Collaboration Methods (XSLT)

Method Box	Description/Usage
	At run time, the **system-property** method returns an object representing the value of the system property identified by *string1*. If there is no such system property, the empty string should be returned.
	At run time, the **translate** method returns the first argument (*string1*) with occurrences of characters in the second argument (*string2*) replaced by the character at the corresponding position in the third argument (*string3*). If a character occurs more than once in the second argument (*string2*), then the first occurrence determines the replacement character. If the third argument (*string3*) is longer than the second argument (*string2*), then excess characters are ignored. Refer to the W3C *XML Path Language* documentation for additional conditions.
	At run time, the **unparsed-entity-uri** method returns the URI of the unparsed entity with the specified name (*string1*) in the same document as the context node. It returns an empty string if there is no such entity.

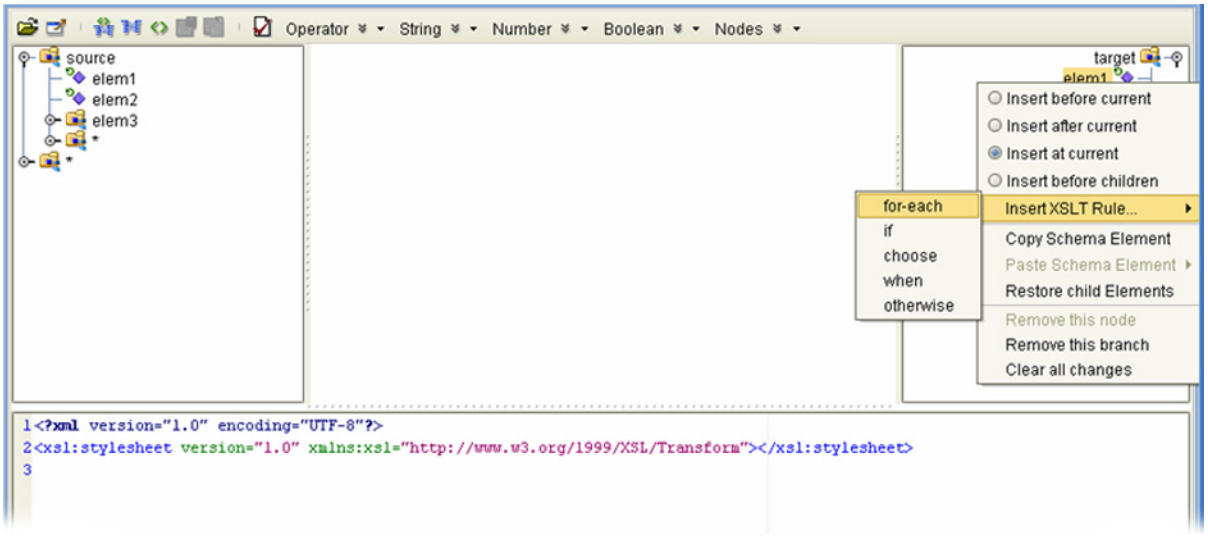
11.6 Design Procedures

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 354), and right-click to display the context menu.
3. Select **Insert XSLT Rule** from the menu, and then **for-each**.

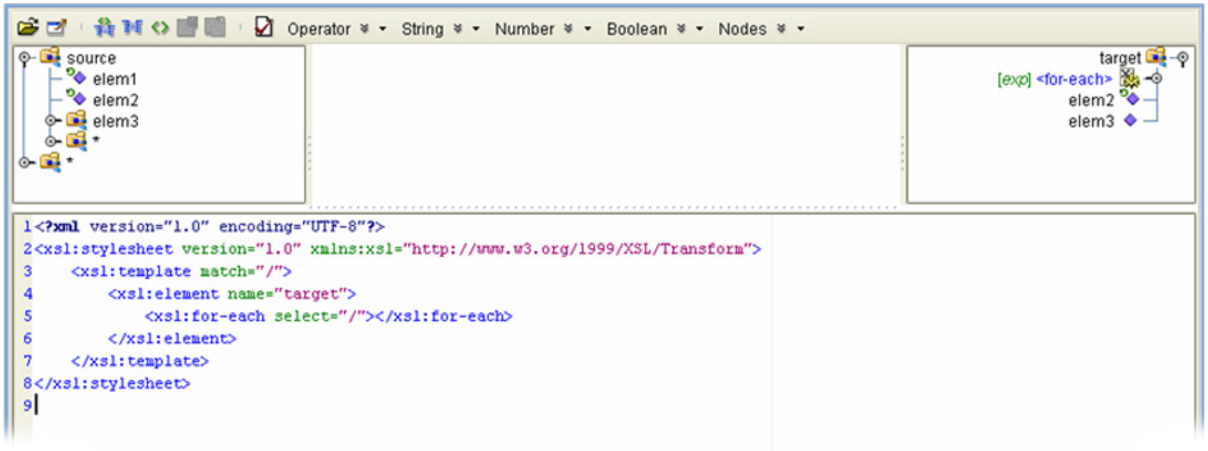
Figure 354 For-Each Construct - Step 1



```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"></xsl:stylesheet>
```

4. The **for-each** construct appears as shown in Figure 355.

Figure 355 For-Each Construct - Step 2



```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <xsl:element name="target">
            <xsl:for-each select="/"></xsl:for-each>
        </xsl:element>
    </xsl:template>
</xsl:stylesheet>
```

5 As shown in Figure 356:

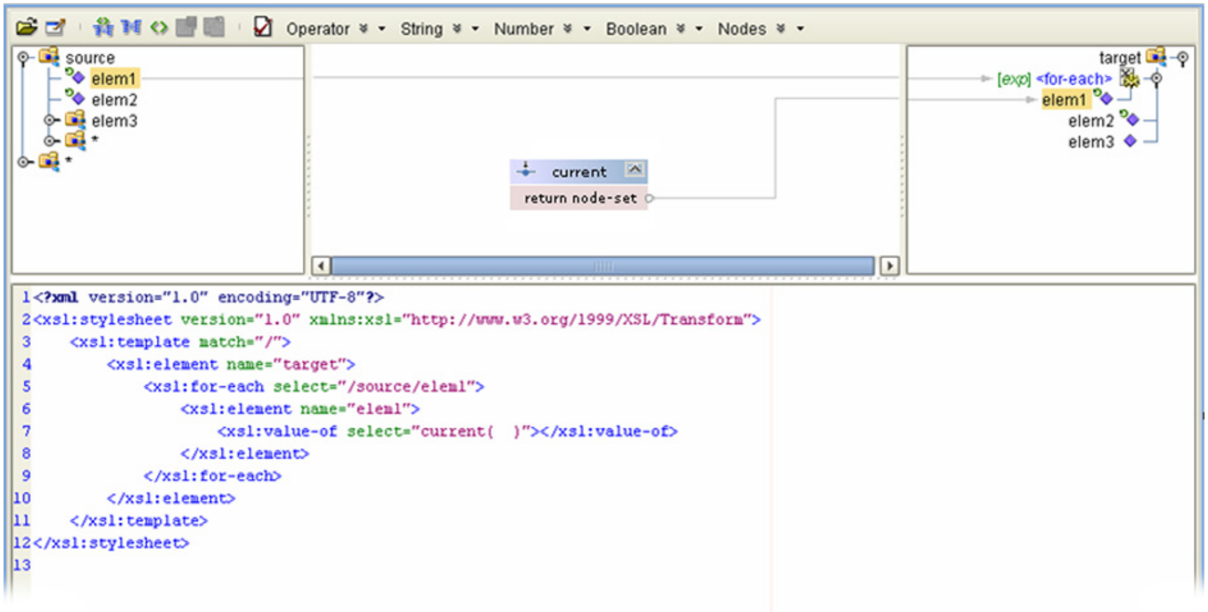
A Expand the node to display **elem1** as a sub-node.

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 356 For-Each Construct - Step 3



```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <xsl:element name="target">
            <xsl:for-each select="/source/elem1">
                <xsl:element name="elem1">
                    <xsl:value-of select="current(  )"></xsl:value-of>
                </xsl:element>
            </xsl:for-each>
        </xsl:element>
    </xsl:template>
</xsl:stylesheet>
```

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>
```

Chapter 12

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 418
- **Developing Web Services** on page 422
- **Creating XML Schema Definitions** on page 427
- **Creating Web Service Definitions** on page 475
- **Configuring WSDL Binding Properties** on page 504

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 475.

- **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 427.

- **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 504).

- **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 426.

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 96 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 WSD Object.
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.
WSD Object	Created using the Web Service Definition Editor or by importing an external WSDL document. 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 424).

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 97 Password Types for User Name Token

Type	Description
Text	The password is clear text or a password equivalent. A *clear text* password is the original, unmodified password. A *password equivalent* 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 *nonce* 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">
            jg3EkQw5Ko924CZl1wEMk0SaFL87MkJS==
        </wsse:Password>
        <wsse:Nonce>lT7eQbk04dcFo2zFVp74==</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 422
- **Publishing a Web Service Project** on page 424

The the resulting web service can then be accessed via a UDDI Registry, as described in:

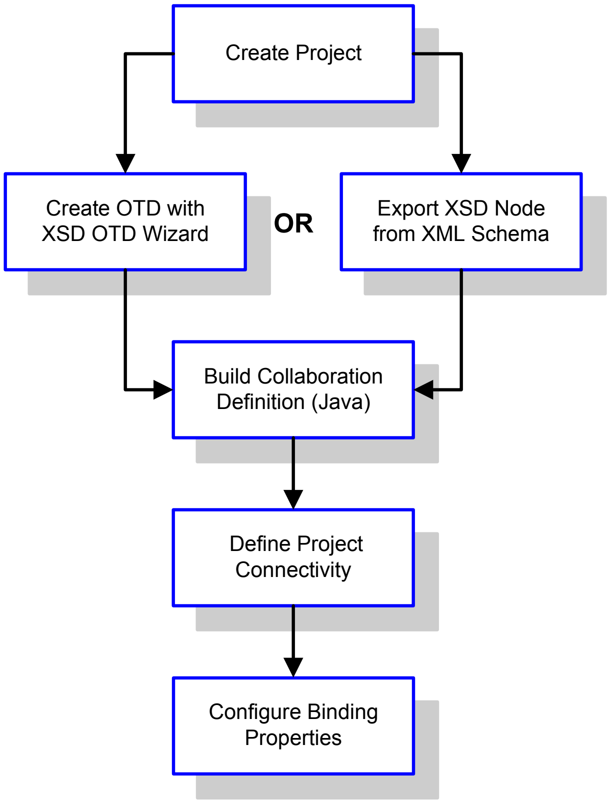
- **Accessing Web Service Definitions** on page 426

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 357.

Figure 357 Development Flow - Creating a Web Service Project


Create Project
Create OTD with XSD OTD Wizard
OR
Export XSD Node from XML Schema
Build Collaboration Definition (Java)
Define Project Connectivity
Configure Binding Properties

Procedure

To create a web service client or server

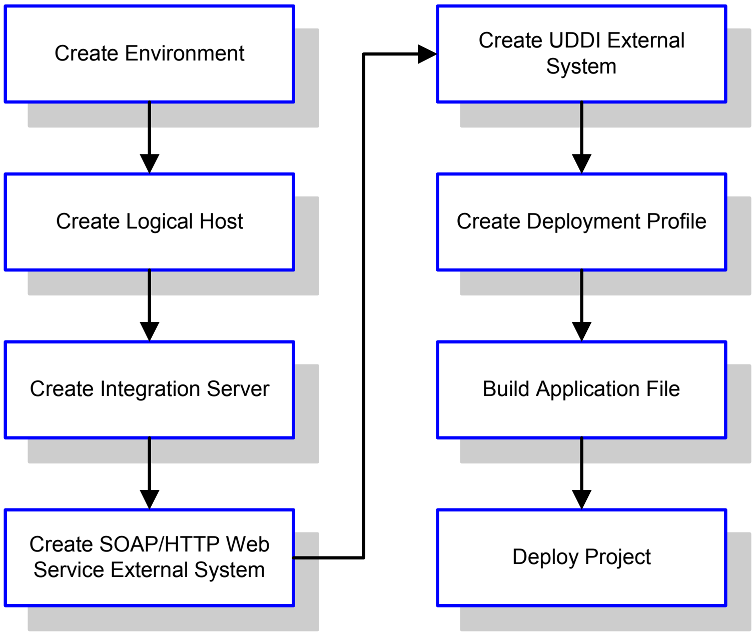
1. Create the Project (see **Creating a Project** on page 115).
2. Build an OTD based on an existing XSD file, using the XSD OTD Wizard (see **Using the XSD Wizard** on page 190), or build an XSD using the XML Schema Editor. and export an XSD Node (see **Using the XML Schema Editor** on page 428).
3. Develop a Java-based Collaboration Definition, using the XSD OTD or XSD Node to provide the input and output messages (see **Creating a Java-based Collaboration Definition** on page 326).
   - 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 129).
5. Modify the default configuration for the transport binding properties, if necessary (see **Configuring WSDL Binding Properties** on page 504).

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 426). The associated WSDL files can be published to the UDDI registry directly from Enterprise Designer as described below.

Figure 358 Development Flow - Publishing a Web Service Definition


Create Environment
Create Logical Host
Create Integration Server
Create SOAP/HTTP Web Service External System
Create UDDI External System
Create Deployment Profile
Build Application File
Deploy Project

Procedure

Note: *The following procedures assume 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 526).

   B. A SOAP/HTTP Web Service External System (see **SOAP/HTTP Web Service External Systems** on page 536).

   C. A UDDI External System (see **UDDI External Systems** on page 550).

2 Create a Deployment Profile for the Project you created in the previous section (see **Creating a Deployment Profile** on page 140).

3 Generate the Project application file (see **Building an Application File** on page 148) 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 154).

B Exporting the archive file to a holding directory for publishing from the command line at a later time (see following procedure).

Important: *Each of a web services Project's active Deployment Profiles must target a separate SOAP/HTTP External System; otherwise, the web containers can cause duplicate servlet names to appear in the domain.*

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.

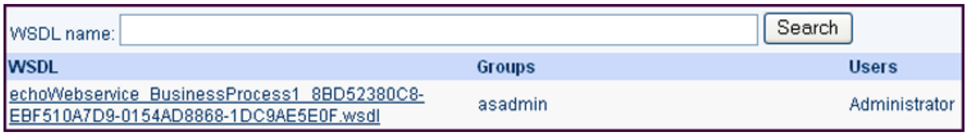
To access a web service from the UDDI registry, the following components must be installed and set up:

- Sun SeeBeyond UDDI Server (see **UDDI External Systems** on page 550).
- 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 359 Web Services Access Manager

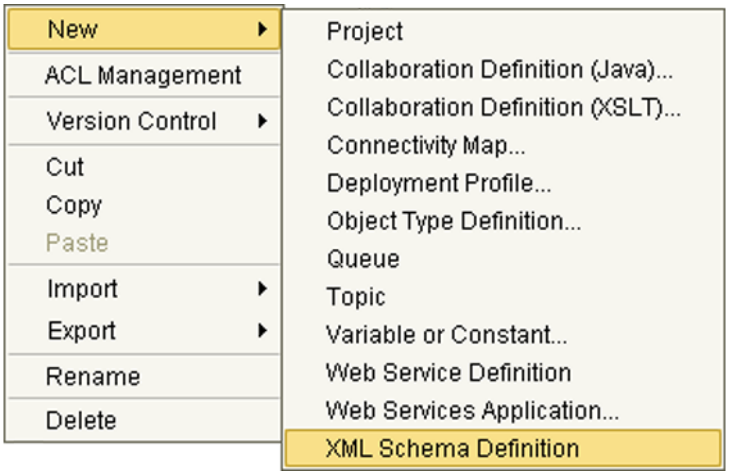


WSDL	Groups	Users
echoWebservice_BusinessProcess1_8BD52380C8-EBF510A7D9-0154AD8868-1DC9AE5E0F.wsdl	asadmin	Administrator

12.3 Creating XML Schema Definitions

Selecting the **New > XML Schema Definition** option from the *Project* context menu (see Figure 360) adds an XSD Object to your Project and opens the XML Schema Editor (see **Using the XML Schema Editor** on page 428).

Figure 360 Project Context Menu - New XML Schema Definition



12.3.1 Using the Context Menu

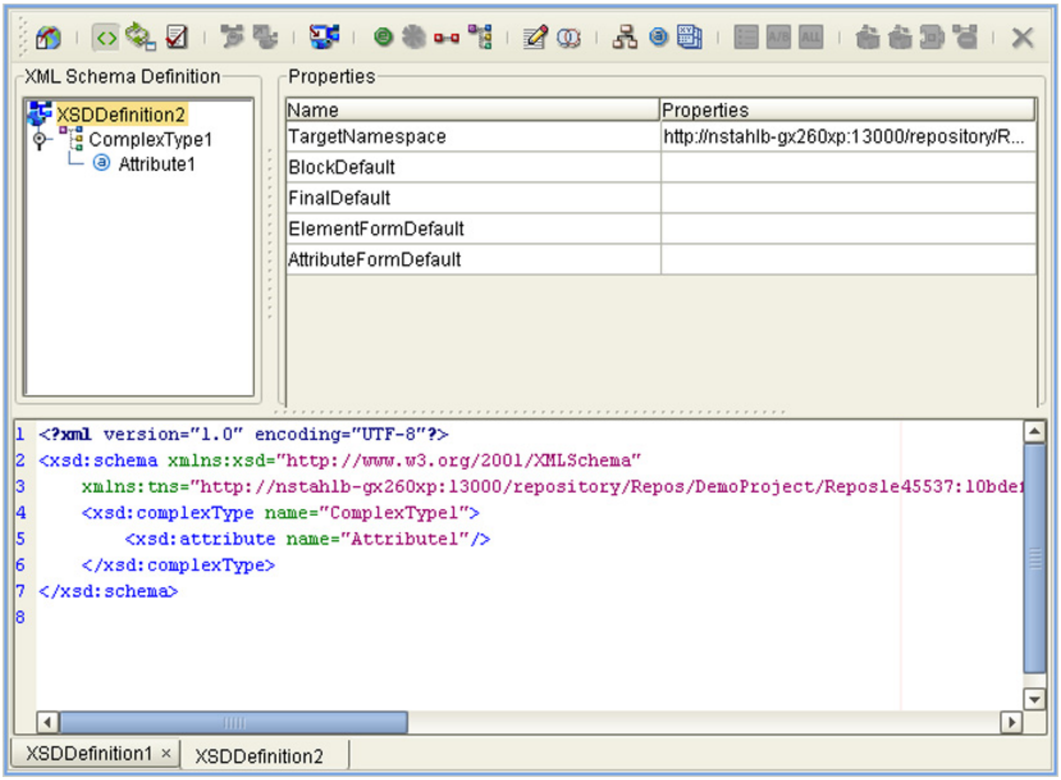
Right-clicking the XSD Object in Project Explorer displays its context menu (see **Using Project Component Context Menus** on page 113).

12.4 Using the XML Schema Editor

12.4.1 Overview

The XML Schema Editor (see Figure 361) 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 361 XML Schema Editor



Name	Properties
TargetNamespace	http://nstahlb-gx260xp:13000/repository/R...
BlockDefault	
FinalDefault	
ElementFormDefault	
AttributeFormDefault	

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://nstahlb-gx260xp:13000/repository/Repos/DemoProject/Repos1e45537:10bde
    <xsd:complexType name="ComplexType1">
        <xsd:attribute name="Attribute1"/>
    </xsd:complexType>
</xsd:schema>
```

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 430 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 437.

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 437.

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 430).

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 430.

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 98 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 **.xsd** 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 **OK** 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 **Previewing and Exporting XSD Nodes** on page 433.
	Export XSD Node	Presents a dialog containing allowed options; clicking **OK** in the dialog exports the selected top-level Element, ComplexType, or SimpleType as an *XSD Node*, which can be used in a Collaboration Definition (Java). See **Previewing and Exporting XSD Nodes** on page 433.
	Import XSD	Opens a dialog (see **Figure 367 on page 440**) in which you can import an XSD file to the currently open XSD (duplicates the action of the **Add Import** option on the context menu for the root node).
ⓔ	Add Element	Adds an **xsd:element** sub-node to the selected XSD root, XML sequence, XML choice, or XML all node (duplicates the action of the **Add Element** option on the context menus for those nodes). See **Element Node** on page 446.

Table 98 XML Schema Editor Toolbar Icons

Icon	Command	Function
✱	Add Any	Adds an XML **any** 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 **Add Any** option on the context menus for those nodes). See **Any Node** on page 464.
□–□	Add SimpleType	Adds an XML **simple type** sub-node to the selected XSD root node (duplicates the action of the **Add Simple Type** option on the XSD root node context menu). See **SimpleType Node** on page 455.
▦	Add ComplexType	Adds an XML **complex type** sub-node to the selected XSD root node (duplicates the action of the **Add ComplexType** option on the XSD root node context menu). See **ComplexType Node** on page 453.
✎	Add ListType	Adds an XML **listType** sub-node to the selected XSD root node (duplicates the action of the **Add ListType** option on the context menus for those nodes). See **ListType Node** on page 460.
⚭	Add Union	Adds an XML **union** sub-node to the selected XSD root node (duplicates the action of the **Add Union** option on the context menus for those nodes). See **Union Node** on page 450.
品	Add Group	Adds an XML **group** sub-node to the selected XSD root node (duplicates the action of the **Add Group** option on the XSD root node context menu). See **Group Node** on page 452.
ⓐ	Add Attribute	Adds an XML **attribute** sub-node to the selected XSD root, XML complex type, or XML attribute group node (duplicates the action of the **Add Attribute** option on the context menus for those nodes). See **Attribute Node** on page 442.
XY	Add AttributeGroup	Adds an XML **attributeGroup** sub-node to the selected XSD root node (duplicates the action of the **Add AttributeGroup** option on the context menus for those nodes). See **AttributeGroup Node** on page 444.
☰	Add Sequence	Adds an XML **sequence** sub-node to the selected XML complex type, XML group, or XML restriction node (duplicates the action of the **Add Sequence** option on the context menus for those nodes). See **Sequence Node** on page 461.
A/B	Add Choice	Adds an XML **choice** sub-node to the selected XML complex type, XML group, or XML restriction node (duplicates the action of the **Add Choice** option on the context menus for those nodes). See **Choice Node** on page 462.

Table 98 XML Schema Editor Toolbar Icons

Icon	Command	Function
ALL	Add All	Adds an XML **all** sub-node to the selected XML complex type, XML group, or XML restriction node (duplicates the action of the **Add All** option on the context menus for those nodes). See **All Node** on page 463.
	Add SimpleContent	Adds an XML **simpleContent** sub-node to the selected XML complexType node (duplicates the action of the **Add SimpleContent** option on the context menus for those nodes). See **SimpleContent Node** on page 467.
	Add ComplexContent	Adds an XML **complexContent** sub-node to the selected XSD complexType node (duplicates the action of the **Add ComplexContent** option on the context menus for those nodes). See **ComplexContent Node** on page 466.
	Add Extension	Adds an XML **extension** sub-node to the selected XSD simpleContent or XML complexContent node (duplicates the action of the **Add Extension** option on the context menus for those nodes). See **Extension Node** on page 468.
⊖	Add Restriction	Adds an XML **restriction** sub-node to the selected XSD complexContent node (duplicates the action of the **Add Restriction** option on the context menus for those nodes). See **Restriction Node** on page 469.
✕	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.*

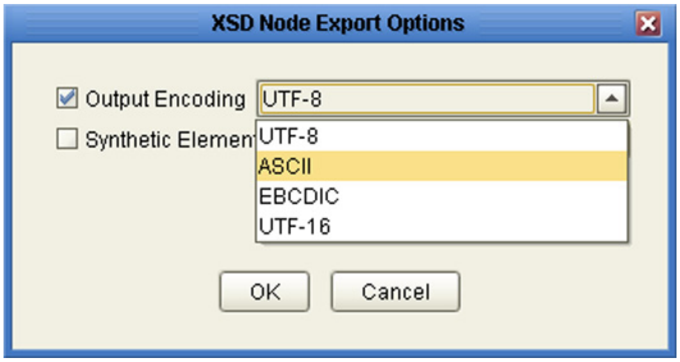
Table 99 XSD Node Preview/Export Options

Name	Description
Output Encoding	Specifies the output encoding. See **Specifying Data Conversion and Encoding** on page 160. Option is enabled only when the extended language option is enabled (see **Setting Up Options** on page 55).
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 55), 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 362), where you can select an alternate data encoding.

Figure 362 XSD Node Preview/Export Options - Output Encoding


XSD Node Export Options

☑ Output Encoding UTF-8

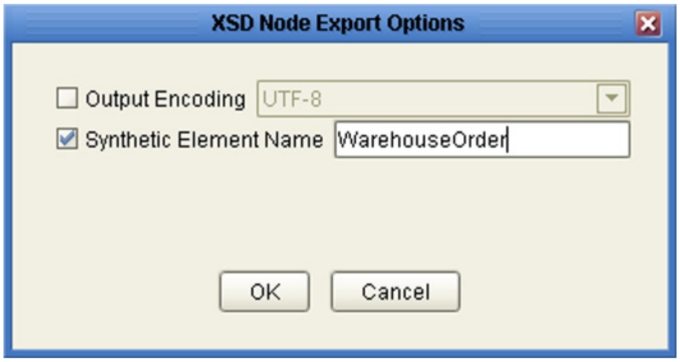
□ Synthetic Elemen

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 363). (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 363 XSD Node Preview/Export Options - Synthetic Element Name


XSD Node Export Options

□ Output Encoding UTF-8

☑ Synthetic Element Name WarehouseOrder

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 57) 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 100 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 **ide.log** 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 **ide.log** 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 101 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 *not* 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.

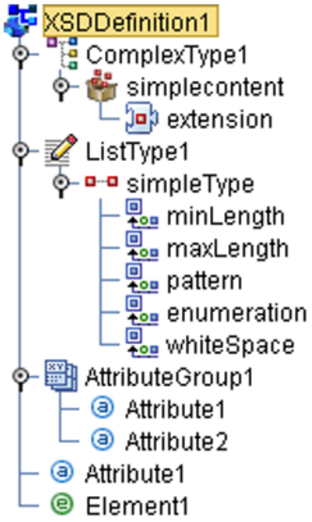
Table 102 Enforce Particle Restriction Rule Options

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 *not* 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 364 shows an XSD tree structure illustrating some basic node combinations.

Figure 364 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 366. No properties are shown for the root node.

Figure 365 XSD Root Node Icon

Context Menu

Figure 366 XSD Root Node Menu

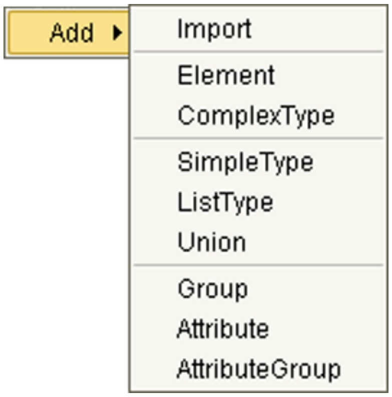


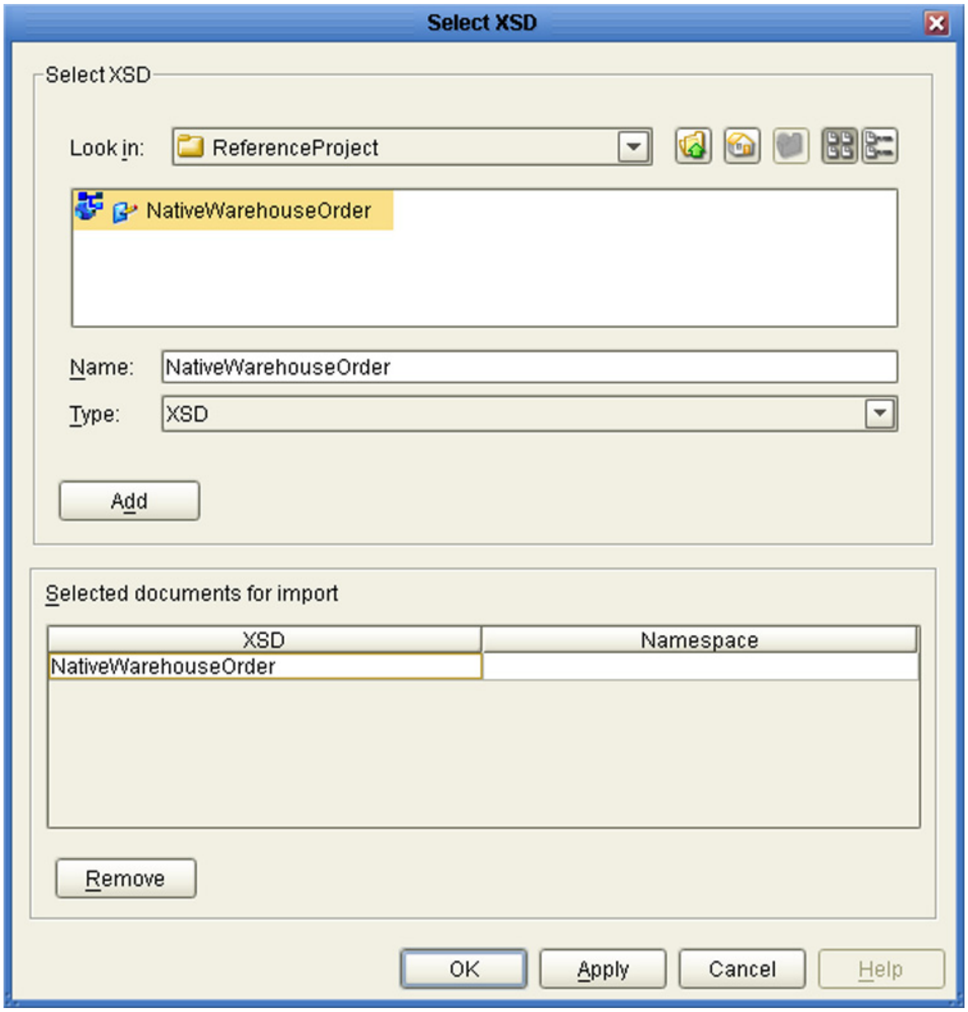
Table 103 XSD Root Node Menu Options

Option		Action
Add	Import	Opens a dialog (see Figure 367) in which you can import an XSD file as a sub-node to selected XSD root node (duplicates the action of the **Import XSD** toolbar icon).
	Element	Adds an **Element** sub-node to the selected XSD root node (duplicates the action of the **Add Element** toolbar icon). An element at this level becomes a *global* element. See **Element Node** on page 446.
	ComplexType	Adds an **ComplexType** sub-node to the selected XSD root node (duplicates the action of the **Add ComplexType** toolbar icon). See **ComplexType Node** on page 453.
	SimpleType	Adds an **SimpleType** sub-node to the selected XSD root node (duplicates the action of the **Add SimpleType toolbar** icon). See **SimpleType Node** on page 455.

Table 103 XSD Root Node Menu Options

Option		Action
Add	ListType	Adds an **ListType** sub-node to the selected XSD root node. See **ListType Node** on page 460.
	Union	Adds an **Union** sub-node to the selected XSD root node. See **Union Node** on page 450.
	Group	Adds an **Group** sub-node to the selected XSD root node (duplicates the action of the **Add Group** toolbar icon). See **Group Node** on page 452.
	Attribute	Adds an **Attribute** sub-node to the selected XSD root node (duplicates the action of the **Add Attribute** toolbar icon). An attribute at this level becomes a *global* attribute. See **Attribute Node** on page 442.
	Attribute Group	Adds an **AttributeGroup** sub-node to the selected XSD root node. See **AttributeGroup Node** on page 444.

Figure 367 Select XSD Dialog



Select XSD

Select XSD

Look in: ReferenceProject

NativeWarehouseOrder

Name: NativeWarehouseOrder

Type: XSD

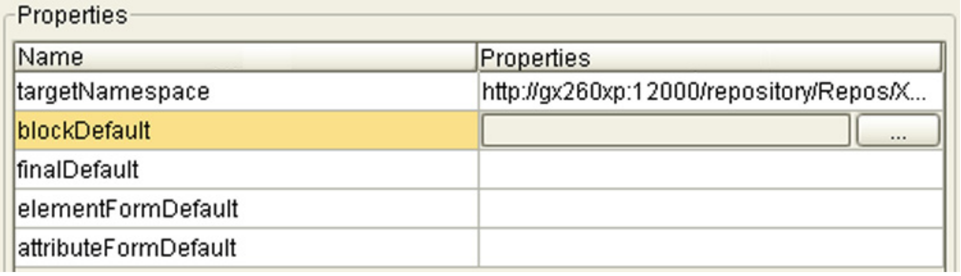
Selected documents for import

XSD	Namespace
NativeWarehouseOrder	

Properties

The set of properties associated with the XSD root node is shown in Figure 370.

Figure 368 XSD Root Node Properties



Properties

Name	Properties
targetNamespace	http://gx260xp:12000/repository/Repos/X...
blockDefault	...
finalDefault	
elementFormDefault	
attributeFormDefault	

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 104 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 **qualified** or **unqualified**.
attributeFormDefault	Specifies whether the attributeFormDefault is **qualified** or **unqualified**.

Attribute Node

The **Attribute** node corresponds to the **<xsd:attribute>** element, which represents an attribute declaration.

Figure 369 Attribute Node Icon

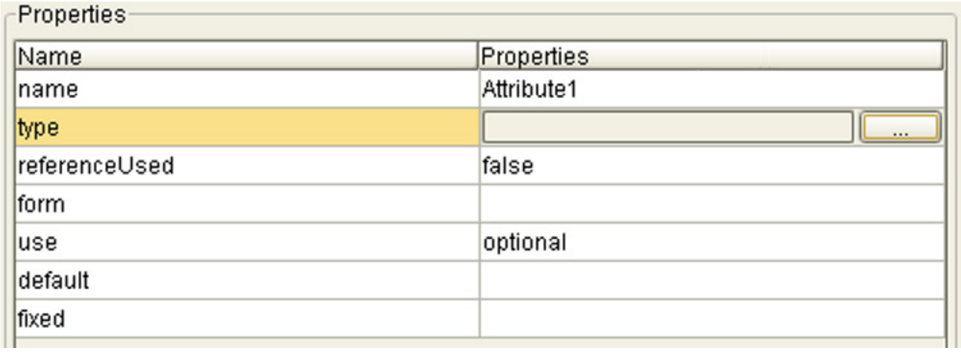
Context Menu

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 370; global *Attribute* nodes have a subset of these properties associated with them, as described in Table 105.

Figure 370 Attribute Node Properties



Properties

Name	Properties
name	Attribute1
type	...
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 105 Attribute Node Property Options

Name	Description
name	Name of the **Attribute** node.
type	Instead of assigning a new name to this node, you can reference an existing attribute name. If the **referenceUsed** property is set to **true**, 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 **name** property are mutually exclusive.

Name	Description
referenceUsed	Specifies whether or not this element is named by reference. Setting this property to **true** enables the **type** property value field (see below). **Note:** Does not apply to global Attribute nodes.
form	Specifies whether or not this (local) attribute must be **qualified** or **unqualified** (or neither). **Note:** Does not apply to global Attribute nodes.
use	Specifies whether the node is optional, required, or prohibited. ▪ **optional** specifies that the **<xsd:attribute>** element is optional and may have any value. ▪ **required** specifies that the **<xsd:attribute>** element is required and may have any value. ▪ **prohibited** specifies that the **<xsd:attribute>** element must not appear. **Note:** Does not apply to global Attribute nodes.
default	If the **use** property is specified to be **optional**, the default value applies when **<xsd:attribute**> is missing from the instance document; otherwise, the value in the instance document applies. The **default** and **fixed** properties are mutually exclusive — you can declare values for one or the other, not both.
fixed	If the **use** property is specified to be **optional**, the fixed value is always used for **<xsd:attribute**> whether or not it is missing from the instance document. The **default** and **fixed** 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 371 Attribute Group Node Icon

Context Menu

Figure 372 AttributeGroup Node Menu

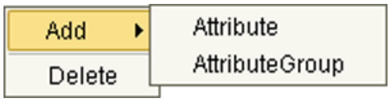


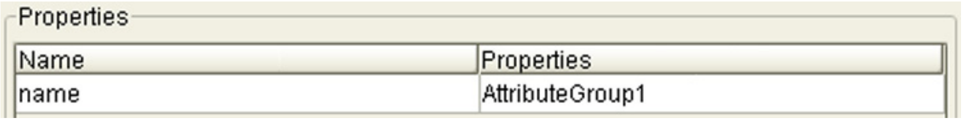
Table 106 AttributeGroup Node Menu Options

Option		Action
Add	Attribute	Adds an **Attribute** sub-node to the selected AttributeGroup node (duplicates the action of the **Add Attribute** toolbar icon). See **Attribute Node** on page 442.
	Attribute Group	Adds an **AttributeGroup** sub-node to the selected ComplexType node. See **AttributeGroup Node** on page 444.
Delete		Deletes the selected node from the XSD tree, including all sub-nodes.

Properties

The properties associated with *AttributeGroup* are shown in Figure 373 and Figure 374.

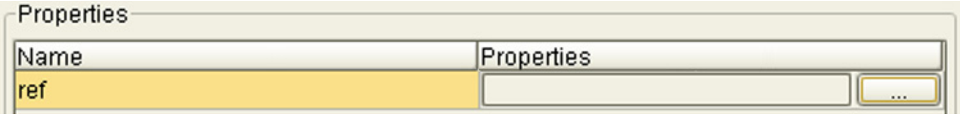
Figure 373 AttributeGroup Node Properties (Global)



Properties

Name	Properties
name	AttributeGroup1

Figure 374 AttributeGroup Node Properties (Non-global)



Properties

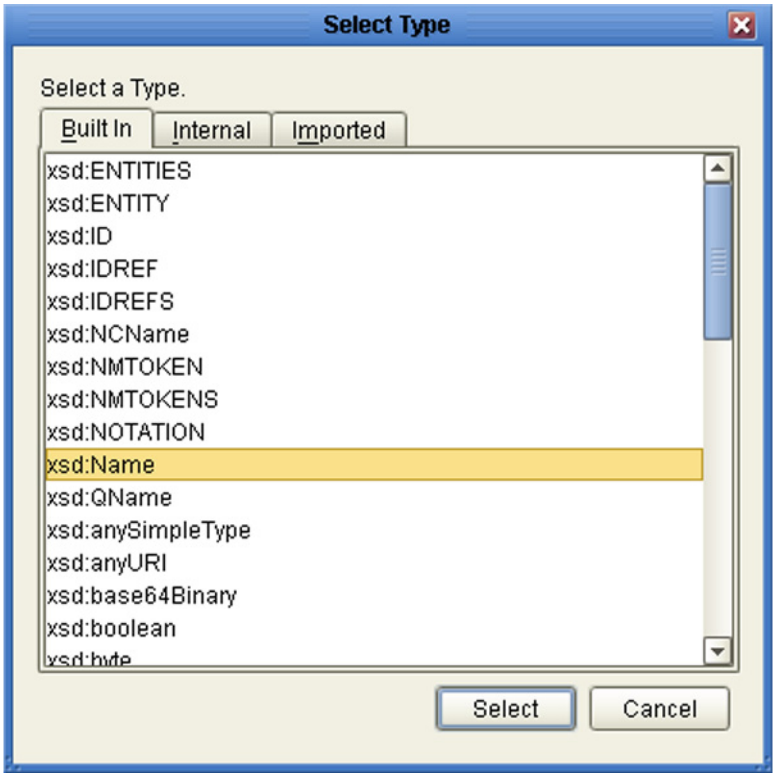
Name	Properties
ref	...

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 107 AttributeGroup Node Property Options

Name	Description
name	Name of the **AttrbuteGroup** node. Parameter is shown only for global AttributeGroups.
ref	If the **AttrbuteGroup** node is non-global, it can only reference an existing, global attrbuteGroup. Selecting the field and clicking the ellipsis (**...**) button displays a dialog containing a list of existing internally-defined types and imported types. (see Figure 375). Parameter is shown only for non-global AttributeGroups.

Figure 375 Select Type Dialog



Element Node

The **Element** node corresponds to the **<xsd:element>** element, which represents an element declaration.

Figure 376 Element Node Icon

Context Menu

Figure 377 Element Node Menu

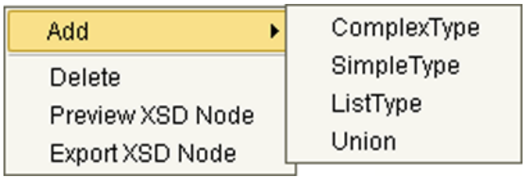


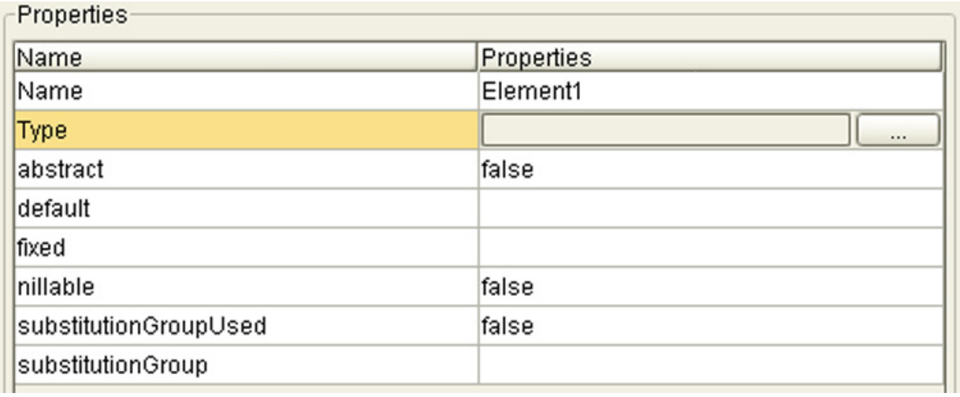
Table 108 Element Node Menu Options

Option		Action
Add	ComplexType	Adds an **ComplexType** sub-node to the selected element node (duplicates the action of the **Add ComplexType toolbar** icon). See **ComplexType Node** on page 453.
	SimpleType	Adds an **SimpleType** sub-node to the selected element node (duplicates the action of the **Add SimpleType toolbar** icon). See **SimpleType Node** on page 455.
	ListType	Adds an **ListType** sub-node to the selected element node. See **ListType Node** on page 460.
	Union	Adds an **Union** sub-node to the selected element node. See **Union Node** on page 450.
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 *Window* 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 378 (global) and Figure 379 (non-global).

Figure 378 Element Node Properties (Global)



Properties

Name	Properties
Name	Element1
Type	...
abstract	false
default	
fixed	
nillable	false
substitutionGroupUsed	false
substitutionGroup	

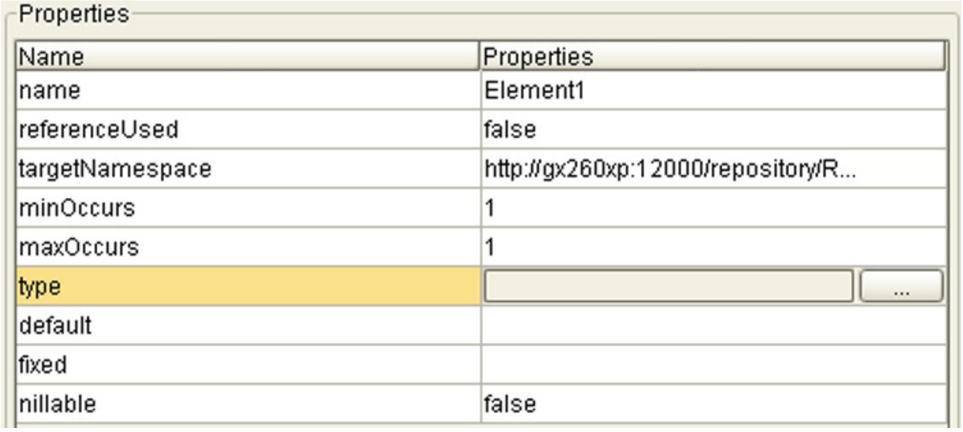
Note: *Clicking the Value field for a property enables it, after which you can edit the text. For exceptions to this rule, see the following table.*

Table 109 Element Node Property Options (Global)

Name	Description
Name	Name of the **Element** 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 **Element** node is empty in the instance document. **Note:** The **default** and **fixed** properties are mutually exclusive — you can declare values for one or the other, not both.
fixed	If a fixed value is given, the **Element** node is defined as optional, and the value is always used for the node whether or not it appears in the instance document. **Note:** The **default** and **fixed** properties are mutually exclusive — you can declare values for one or the other, not both.
nillable	Setting this property to **true** 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 **true** enables the **substitutionGroup** property value field (see below).

Name	Description
substitutionGroup	If the **substitutionGroupUsed** property is set to **true**, selecting the field and clicking the ellipsis (**...**) button displays a dialog containing a lists of elements for which this element can be substituted.

Figure 379 Element Node Properties (Non-global)



Properties

Name	Properties
name	Element1
referenceUsed	false
targetNamespace	http://gx260xp:12000/repository/R...
minOccurs	1
maxOccurs	1
type	...
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 110 Element Node Property Options

Name	Description
name	Name of the **Element** node.
referenceUsed	Specifies whether or not this element is named by reference. Setting this property to **true** enables the **type** 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 **1**.
maxOccurs	Specifies the maximum number of times this element may appear; the default value is **1**.

Name	Description
type	Instead of assigning a new name to this node, you can reference an existing element name. If the **referenceUsed** property is set to **true**, selecting the field and clicking the button displays a dialog containing a list of existing elements within the target namespace. **Note:** This property and the **name** property are mutually exclusive.
default	If a default value is given, the value is used when the **Element** node is empty in the instance document. **Note:** The **default** and **fixed** properties are mutually exclusive — you can declare values for one or the other, not both.
fixed	If a fixed value is given, the **Element** node is defined as optional, and the value is always used for the node whether or not it appears in the instance document. **Note:** The **default** and **fixed** properties are mutually exclusive — you can declare values for one or the other, not both.
nillable	Setting this property to **true** 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 380 Union Node Icon

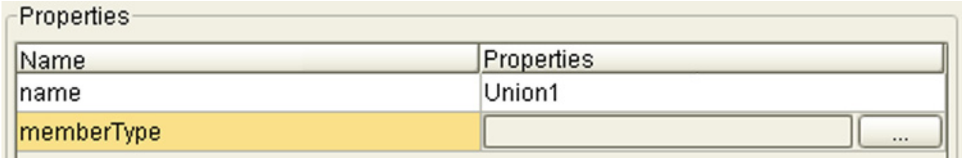
Context Menu

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 381.

Figure 381 Union Node Properties



Properties

Name	Properties
name	Union1
memberType	...

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 382.*

Table 111 Union Node Property Options

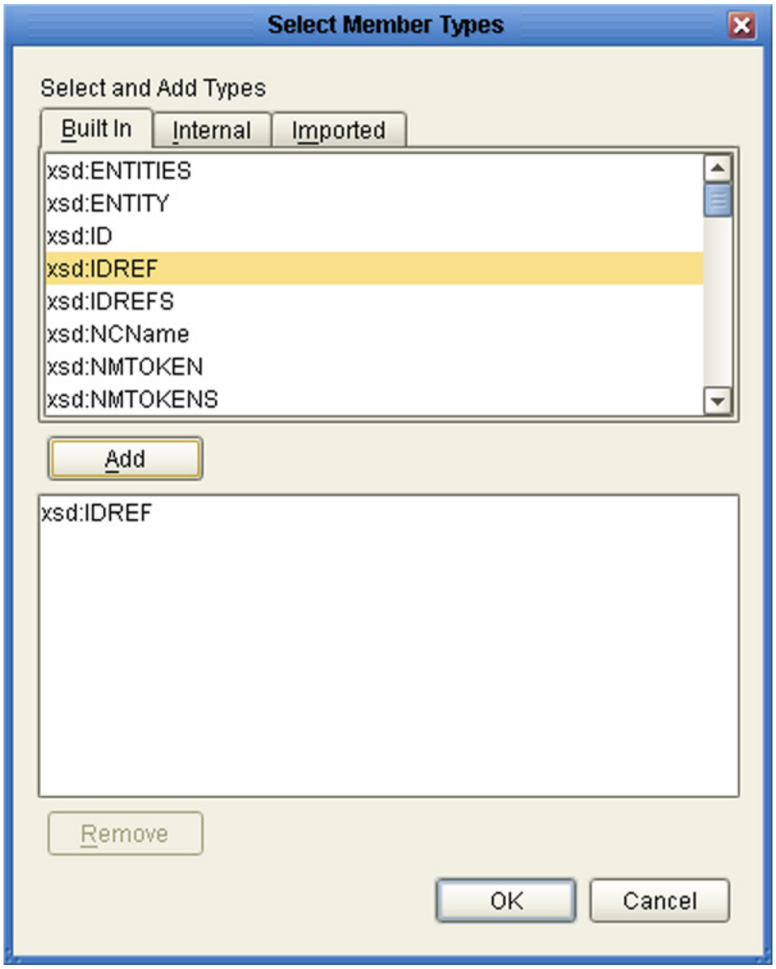
Name	Description
name	For global nodes only, name of the **Union** node.
memberType	A list of all types in the union. Double-clicking the value field and clicking the ellipsis (**...**) button displays a dialog containing a list of existing simple types, from which you can build a list for the union.

The dialog shown in Figure 382 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 382 Select Member Types Dialog



xsd:IDREF

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 383 Group Node Icon

Context Menu

Figure 384 Group Node Menu

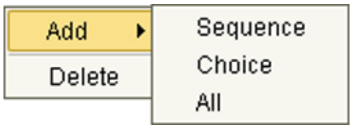


Table 112 Group Node Menu Options

Option		Action
Add	Sequence	Adds a **Sequence** sub-node to the selected Group node (duplicates the action of the **Add Sequence** toolbar icon). See **Sequence Node** on page 461.
	Choice	Adds a **Choice** sub-node to the selected Group node (duplicates the action of the **Add Choice** toolbar icon). See **Choice Node** on page 462.
	All	Adds an **All** sub-node to the selected Group node (duplicates the action of the **Add All** toolbar icon). See **All Node** on page 463.
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 385 ComplexType Node Icon

Context Menu

Figure 386 ComplexType Node Menu

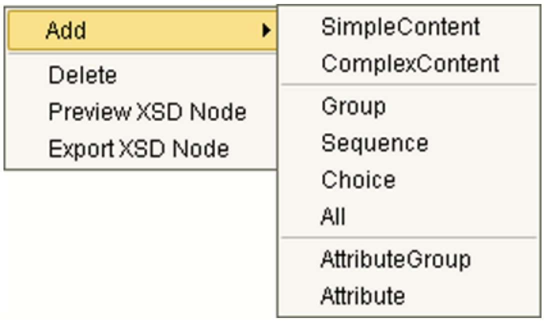


Table 113 ComplexType Node Menu Options

Option		Action
Add	Simple Content	Adds a **SimpleContent** sub-node to the selected ComplexType node. See **SimpleContent Node** on page 467.
	Complex Content	Adds a **ComplexContent** sub-node to the selected ComplexType node. See **ComplexContent Node** on page 466.
	Group	Adds a **Group** sub-node to the selected ComplexType node (duplicates the action of the **Add Group** toolbar icon). See **Group Node** on page 452.
	Sequence	Adds a **Sequence** sub-node to the selected ComplexType node (duplicates the action of the **Add Sequence** toolbar icon). See **Sequence Node** on page 461.
	Choice	Adds a **Choice** sub-node to the selected ComplexType node (duplicates the action of the **Add Choice** toolbar icon). See **Choice Node** on page 462.
	All	Adds an **All** sub-node to the selected ComplexType node (duplicates the action of the **Add All** toolbar icon). See **All Node** on page 463.

Table 113 ComplexType Node Menu Options

Option		Action
Add	Attribute Group	Adds an **AttributeGroup** sub-node to the selected ComplexType node. See **AttributeGroup Node** on page 444.
	Attribute	Adds an **Attribute** sub-node to the selected ComplexType node (duplicates the action of the **Add Attribute** toolbar icon). See **Attribute Node** on page 442.
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 *Window* 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 387 SimpleType Node Icon

Context Menu

Figure 388 SimpleType Node Menu

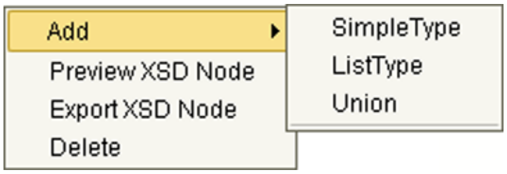


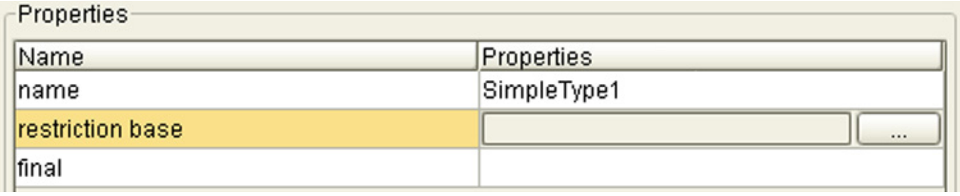
Table 114 SimpleType Node Menu Options

Option		Action
Add	SimpleType	Adds an **SimpleType** sub-node to the selected node (duplicates the action of the **Add SimpleType**toolbar icon). The sub-node then restricts the parent SimpleType node.
	ListType	Adds an **ListType** sub-node to the selected element node. See **ListType Node** on page 460.
	Union	Adds an **Union** sub-node to the selected element node. See **Union Node** on page 450.
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 *Window* 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 389.

Figure 389 SimpleType Node Properties



Properties

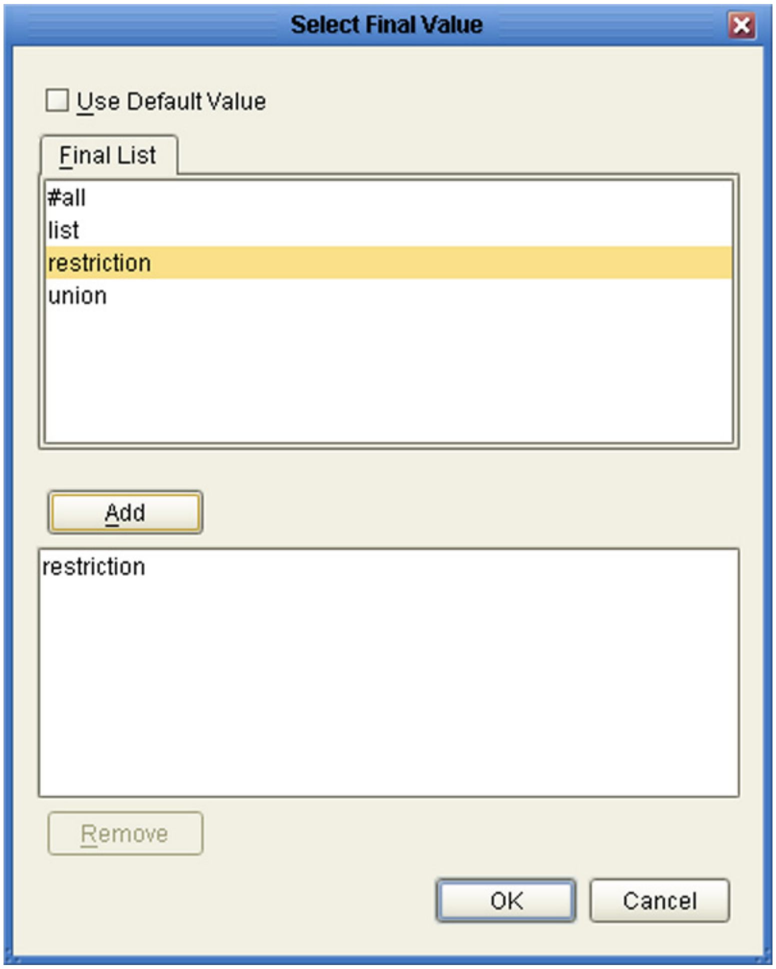
Name	Properties
name	SimpleType1
restriction base	...
final	

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 115 SimpleType Node Property Options

Name	Description
name	For global nodes only, name of the **SimpleType** 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 390).

Figure 390 Select Final Value Dialog



☐ Use Default Value

restriction

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 391 SimpleType Subnode Menu

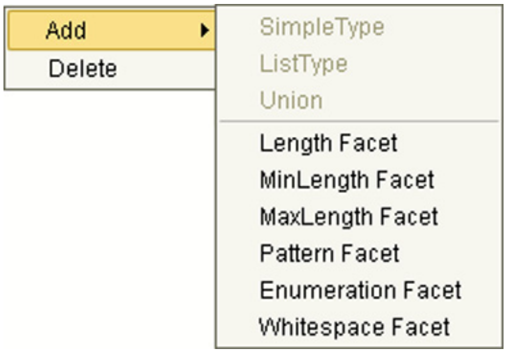


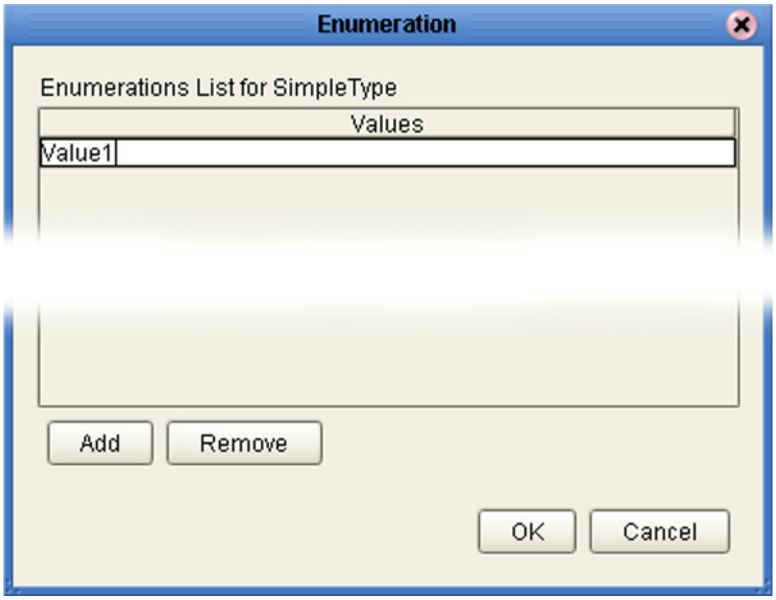
Table 116 SimpleType Subnode Menu Options

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 392).

Table 116 SimpleType Subnode Menu Options

Option		Action
Add	Whitespace Facet	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. ▪ **preserve** specifies that no normalization is to be performed, thereby defining the normalized value to be the initial value. ▪ **replace** specifies that all occurrences of **tab** (#x9), **line feed** (#xA), and **carriage return** (#xD) be replaced by a **space** (#x20). ▪ **collapse** specifies that a **replace** be performed, following which contiguous sequences of spaces be collapsed to a single space, and initial and/or final spaces be deleted.
Delete		Deletes the selected node from the XSD tree, including all sub-nodes.

Figure 392 Enumeration Dialog



In the dialog shown in Figure 392, 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 115 on page 456).

ListType Node

The **ListType** node corresponds to a special case of the **<xsd:simpleType>** element, which contains a sequence of simple (atomic) types.

Figure 393 List Type Node Icon

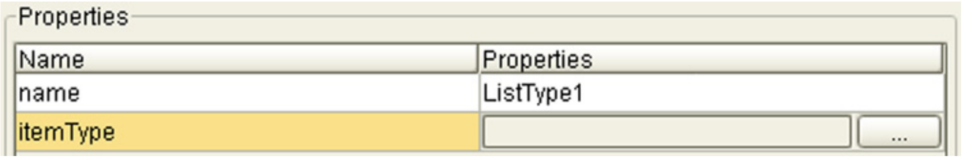
Context Menu

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 394.

Figure 394 ListType Node Properties



Properties

Name	Properties
name	ListType1
itemType	...

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 117 ListType Node Property Options

Name	Description
name	For global nodes only, name of the **ListType** 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 **string**.

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 395 Sequence Node Icon

Context Menu

Figure 396 Sequence Node Menu

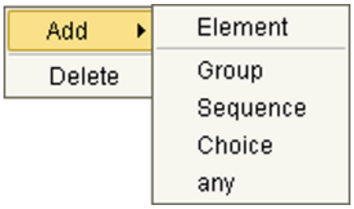


Table 118 Sequence Node Menu Options

Option		Action
Add	Element	Adds an **Element** sub-node to the selected Sequence node (duplicates the action of the **Add Element** toolbar icon). See **Element Node** on page 446.
	Group	Adds a **Group** sub-node to the selected Sequence node (duplicates the action of the **Add Group** toolbar icon). See **Group Node** on page 452.
	Sequence	Adds a **Sequence** sub-node to the selected Sequence node (duplicates the action of the **Add Sequence** toolbar icon). See **Sequence Node** on page 461.
	Choice	Adds a **Choice** sub-node to the selected Sequence node (duplicates the action of the **Add Choice** toolbar icon). See **Choice Node** on page 462.
	Any	Adds an **Any** sub-node to the selected Sequence node. See **Any Node** on page 464.
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 397 Choice Node Icon

Context Menu

Figure 398 Choice Node Menu

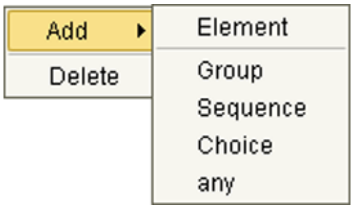


Table 119 Choice Node Menu Options

Option		Action
Add	Element	Adds an **Element** sub-node to the selected Choice node (duplicates the action of the **Add Element** toolbar icon). See **Element Node** on page 446.
	Group	Adds a **Group** sub-node to the selected Choice node (duplicates the action of the **Add Group** toolbar icon). See **Group Node** on page 452.
	Sequence	Adds a **Sequence** sub-node to the selected Choice node (duplicates the action of the **Add Sequence** toolbar icon). See **Sequence Node** on page 461.
	Choice	Adds a **Choice** sub-node to the selected Choice node (duplicates the action of the **Add Choice** toolbar icon). See **Choice Node** on page 462.
	Any	Adds an **Any** sub-node to the selected Choice node. See **Any Node** on page 464.
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 399 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 400 All Node Menu

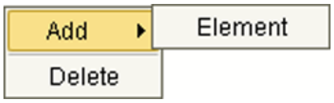


Table 120 All Node Menu Options

Option		Action
Add	Element	Adds an **Element** sub-node to the selected All node (duplicates the action of the **Add Element** toolbar icon). See **Element Node** on page 446.
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 401 Any Node Icon

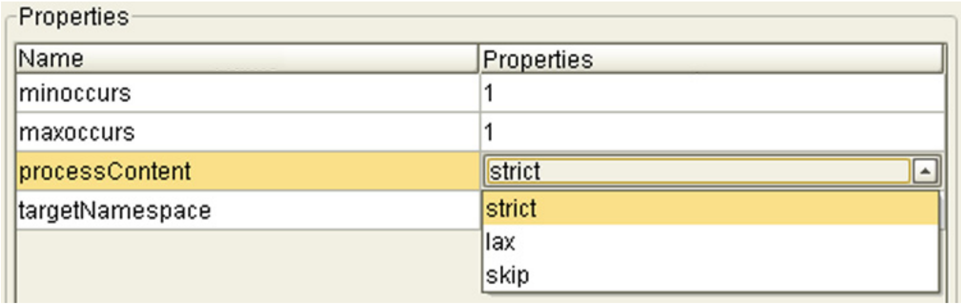
Context Menu

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 394.

Figure 402 Any Node Properties



Properties

Name	Properties
minoccurs	1
maxoccurs	1
processContent	strict
targetNamespace	

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 121 Any Node Property Options

Name	Description
minOccurs	Specifies the minimum number of times this element may appear; the default value is **1**.
maxOccurs	Specifies the maximum number of times this element may appear; the default value is **1**.

Name	Description
processContent	Specifies how the XML processor treats the element content. ▪ **strict** requires that the content be guaranteed valid (default). ▪ **lax** 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. ▪ **skip** instructs the XML processor not to attempt validation of the content.
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 403 Complex Content Node Icon

Context Menu

Figure 404 ComplexContent Node Menu

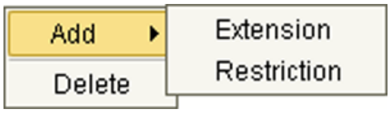


Table 122 ComplexContent Node Menu Options

Option		Action
Add	Extension	Adds an **Extension** sub-node to the selected ComplexContent node. See **Extension Node** on page 468.
	Restriction	Adds a **Restriction** sub-node to the selected ComplexContent node. See **Restriction Node** on page 469.

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 405 Simple Content Node Icon

Context Menu

Figure 406 SimpleContent Node Menu

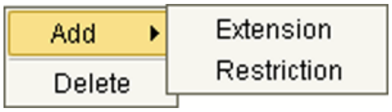


Table 123 SimpleContent Node Menu Options

Option		Action
Add	Extension	Adds an **Extension** sub-node to the selected SimpleContent node. See **Extension Node** on page 468.
	Restriction	Adds a **Restriction** sub-node to the selected SimpleContent node. See **Restriction Node** on page 469.

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 407 Extension Node Icon

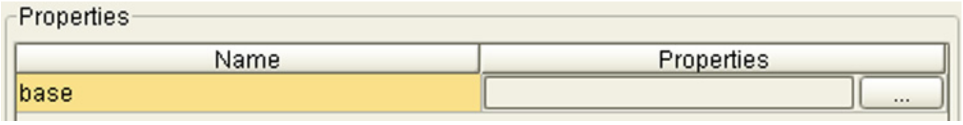
Context Menu

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 408.

Figure 408 Extension Node Properties



Properties

Name	Properties
base	...

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 124 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 409 Restriction Node Icon

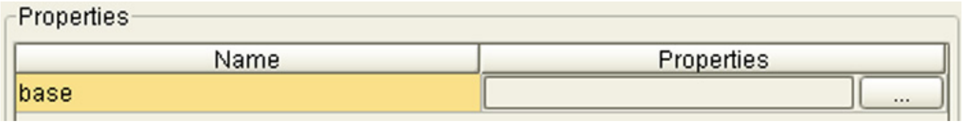
Context Menu

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 410.

Figure 410 Restriction Node Properties



Properties

Name	Properties
base	...

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 125 Restriction Node Property Options

Name	Description
base	Specifies the existing type upon which the declared restriction is based.

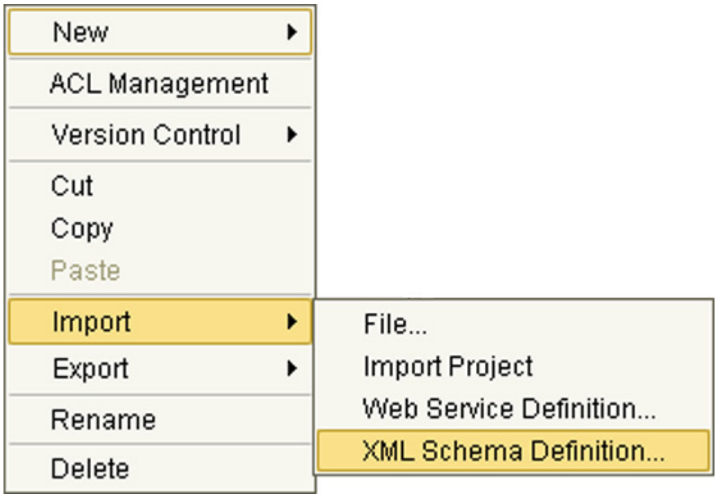
12.4.6 Importing Existing XML Schema Definitions

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

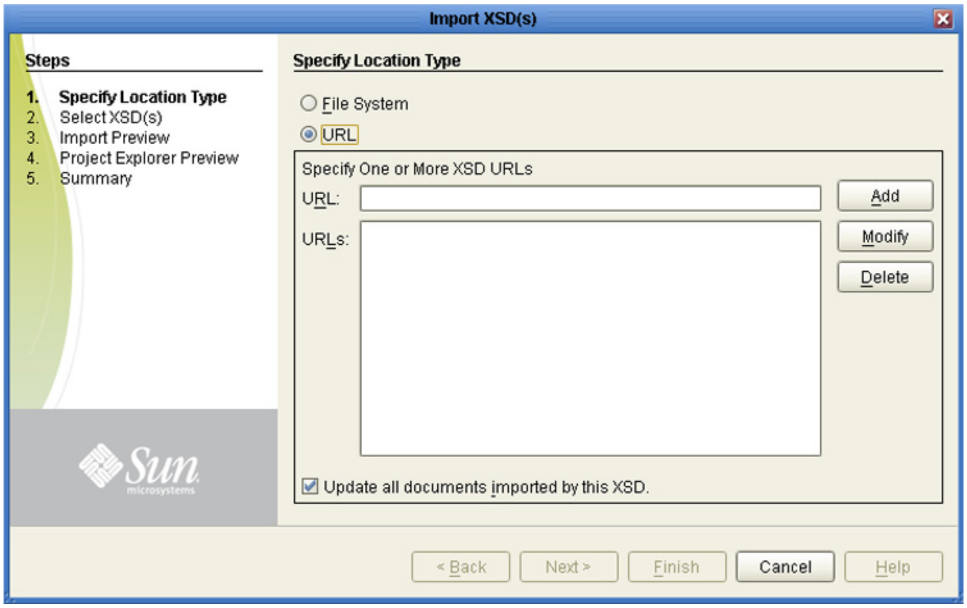
1. In Project Explorer, right-click the selected Project to display its context menu (see Figure 411).

Figure 411 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 412.

Figure 412 Import XSD File Dialog - Step 1



Import XSD(s)

Steps

1. **Specify Location Type**
2. Select XSD(s)
3. Import Preview
4. Project Explorer Preview
5. Summary

Specify Location Type

○ File System
● URL

Specify One or More XSD URLs

URL:

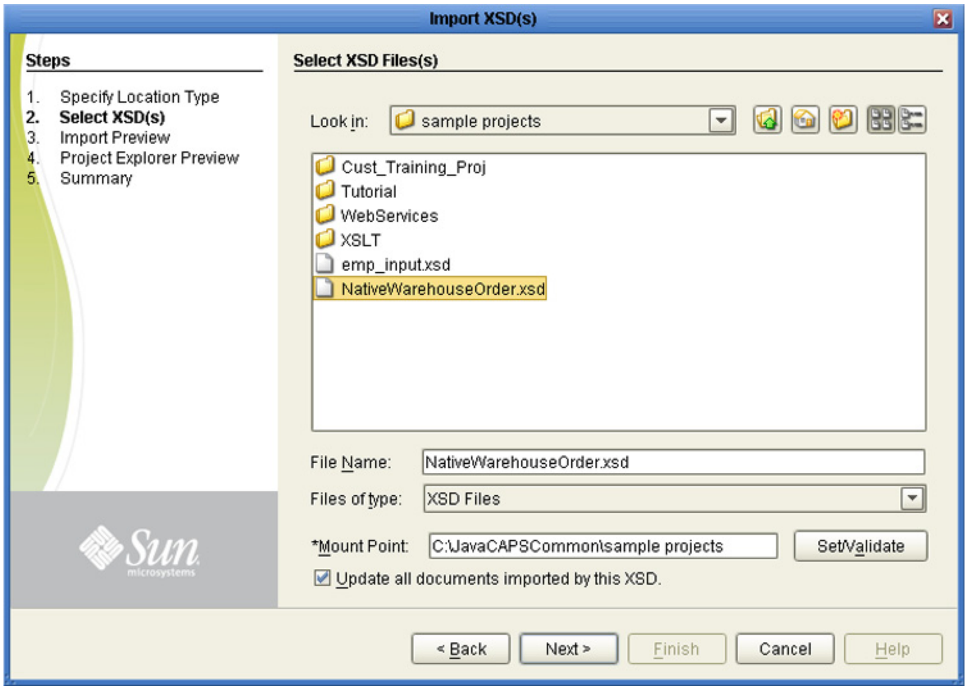
URLs:

☑ Update all documents imported by this XSD.

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 413, which shows a local file system).

Figure 413 Import XSD File Dialog - Step 2



Import XSD(s)

Steps

1. Specify Location Type
2. **Select XSD(s)**
3. Import Preview
4. Project Explorer Preview
5. Summary

Select XSD Files(s)

Look in: sample projects

- Cust_Training_Proj
- Tutorial
- WebServices
- XSLT
- emp_input.xsd
- NativeWarehouseOrder.xsd

File Name: NativeWarehouseOrder.xsd

Files of type: XSD Files

*Mount Point: C:\JavaCAPSCommon\sample projects

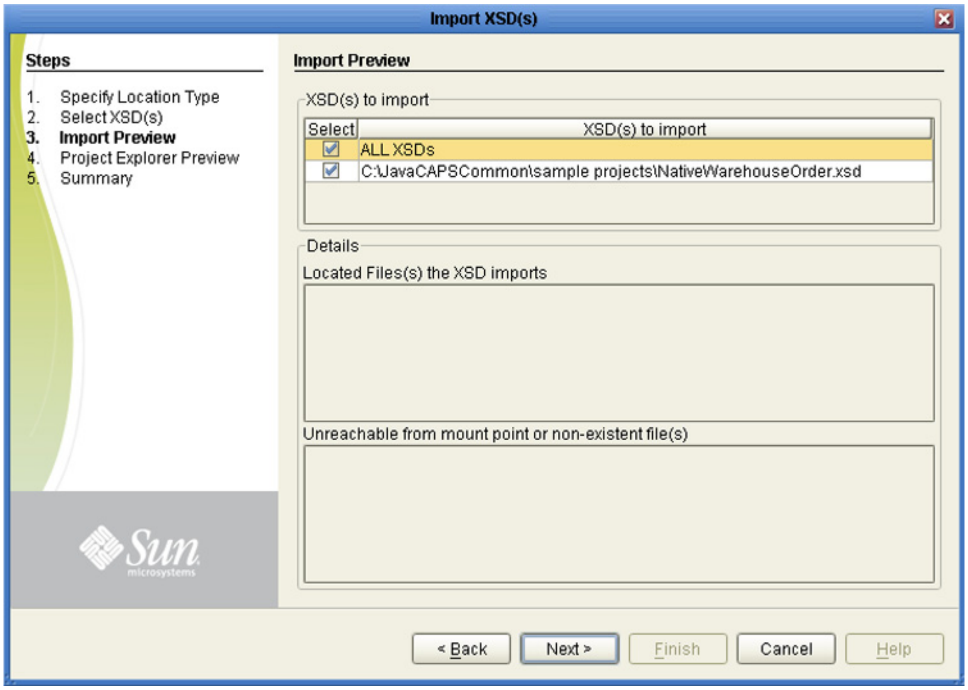
☑ Update all documents imported by this XSD.

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 414).

Figure 414 Import XSD File Dialog - Step 3



Import XSD(s)
Steps
Import Preview
XSD(s) to import
Details
Located Files(s) the XSD imports
Unreachable from mount point or non-existent file(s)
< Back
Next >
Finish
Cancel
Help

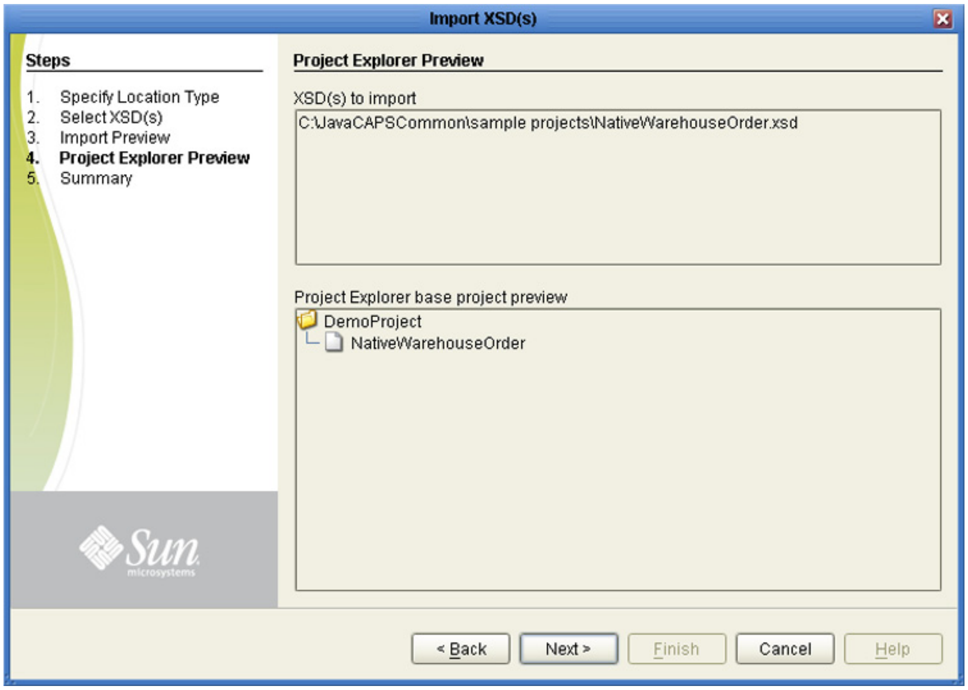
1. Specify Location Type
2. Select XSD(s)
3. **Import Preview**
4. Project Explorer Preview
5. Summary

Select	XSD(s) to import
☑	ALL XSDs
☑	C:\JavaCAPSCommon\sample projects\NativeWarehouseOrder.xsd

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 415).

Figure 415 Import XSD File Dialog - Step 4



XSD(s) to import

C:\JavaCAPSCommon\sample projects\NativeWarehouseOrder.xsd

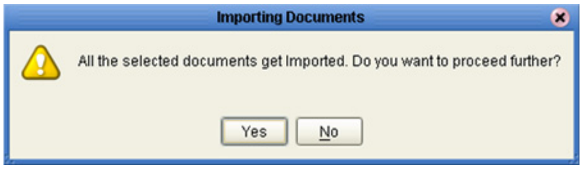
Project Explorer base project preview

DemoProject
└ NativeWarehouseOrder

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 416.

Figure 416 Import XSD File Dialog - Step 4 Confirmation

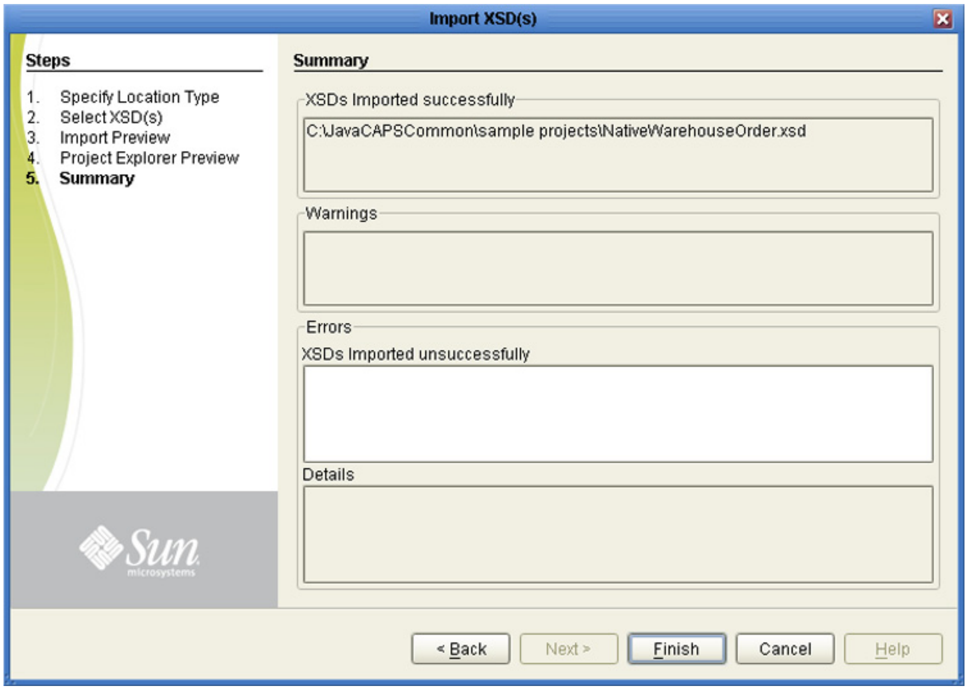


All the selected documents get Imported. Do you want to proceed further?

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 417).

Figure 417 Import XSD File Dialog - Step 5



C:\JavaCAPSCommon\sample projects\NativeWarehouseOrder.xsd

13 Assuming your files were imported successfully, click **Finish**.

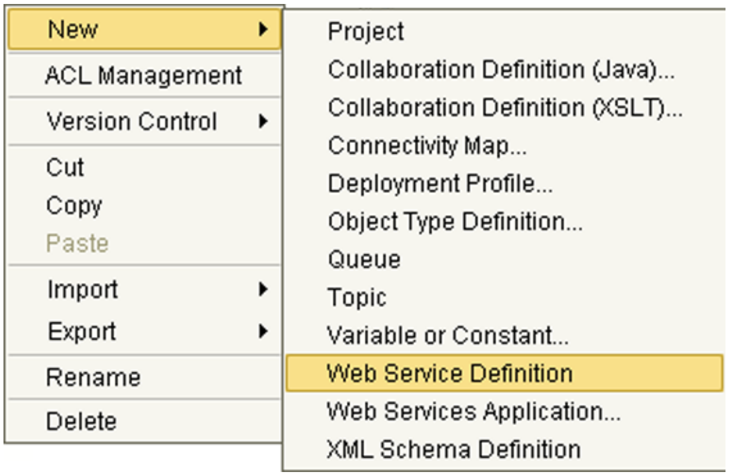
12.5 Creating Web Service Definitions

12.5.1 Overview

A Web Service Definition, or WSD Object, can either be formed by importing an existing WSDL document (see **Importing Existing Web Service Definitions** on page 499) or newly created using the Web Service Definition Editor. The minimum requirement for a WSD Object is that it represent an abstract WSDL document by containing (directly or by import) one or more Operations and related messages. The WSD Object can also contain binding information (see **Configuring WSDL Binding Properties** on page 504).

Selecting the **New > Web Service Definition** option from the *Project* context menu (see Figure 418) adds a WSD Object to your Project and automatically opens the Web Service Definition Editor (see **Using the Web Service Definition Editor** on page 476).

Figure 418 Project Context Menu - New Web Service Definition



12.5.2 Using the Context Menu

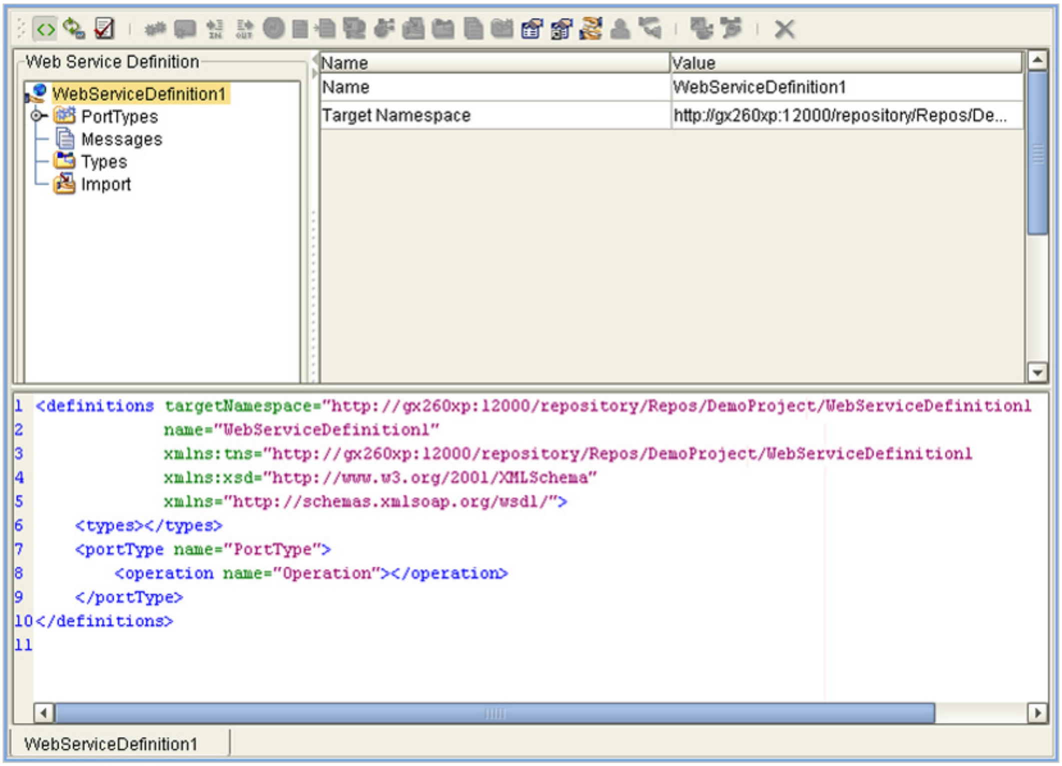
Right-clicking the WSD Object in Project Explorer displays its context menu, which follows the standard Project component format (see **Using Project Component Context Menus** on page 113).

12.6 Using the Web Service Definition Editor

12.6.1 Overview

The Web Service Definition Editor (see Figure 419) is an editor within Enterprise Designer, with which you can develop new, or edit existing, WSD Objects with a minimum of coding.

Figure 419 Web Service Definition Editor User Interface



Name	Value
Name	WebServiceDefinition1
Target Namespace	http://gx260xp:12000/repository/Repos/De...

```
<definitions targetNamespace="http://gx260xp:12000/repository/Repos/DemoProject/WebServiceDefinition1
             name="WebServiceDefinition1"
             xmlns:tns="http://gx260xp:12000/repository/Repos/DemoProject/WebServiceDefinition1
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns="http://schemas.xmlsoap.org/wsdl/">
    <types></types>
    <portType name="PortType">
        <operation name="Operation"></operation>
    </portType>
</definitions>
```

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 WSD Object graphically. See **Web Service Definition Editor Toolbar Icons** on page 478 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 420. 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 481.

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 481.

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 478).

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 478.

12.6.2 Web Service Definition Editor Toolbar Icons

Table 126 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 **.wsdl** 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 *PortTypes* folder (duplicates the action of the **Add Port Type** option on the *PortTypes* folder context menu). See PortTypes on page 489.
	Operation	Adds a new Operation to the selected PortType (duplicates the action of the **Add Operation** option on the PortType context menu). See Operations on page 490.
IN	Input Message	Adds a new input message to the selected Operation (duplicates the action of the **Add Input Message** option on the Operation context menu). See Operations on page 490.
OUT	Output Message	Adds a new output message to the selected Operation (duplicates the action of the **Add Output Message** option on the Operation context menu). See Operations on page 490.
N	Exception	Adds an exception (or fault message) to an output message (duplicates the action of the **Add Fault** option on the Operation context menu). Option is disabled unless an output message exists for the Operation. See Operations on page 490.
	Message Part	Adds a new message under the *Messages* folder (duplicates the action of the **Add Message** option on the *Messages* folder context menu). See Messages on page 485.
	Message Value	Adds a new value to the selected message (duplicates the action of the **Add Part** option on the Message context menu). See Message Parts on page 487.

Table 126 Web Service Definition Editor Toolbar Icons

Icon	Command	Function
	Import Web Service	Displays a dialog for importing a WSDL or XSD file to the *Import* folder (duplicates the action of the **Import WSDL, XSD or OTD** option on the *Import* folder context menu). **Important:** 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 *Types* folder. **Note:** This functionality is not used in Java CAPS 5.1.x.
	Add Import Folder	Adds a new *Import* folder under the Web Service Definition root node (duplicates the action of the **Add Import Folder** option on the Web Service Definition root node context menu). Icon is disabled if an *Import* folder already exists for the WSD Object. See **Import Folder** on page 483.
	Add Schema Folder	Adds a new *Types* folder under the Web Service Definition root node (duplicates the action of the **Add Schema Types** option on the Web Service Definition root node context menu). Icon is disabled if a *Types* folder already exists for the WSD Object. See **Types Folder** on page 498.
	Add Message Folder	Adds a new *Messages* folder under the Web Service Definition root node (duplicates the action of the **Add Message Folder** option on the Web Service Definition root node context menu). Icon is disabled if a *Messages* folder already exists for the WSD Object. See **Messages Folder** on page 485.
	Add PortTypes Folder	Adds a new *PortTypes* folder under the Web Service Definition root node (duplicates the action of the **Add PortType Folder** option on the Web Service Definition root node context menu). Icon is disabled if a *PortTypes* folder already exists for the WSD Object. See **PortTypes Folder** on page 489.
	Add Property	Adds a new *Properties* folder under the Web Service Definition root node and a new property in the folder (duplicates the action of the **Add Property** option on the Web Service Definition root node context menu). Icon is disabled if a *Properties* folder already exists for the WSD Object. See **Property** on page 493. **Note:** This functionality is not used in Java CAPS 5.1.x.
	Add Property Alias	Adds a new *Property Aliases* folder under the root node (duplicates the action of the **Add Property Alias** option on the Web Service Definition root node context menu). Icon is disabled if a *Property Aliases* folder already exists for the WSD Object. See **Property Aliases Folder** on page 495. **Note:** This functionality is not used in Java CAPS 5.1.x.

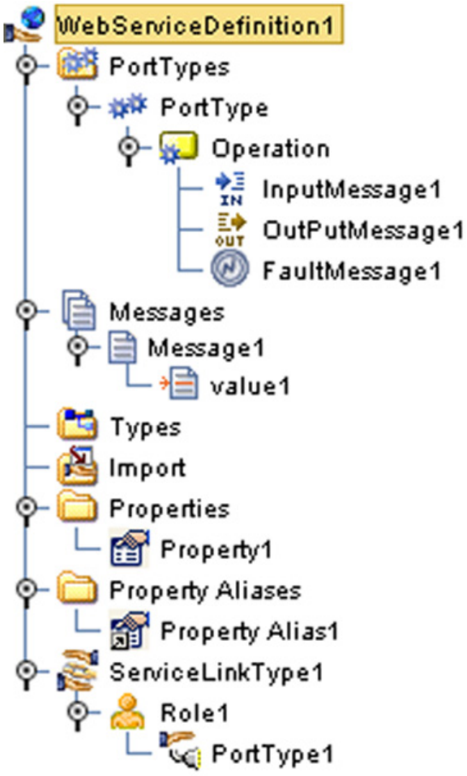
Table 126 Web Service Definition Editor Toolbar Icons

Icon	Command	Function
	Add Service Link Type	Adds a new *Service Link Type* folder under the root node (duplicates the action of the **Add Service Link Type** option on the Web Service Definition root node context menu). Icon is disabled if a *Service Link Type* folder already exists for the WSD Object. See **Service Link Type Folder** on page 497. **Note:** This functionality is not used in Java CAPS 5.1.x.
	Add Role	Adds a new Role under the Service Link Type folder (duplicates the action of the **Add Service Link Role** option on the *Service Link Type* folder context menu). See **Service Link Role** on page 497. **Note:** This functionality is not used in Java CAPS 5.1.x.
	Add PortType	Adds a new PortType to the Service Link Role (duplicates the action of the **Add Service Link PortType** option on the *Role* context menu). See **PortTypes** on page 506.
	Preview XSD Node	Presents a dialog containing allowed options; clicking **OK** 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 **Previewing and Exporting XSD Nodes** on page 433.
	Export XSD Node	Presents a dialog containing allowed options; clicking **OK** in the dialog exports the selected top-level Element, ComplexType, or SimpleType as an *XSD Node*, which can be used in a Collaboration Definition (Java). See **Previewing and Exporting XSD Nodes** on page 433.
✕	Delete	Deletes the selected node from the XSD tree, including all sub-nodes.

12.6.3 Building a Web Service Definition

A WSD 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 420 shows an example WSDL tree structure demonstrating all possible node combinations (the *Types* and *Import* folders are unpopulated, however).

Figure 420 WSDL Tree - All Options



You can also build the WSD 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 478. 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 478).

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 421 or the corresponding icons in the toolbar.

Context Menu

Figure 421 Web Service Definition Root Node Menu

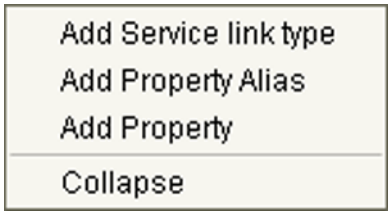


Table 127 Web Service Definition Root Node Menu Options

Command	Function
Add Service Link Type	Adds a new *Service Link Type* folder under the Web Service Definition root node (duplicates the action of the **Add Service Link Type** icon). The folder is automatically populated with a default Role and Port Type. Option is not shown if a *Service Link Type* folder already exists for the WSD Object. **Note:** This functionality is not used in Java CAPS 5.1.x.
Add Property Alias Folder	Adds a new Property Alias Folder under the Web Service Definition root node (duplicates the action of the **Add Property Alias** icon). The folder is automatically populated with a default Property Alias. Option is not shown if a *Property Alias* folder already exists for the WSD Object. **Note:** This functionality is not used in Java CAPS 5.1.x.
Add Property	Adds a new *Properties* folder under the Web Service Definition root node and a new property in the folder (duplicates the action of the **Add Property** icon). The folder is automatically populated with a default Property. Option is not shown if a *Properties* folder already exists for the WSD Object. **Note:** This functionality is not used in Java CAPS 5.1.x.
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 422) or by using the *Import Web Service* icon in the toolbar.

Context Menu

Figure 422 Import Folder Menu

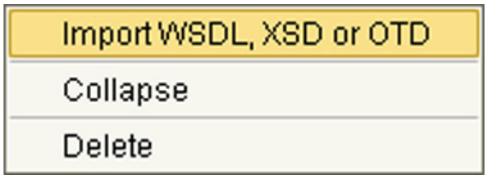
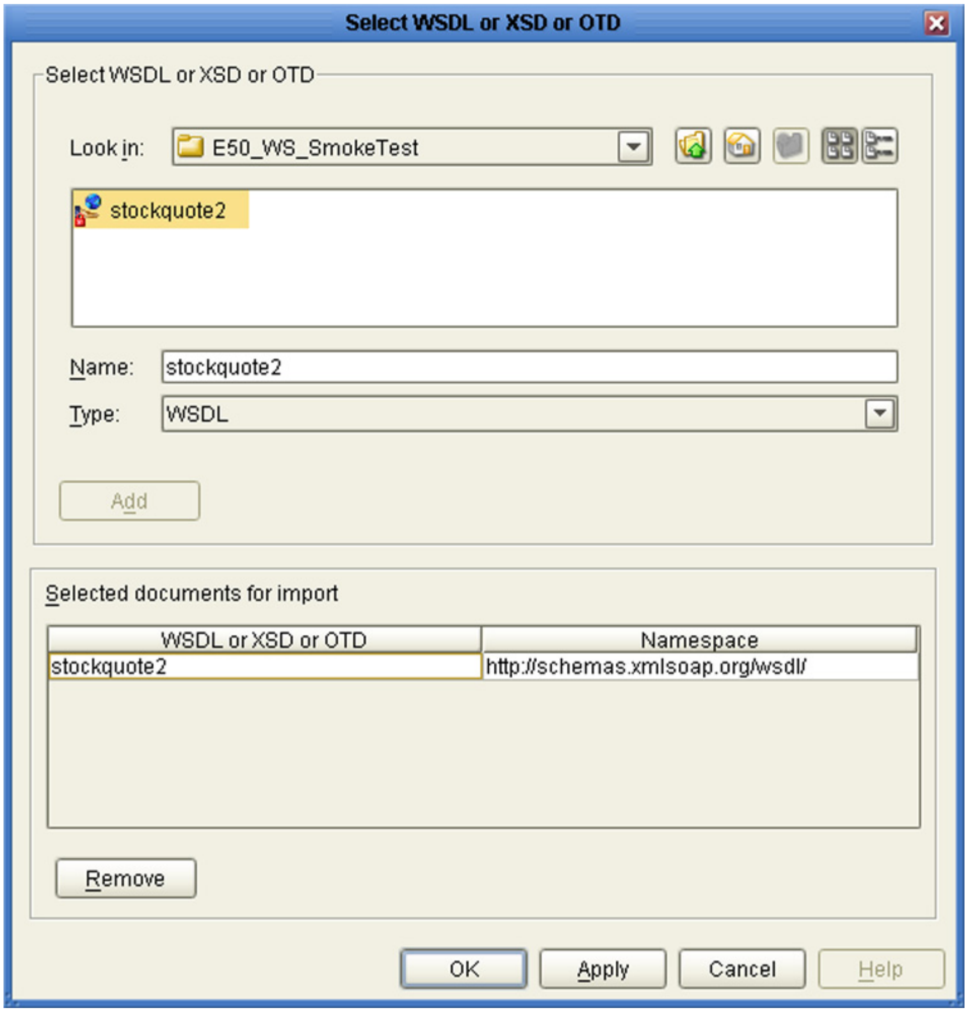


Table 128 Import Folder Menu Options

Command	Function
Import WSDL, XSD or OTD	Displays a dialog (see Figure 423) for importing a WSDL or XSD file, or an OTD, to the Import folder (duplicates the action of the **Import Web Service** 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 423 Select WSDL, XSD or OTD Dialog



Select WSDL or XSD or OTD

Select WSDL or XSD or OTD

Look in: E50_WS_SmokeTest

stockquote2

Name: stockquote2

Type: WSDL

Add

Selected documents for import

WSDL or XSD or OTD	Namespace
stockquote2	http://schemas.xmlsoap.org/wsdl/

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 424) or by using the *Message Part* icon in the toolbar.

Context Menu

Figure 424 Messages Folder Menu

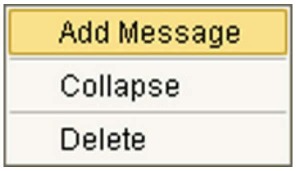


Table 129 Messages Folder Menu Options

Command	Function
Add Message	Adds a new message under the Messages folder (duplicates the action of the **Message Part** 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 425) or by using the *Message Value* icon in the toolbar.

Context Menu

Figure 425 Message Menu

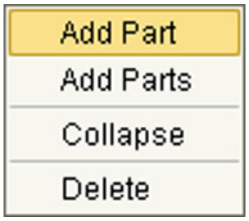


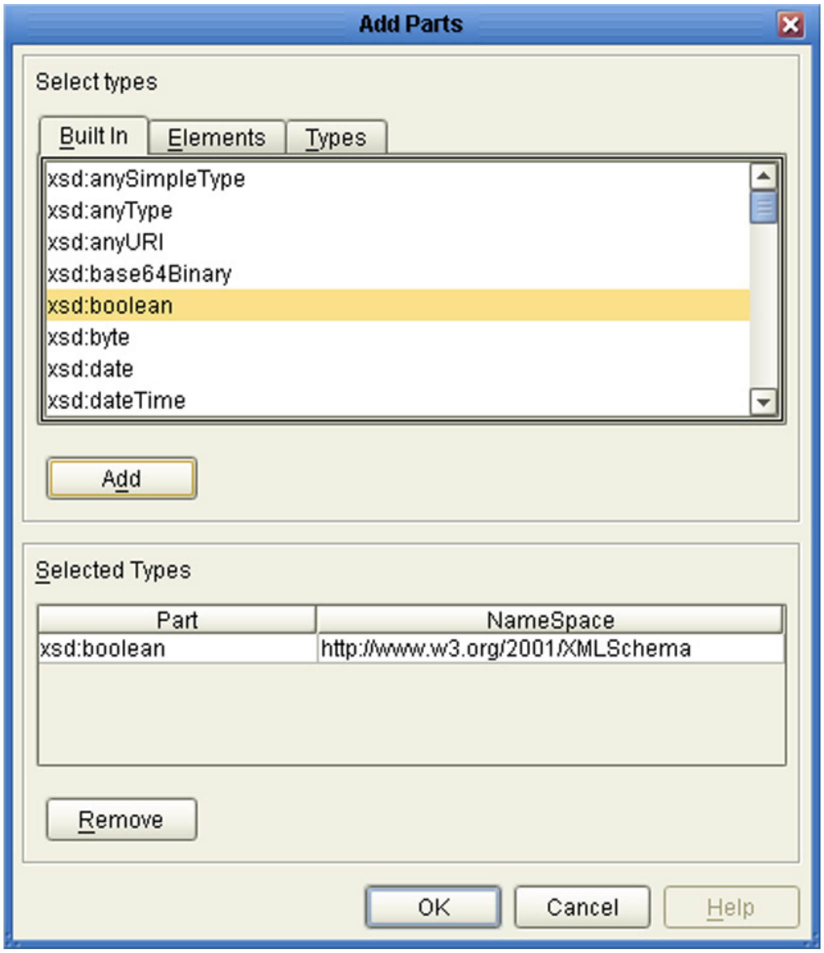
Table 130 Message Menu Options

Command	Function
Add Part	Adds a new value to the selected message (duplicates the action of the **Message Value** icon).

Table 130 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 426).
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 426 Add Parts Dialog



Part	NameSpace
xsd:boolean	http://www.w3.org/2001/XMLSchema

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.

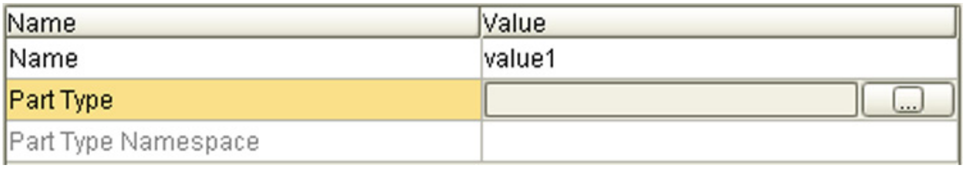
Context Menu

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 427. 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 427 Message Part Properties (Initial)



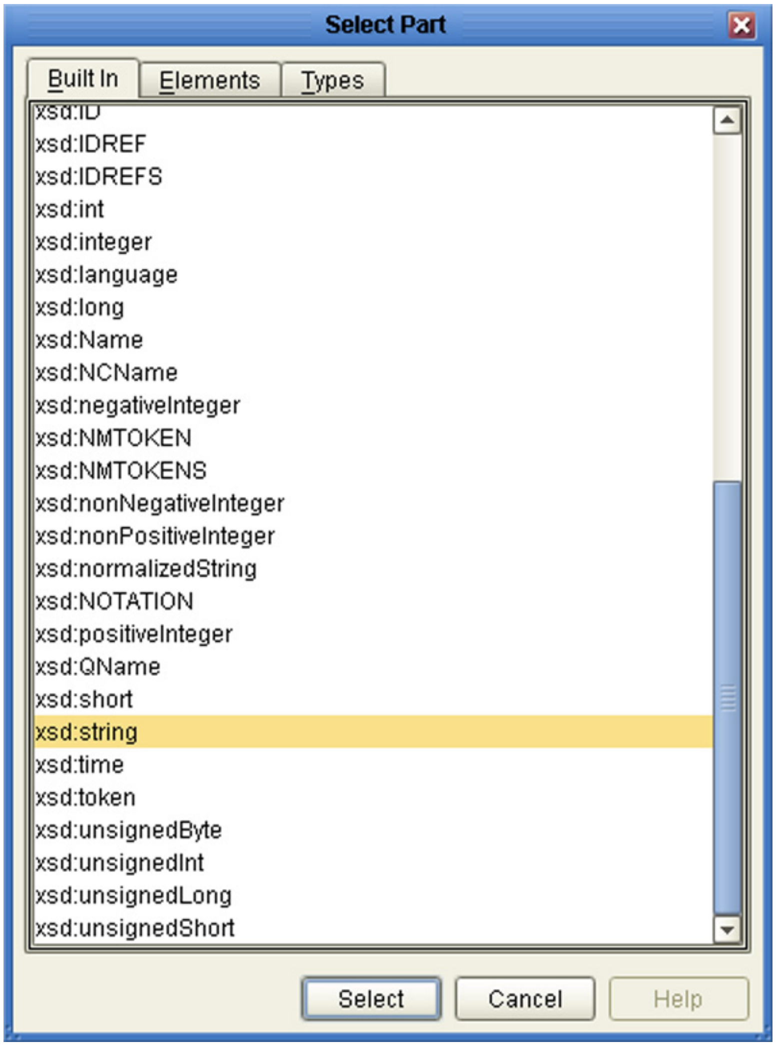
Name	Value
Name	value1
Part Type	...
Part Type Namespace	

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 131 Message Value Node Property Options

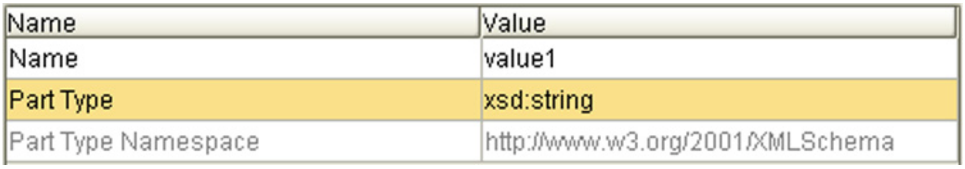
Name	Description
name	Name of the value.
Part Type	Selecting the field and clicking the ellipsis (**...**) button displays a dialog containing a list of existing types (see Figure 428). You can select a type based on standard simple types, or other types existing in internal or imported documents.
Part Type Namespace	The namespace within which this part type belongs (see Figure 429 for an example). The value for this property is entered automatically when you select the Part Type.

Figure 428 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 429.

Figure 429 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 430) or by using the *PortType* icon in the toolbar.

Context Menu

Figure 430 PortTypes Folder Menu

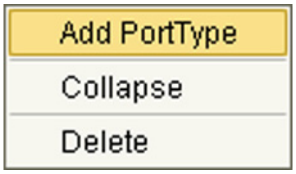


Table 132 PortTypes Folder Menu Options

Command	Function
Add PortType	Adds a new PortType under the PortType folder (duplicates the action of the **PortType** 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 431) or by using the *Operation* icon in the toolbar.

Context Menu

Figure 431 PortType Menu

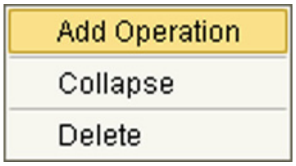


Table 133 PortType Menu Options

Command	Function
Add Operation	Adds a new Operation to the selected PortType (duplicates the action of the **Operation** 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 133 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 432 Operation Menu

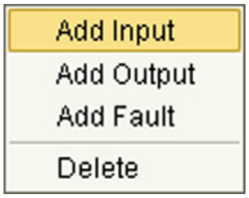


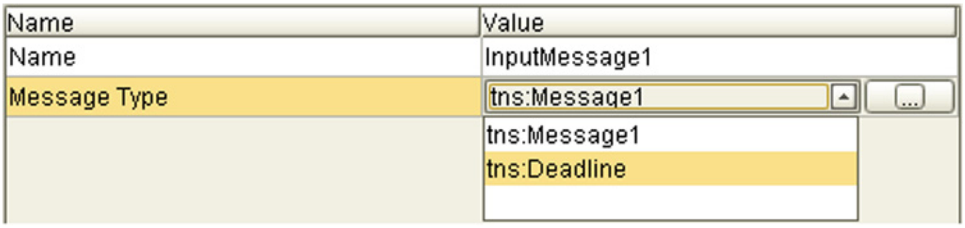
Table 134 Operation Menu Options

Command	Function
Add Input	Adds an input message to the selected Operation (duplicates the action of the **Input Message** icon). Each Operation can have only one input message.
Add Output	Adds an output message to the selected Operation (duplicates the action of the **Output Message** 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 **Exception** 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 433. 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.

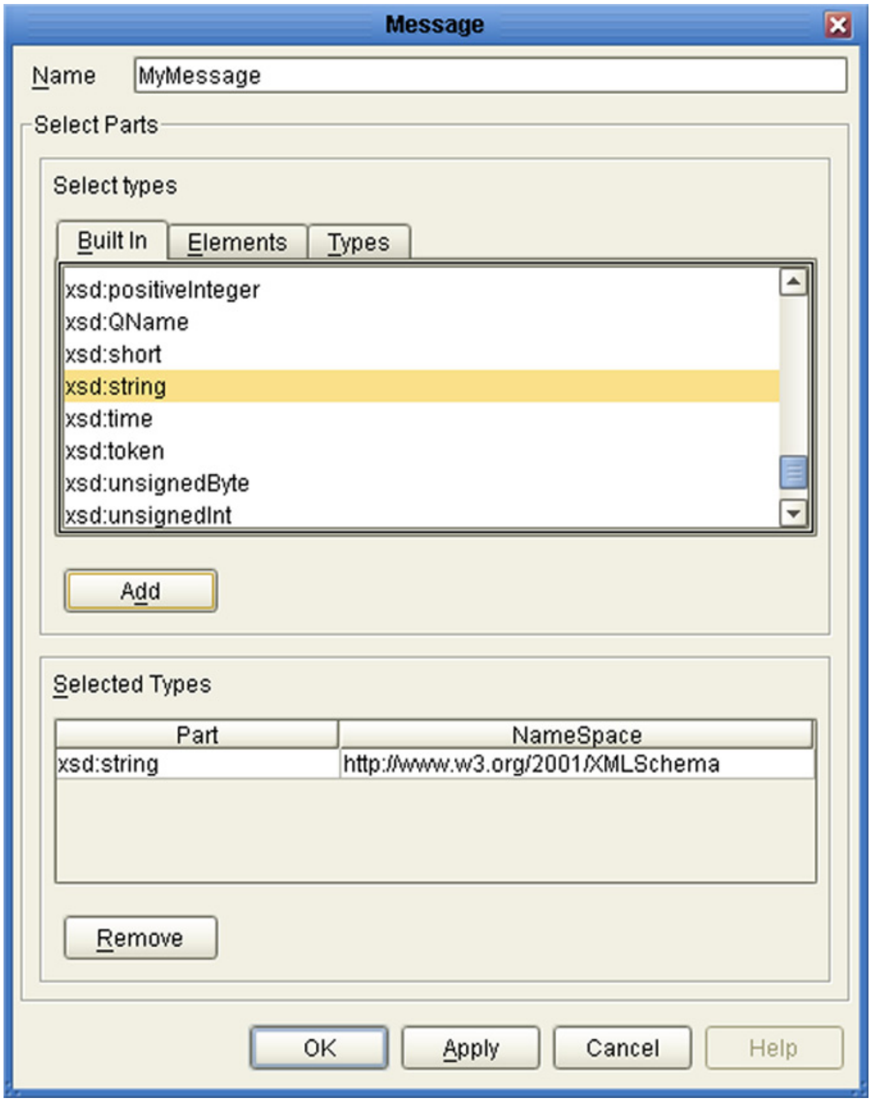
Figure 433 Message Properties



Name	Value
Name	InputMessage1
Message Type	tns:Message1

To add a message type, click the ellipsis (**...**) button, which opens the *Message* dialog, shown in Figure 434. 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 433.

Figure 434 Message Dialog



Name: MyMessage

xsd:positiveInteger
xsd:QName
xsd:short
xsd:string
xsd:time
xsd:token
xsd:unsignedByte
xsd:unsignedInt

Part	NameSpace
xsd:string	http://www.w3.org/2001/XMLSchema

Properties Folder

Note: *This functionality is not used in Java CAPS 5.1.x.*

The Properties folder serves as a container for individual Properties, which can be virtually any type of classification or identification information.

Context Menu

Figure 435 Properties Folder Menu

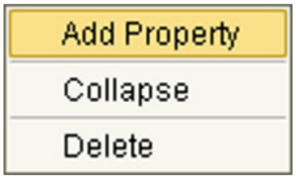


Table 135 Properties Folder Menu Options

Command	Function
Add Property	Adds a new property under the Properties folder (duplicates the action of the **Add Property** icon). **Note:** This functionality is not used in Java CAPS 5.1.x.
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

Note: *This functionality is not used in Java CAPS 5.1.x.*

Properties are usually expressed as text or XML. References to resources are entered as property aliases (see **Property Alias** on page 496).

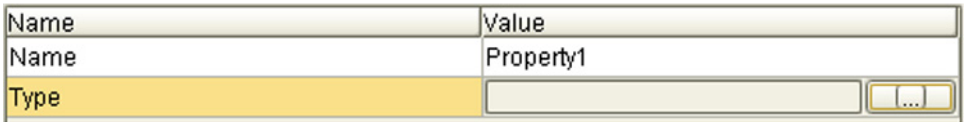
Context Menu

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 436.

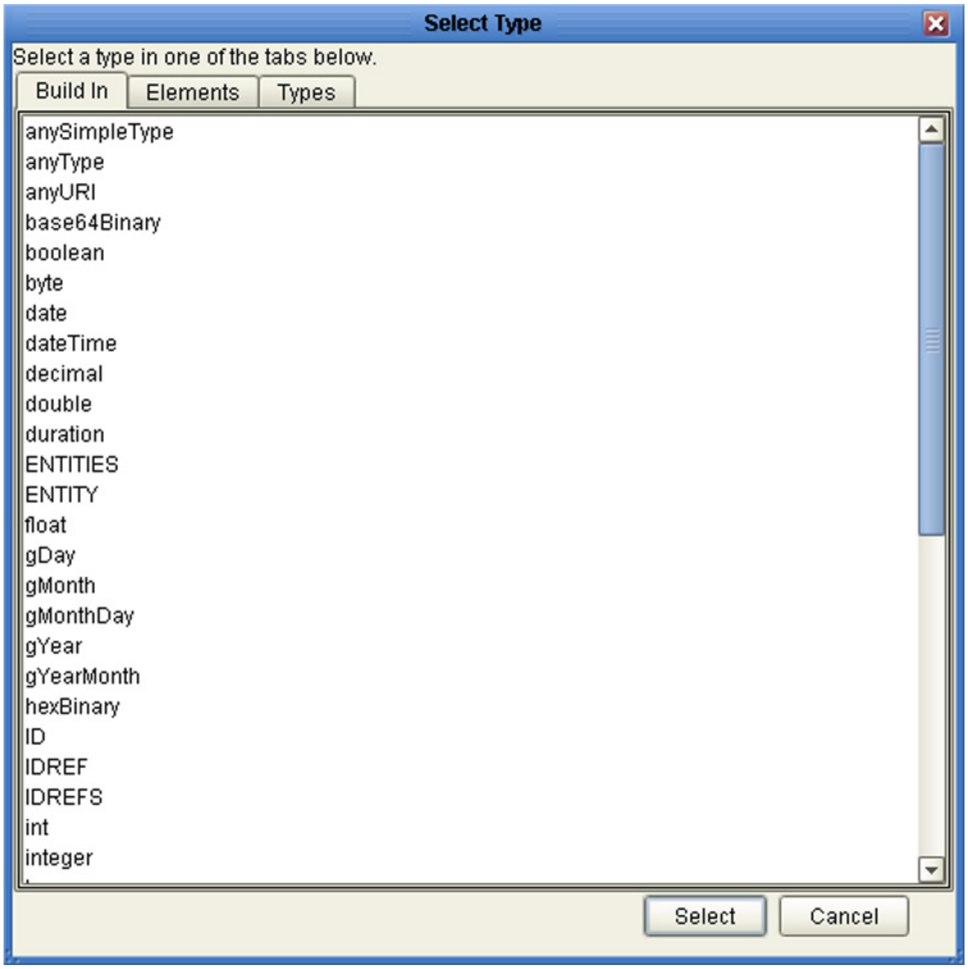
Figure 436 Property Properties



Name	Value
Name	Property1
Type	

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 437.

Figure 437 Select Type Dialog



Select a type in one of the tabs below.

anySimpleType
anyType
anyURI
base64Binary
boolean
byte
date
dateTime
decimal
double
duration
ENTITIES
ENTITY
float
gDay
gMonth
gMonthDay
gYear
gYearMonth
hexBinary
ID
IDREF
IDREFS
int
integer

Property Aliases Folder

Note: *This functionality is not used in Java CAPS 5.1.x.*

Context Menu

Figure 438 Property Aliases Folder Menu

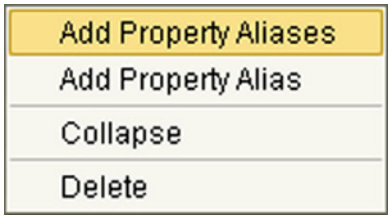
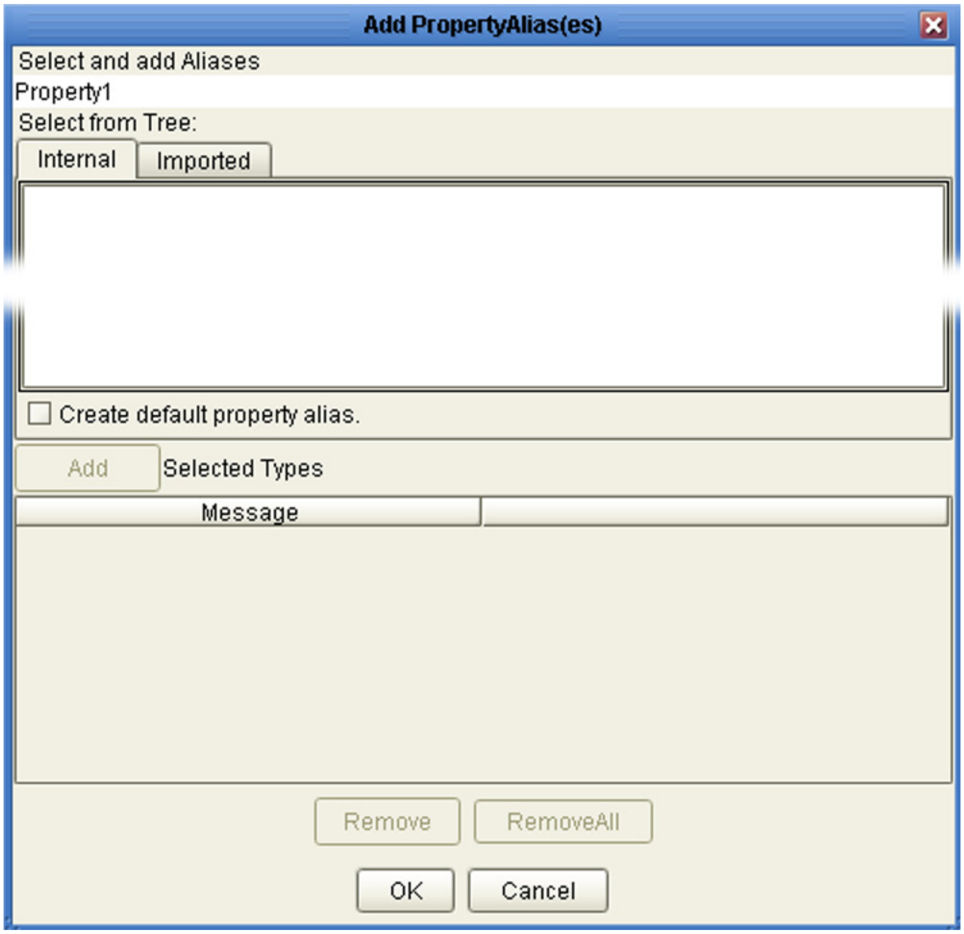


Table 136 Properties Folder Menu Options

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 **Add Property Alias** icon). **Note:** This functionality is not used in Java CAPS 5.1.x.
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 439 Add Property Aliases Dialog



Add PropertyAlias(es)

Select and add Aliases

Property1

Select from Tree:

□ Create default property alias.

Selected Types

Message	

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.x.*

Context Menu

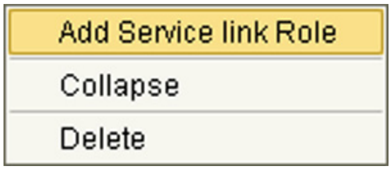
This node cannot have sub-nodes assigned to it; its context menu only offers the ***Delete*** option.

Service Link Type Folder

Note: *This functionality is not used in Java CAPS 5.1.x.*

Context Menu

Figure 440 Service Link Type Folder Menu



Add Service link Role
Collapse
Delete

Table 137 Service Link Type Folder Menu Options

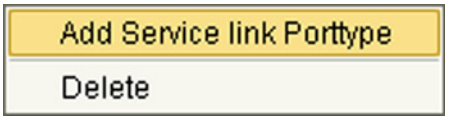
Command	Function
Add Service Link Role	Adds a new role under the Service Link Type folder (duplicates the action of the **Add Role** icon). **Note:** This functionality is not used in Java CAPS 5.1.x.
Collapse/Expand	Toggles to hide or display the sub-node structure, when sub-nodes are present (option 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.x.*

Context Menu

Figure 441 Role Menu



Add Service link Porttype
Delete

Table 138 Role Menu Options

Command	Function
Add Service Link PortType	Adds a new PortType for the selected Service Link Role (duplicates the action of the **Add PortType** icon). **Note:** Only one PortType can be added as a sub-node. After the PortType is added, this option is removed from the menu.

Table 138 Role Menu Options

Command	Function
Collapse	Hides the sub-node structure. **Note:** After a PortType is added, this option is added to the menu.
Delete	Removes the node and all sub-nodes.

Types Folder

Note: *This functionality is not used in Java CAPS 5.1.x.*

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 442.

Context Menu

Figure 442 Schema Type Menu

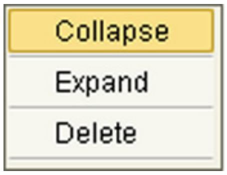


Table 139 Schema Type Menu Options

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 WSD Object by selecting the **Import > Web Service Definition** option from the *Project* context menu, which adds the WSD Object to your Project and automatically opens the Web Service Definition Editor. You can use the Web Service Definition Editor to edit the WSD Object as described in the previous section. See the *Sun SeeBeyond eInsight™ Business Process Manager User's Guide* for additional information.

Important: *Before importing a WSDL file that supports SOAP encoding, you must perform the following procedure:*

1. Using a text editor, create a new file containing the following line:

   ```
   com.stc.wscommon.design.codegen.impl.WsdlMsgNormalizer=enabled
   ```

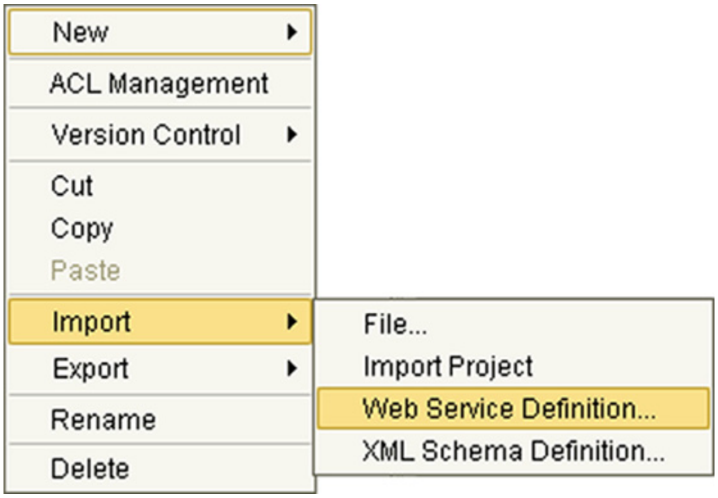
2. Save the file with a **.ecrc** extension in the following directory:

   ```
   C:\Documents and Settings\<your login ID>\
   ```

To import a WSD Object

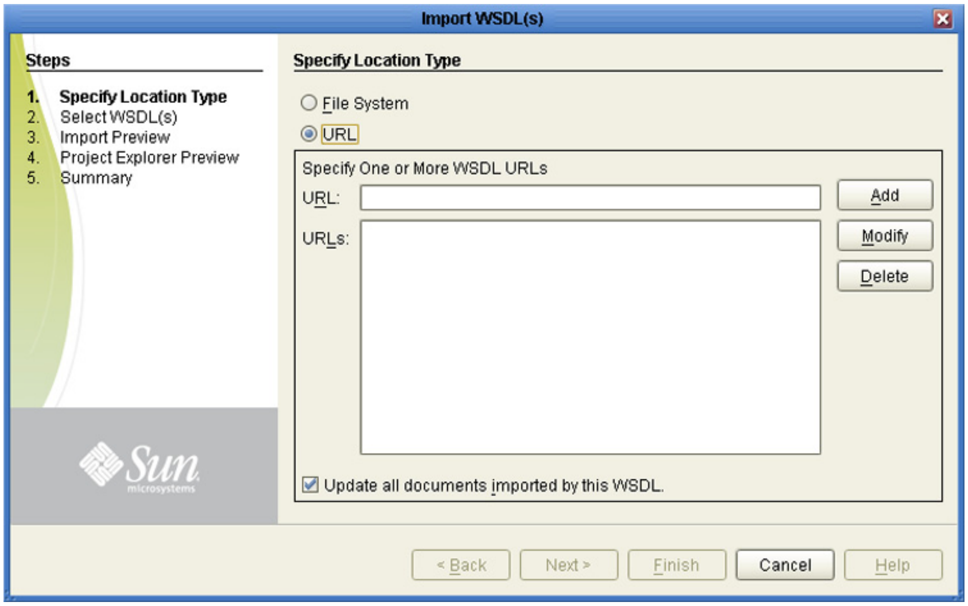
1. In Project Explorer, right-click the selected Project to display its context menu (see Figure 443).

Figure 443 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 444.

Figure 444 WSDL Import Wizard - Step 1



Import WSDL(s)

Steps

1. **Specify Location Type**
2. Select WSDL(s)
3. Import Preview
4. Project Explorer Preview
5. Summary

Specify Location Type

○ File System
● URL

Specify One or More WSDL URLs

URL:

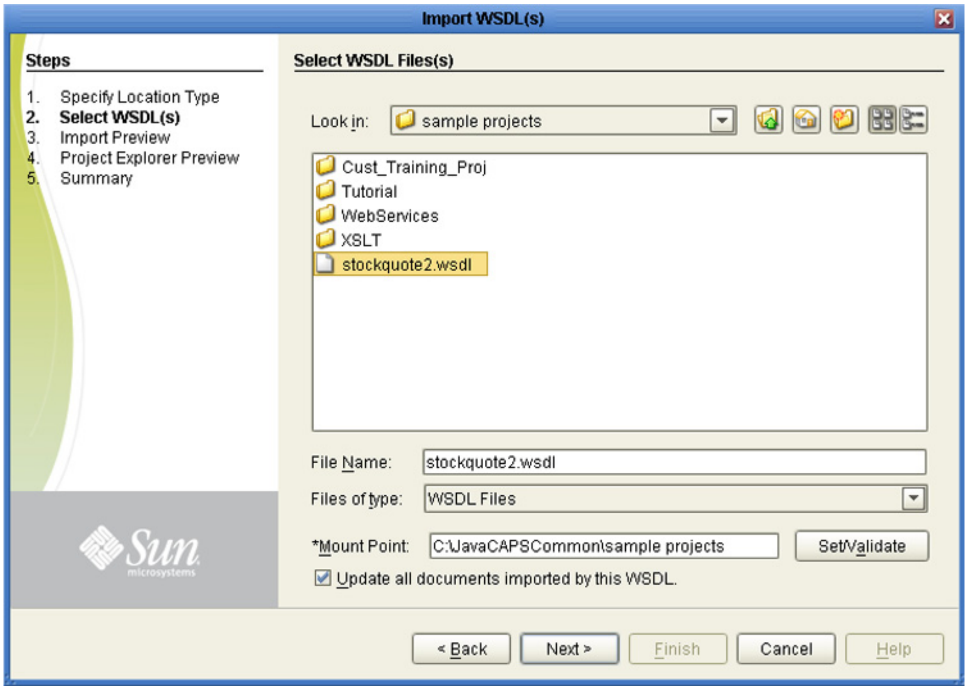
URLs:

☑ Update all documents imported by this WSDL.

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 445, which shows a local file system).

Figure 445 WSDL Import Wizard - Step 2



Import WSDL(s)

Steps

1. Specify Location Type
2. **Select WSDL(s)**
3. Import Preview
4. Project Explorer Preview
5. Summary

Select WSDL Files(s)

Look in: sample projects

Cust_Training_Proj
Tutorial
WebServices
XSLT
stockquote2.wsdl

File Name: stockquote2.wsdl

Files of type: WSDL Files

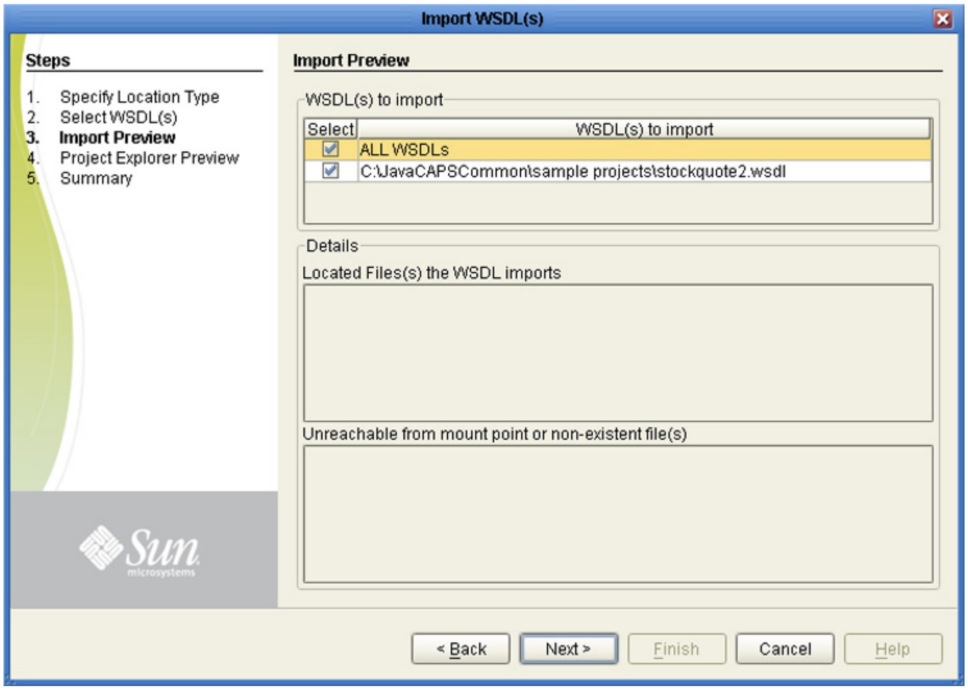
*Mount Point: C:\JavaCAPSCommon\sample projects

☑ Update all documents imported by this WSDL.

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 446).

Figure 446 WSDL Import Wizard - Step 3



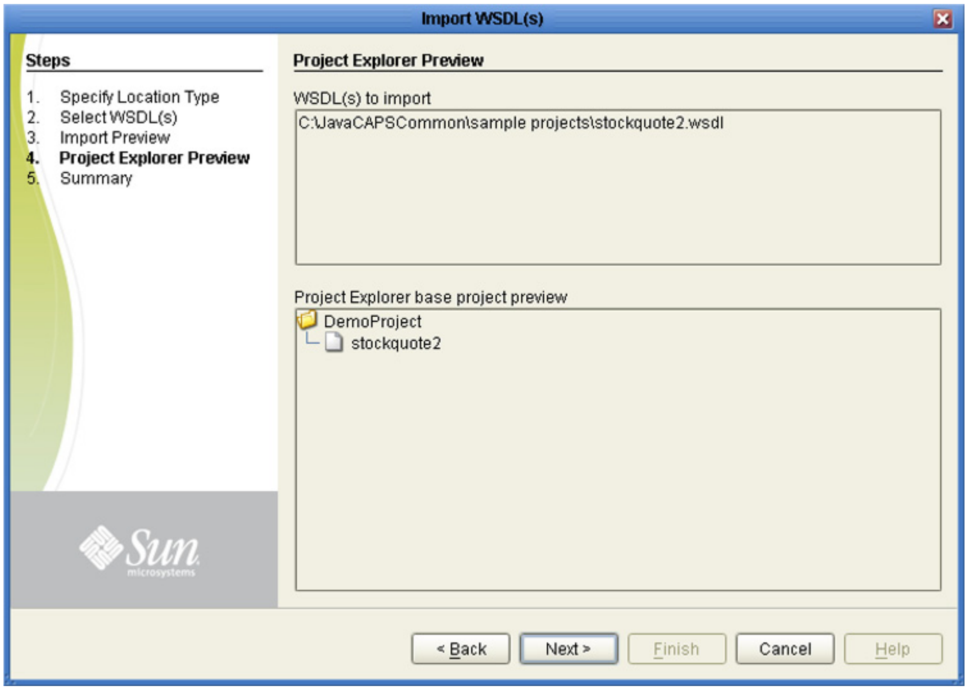
1. Specify Location Type
2. Select WSDL(s)
3. **Import Preview**
4. Project Explorer Preview
5. Summary

Select	WSDL(s) to import
☑	ALL WSDLs
☑	C:\JavaCAPSCommon\sample projects\stockquote2.wsdl

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 447).

Figure 447 WSDL Import Wizard - Step 4



Project Explorer Preview

WSDL(s) to import

C:\JavaCAPSCommon\sample projects\stockquote2.wsdl

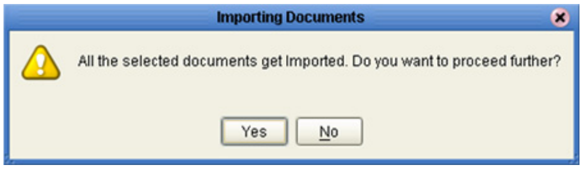
Project Explorer base project preview

DemoProject
└ stockquote2

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 448.

Figure 448 WSDL Import Wizard - Step 4 Confirmation

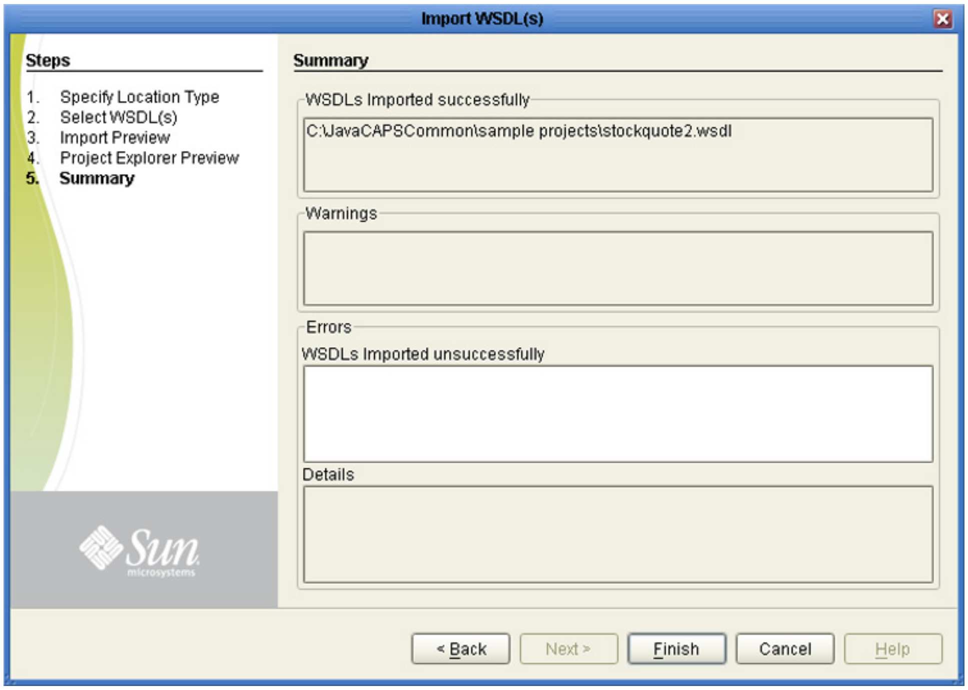


All the selected documents get Imported. Do you want to proceed further?

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 449).

Figure 449 WSDL Import Wizard - Step 5



Import WSDL(s)

Steps

1. Specify Location Type
2. Select WSDL(s)
3. Import Preview
4. Project Explorer Preview
5. **Summary**

Summary

WSDLs Imported successfully

C:\JavaCAPSCommon\sample projects\stockquote2.wsdl

Warnings

Errors

WSDLs Imported unsuccessfully

Details

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 450). The editor appears as shown in the following sections; icons appearing in the toolbar are described in Table 140.

Figure 450 Web Service Connection

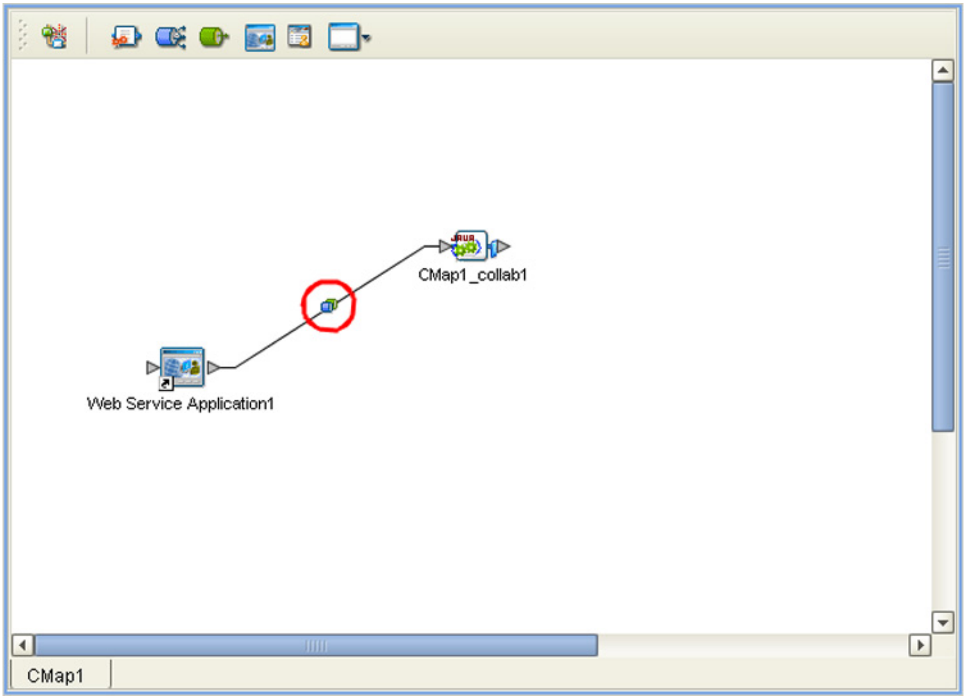


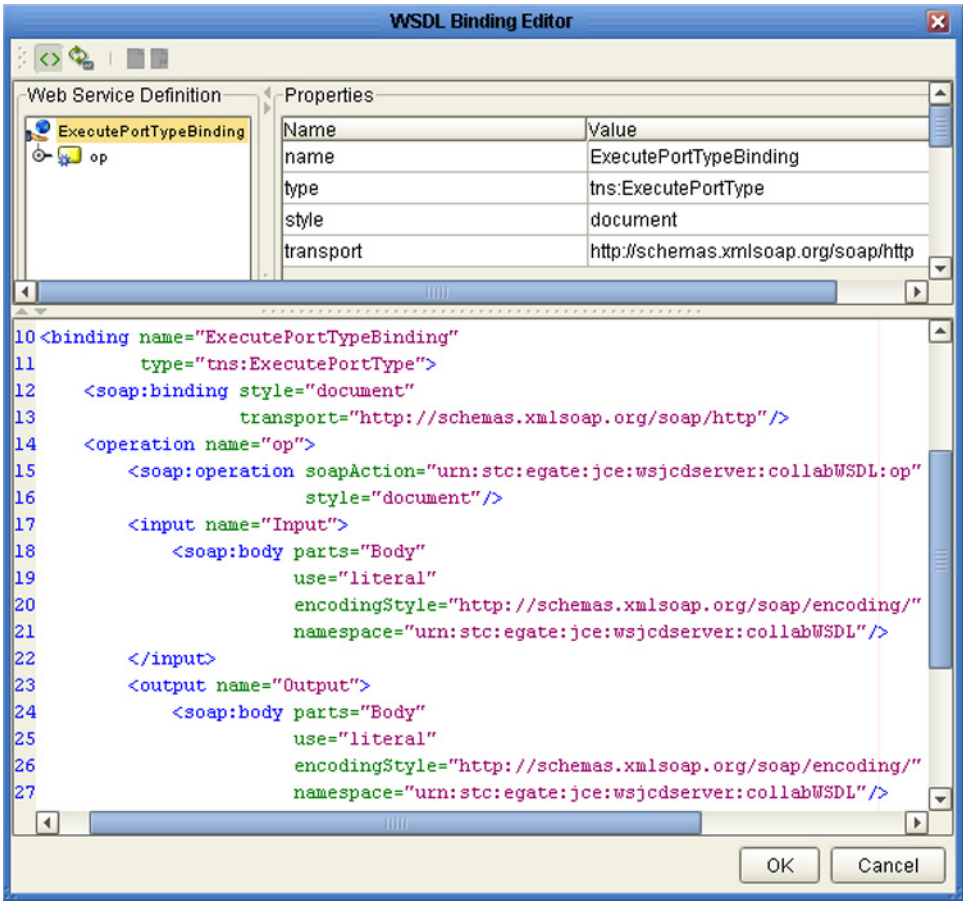
Table 140 WSDL Binding Editor Toolbar Icons

Icon	Command	Action
<>	Show Web Service Definition Code	Displays the WSDL code (after generation) in the lower half of the WSDL Binding Editor (see **Figure 451 on page 505**).
⟳	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 **Figure 454 on page 508**.)

Table 140 WSDL Binding Editor Toolbar Icons

Icon	Command	Action
N	Add Header Fault	Adds a fault (error message) to the selected header. (Enabled only for input or output message headers; see **Figure 456 on page 510**.)

Figure 451 WSDL Binding Editor Displaying Code



Name	Value
name	ExecutePortTypeBinding
type	tns:ExecutePortType
style	document
transport	http://schemas.xmlsoap.org/soap/http

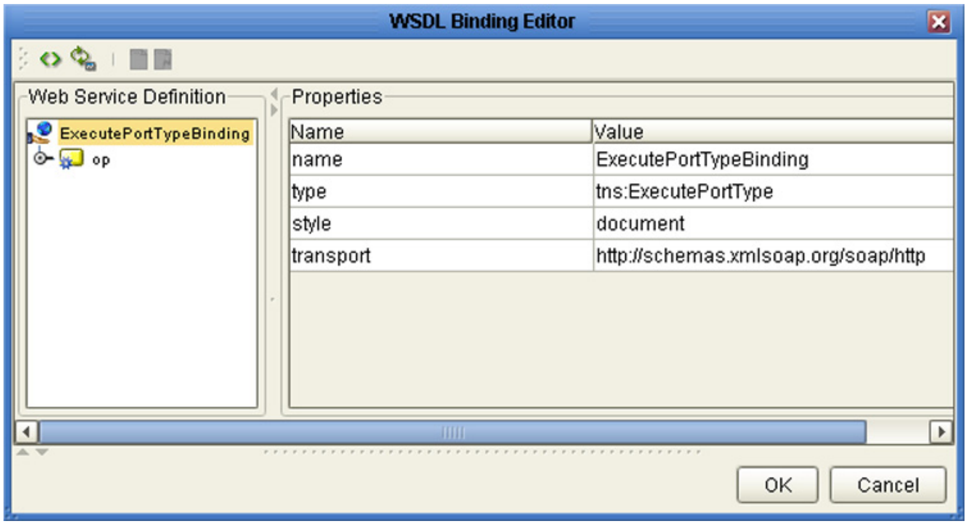
```
<binding name="ExecutePortTypeBinding"
         type="tns:ExecutePortType">
    <soap:binding style="document"
                  transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="op">
        <soap:operation soapAction="urn:stc:egate:jce:wsjcdserver:collabWSDL:op"
                        style="document"/>
        <input name="Input">
            <soap:body parts="Body"
                       use="literal"
                       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                       namespace="urn:stc:egate:jce:wsjcdserver:collabWSDL"/>
        </input>
        <output name="Output">
            <soap:body parts="Body"
                       use="literal"
                       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                       namespace="urn:stc:egate:jce:wsjcdserver:collabWSDL"/>
```

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 452 WSDL Binding Editor - PortType Binding Properties



Name	Value
name	ExecutePortTypeBinding
type	tns:ExecutePortType
style	document
transport	http://schemas.xmlsoap.org/soap/http

Table 141 PortType Binding Property Options

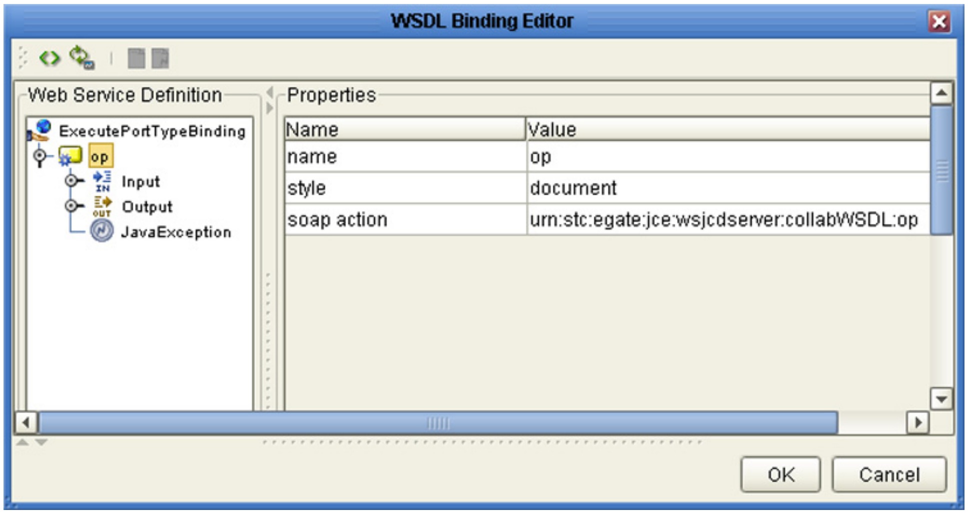
Name	Description
name	Name of the PortType Binding. The default is **ExecutePortTypeBinding**.
type	The type of binding. The default is **tns:ExecutePortType**, and cannot be changed.
style	Declares whether the interaction is document passing (**document**) or request/response (**rpc**). The default is **document**.
transport	Specifies the URL where the transport implementation can be found. The default is **http://schemas.xmlsoap.org/soap/http**.

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 453 WSDL Binding Editor - Operation Properties



Name	Value
name	op
style	document
soap action	urn:stc:egate:jce:wsjcdserver:collabWSDL:op

Table 142 Operation Property Options

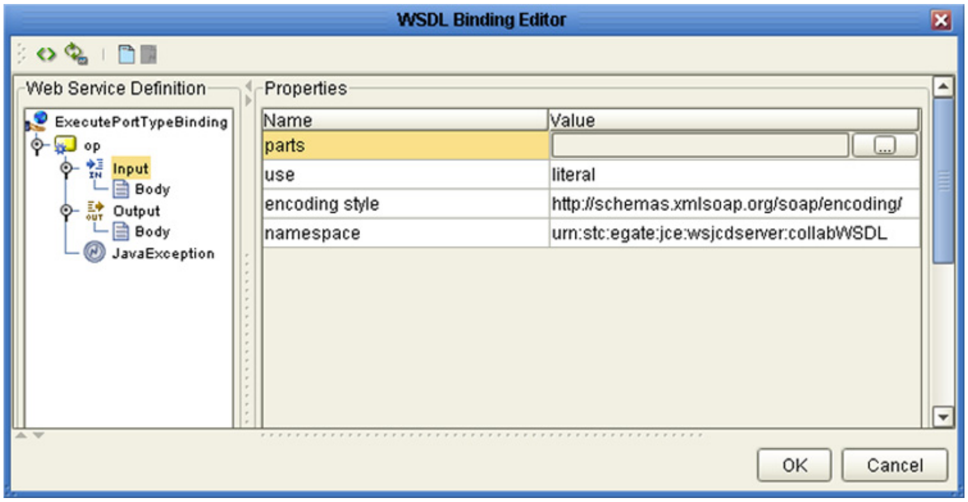
Name	Description
name	Name of the operation. The default is **op**, and cannot be changed.
style	Declares whether the interaction is document passing (**document**) or request/response (**rpc**). The default is **document**.
soap action	Specifies the Uniform Resource Name (URN) for the action. The default is **urn:stc:egate:jce.wsjcdserver:collabWSDL:op**.

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 454 WSDL Binding Editor - Message Properties



Name	Value
parts	...
use	literal
encoding style	http://schemas.xmlsoap.org/soap/encoding/
namespace	urn:stc:egate:jce:wsjcdserver:collabWSDL

Table 143 Message Property Options

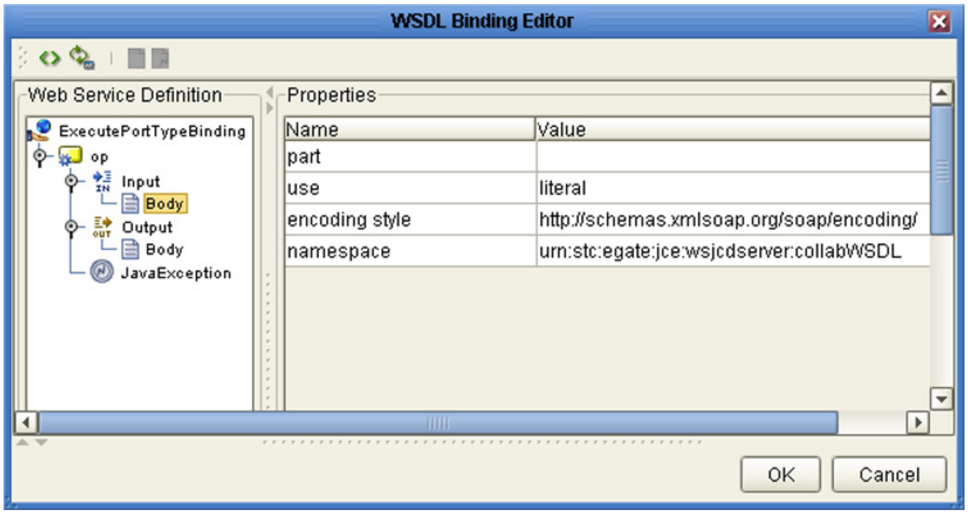
Name	Description
part	Clicking the ellipsis (**...**) button displays a dialog containing a list of existing parts, from which you can select a part to associate with this message.
use	Allowed values are **encoded** and **literal**. The default is **literal**.
encoding style	Specifies the Uniform Resource Locator (URL) referencing the optional SOAP encoding style to be used in marshaling the associated message. The default is **http://schemas.xmlsoap.org/soap/encoding/**.
namespace	Specifies the Uniform Resource Name (URN) for the application-defined namespace. The default is **urn:stc:egate:jce.wsjcdserver:collabWSDL**.

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 455 WSDL Binding Editor - Message Body Configuration



Name	Value
part	
use	literal
encoding style	http://schemas.xmlsoap.org/soap/encoding/
namespace	urn:stc:egate:jce:wsjcdserver:collabWSDL

Table 144 Message Body Property Options

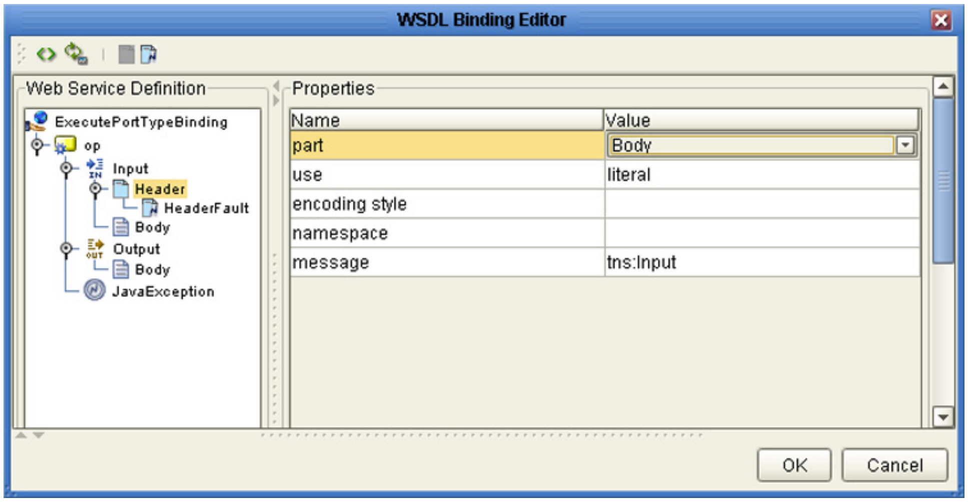
Name	Description
part	Clicking the ellipsis (**...**) button displays a dialog containing a list of existing parts, from which you can select a part to associate with this message body.
use	Allowed values are **encoded** and **literal**. The default is **literal**.
encoding style	Specifies the Uniform Resource Locator (URL) referencing the optional SOAP encoding style to be used in marshaling the message body. The default is **http://schemas.xmlsoap.org/soap/encoding/**.
namespace	Specifies the Uniform Resource Name (URN) for the application-defined namespace. The default is **urn:stc:egate:jce.wsjcdserver:collabWSDL**.

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 456 WSDL Binding Editor - Input Header Properties



WSDL Binding Editor

Web Service Definition

- ExecutePortTypeBinding
  - op
    - Input
      - Header
        - HeaderFault
      - Body
    - Output
      - Body
    - JavaException

Properties

Name	Value
part	Body
use	literal
encoding style	
namespace	
message	tns:Input

Table 145 Header Property Options

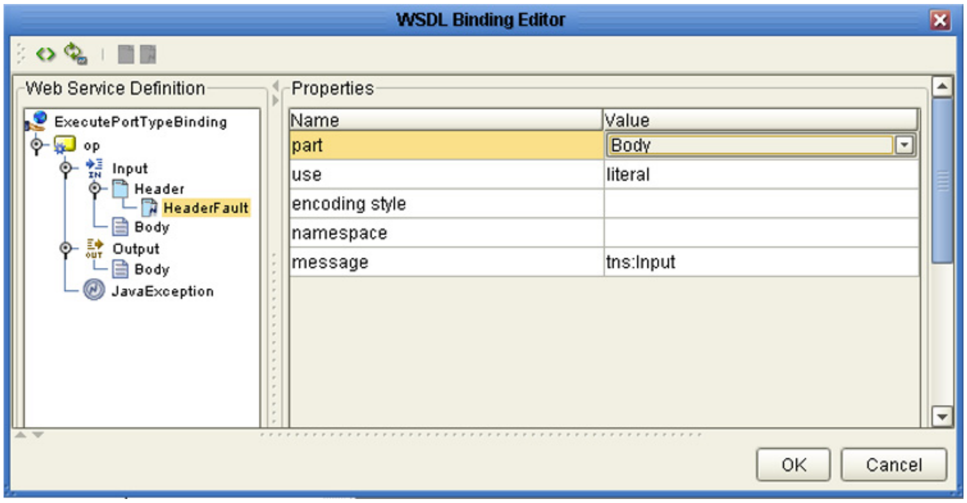
Name	Description
part	Clicking the arrow button displays a menu listing allowed parts to associate with this message header.
use	Allowed values are **encoded** and **literal**. The default is **literal**.
encoding style	Specifies the Uniform Resource Locator (URL) referencing the optional SOAP encoding 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 **tns:Input** and **tns:Output**.

12.7.6 Configuring a Header Fault

Header Fault

Fault messages can optionally be added to a message header.

Figure 457 WSDL Binding Editor - Header Fault Properties



Name	Value
part	Body
use	literal
encoding style	
namespace	
message	tns:Input

Table 146 Header Fault Property Options

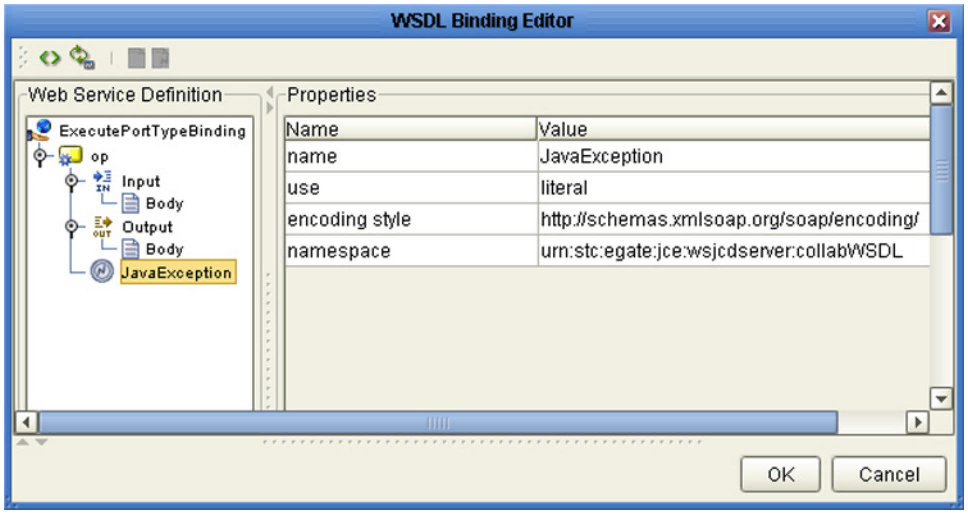
Name	Description
part	Clicking the arrow button displays a menu listing allowed parts to associate with this header fault.
use	Allowed values are **encoded** and **literal**. The default is **literal**.
encoding style	Specifies the Uniform Resource Locator (URL) referencing the optional SOAP encoding 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 458 WSDL Binding Editor - Java Exception Properties



Name	Value
name	JavaException
use	literal
encoding style	http://schemas.xmlsoap.org/soap/encoding/
namespace	urn:stc:egate:jce:wsjcdserver:collabWSDL

Table 147 Java Exception Property Options

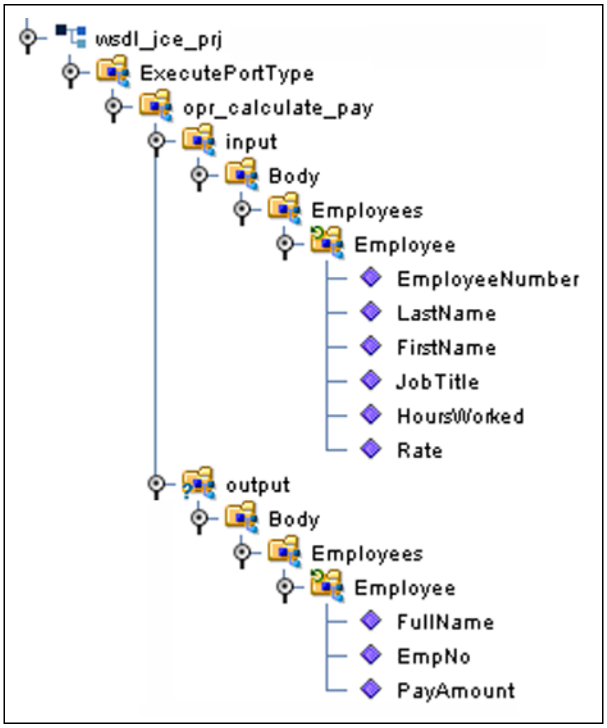
Name	Description
name	Name of the exception. The default is **JavaException**, and cannot be changed.
use	Allowed values are **encoded** and **literal**. The default is **literal**.
encoding style	Specifies the Uniform Resource Locator (URL) referencing the optional SOAP encoding style to be used in marshaling the exception message. The default is **http://schemas.xmlsoap.org/soap/encoding/**.
namespace	Specifies the Uniform Resource Name (URN) for the application-defined namespace. The default is **urn:stc:egate:jce.wsjcdserver:collabWSDL**.

12.8 Invoking Web Services From Collaboration Definitions

You can invoke one or more web services from within a Java-based Collaboration Definition if you include their WSD Objects in the Collaboration Definition when you create it using the Collaboration Definition Wizard (see **Creating a Java-based Collaboration Definition** on page 326). When you include these WSD Objects:

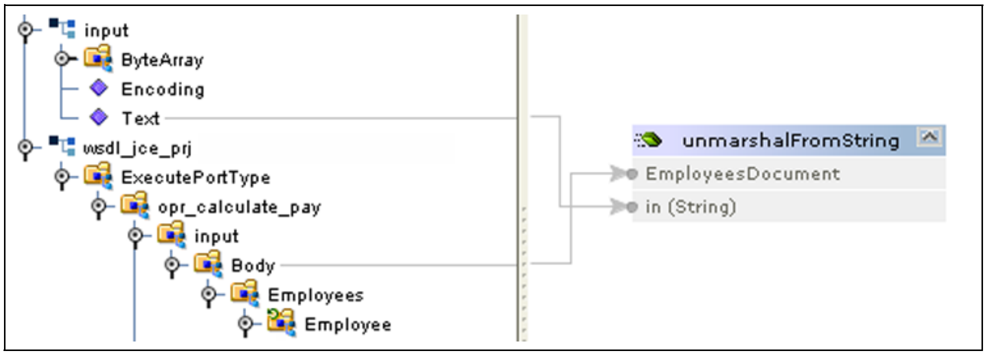
- You can select the input and output messages of all operations from all included WSD Objects in the Collaboration Definition Editor's Business Rules Designer (see Figure 459).

Figure 459 Example WSD Object in Business Rules Designer



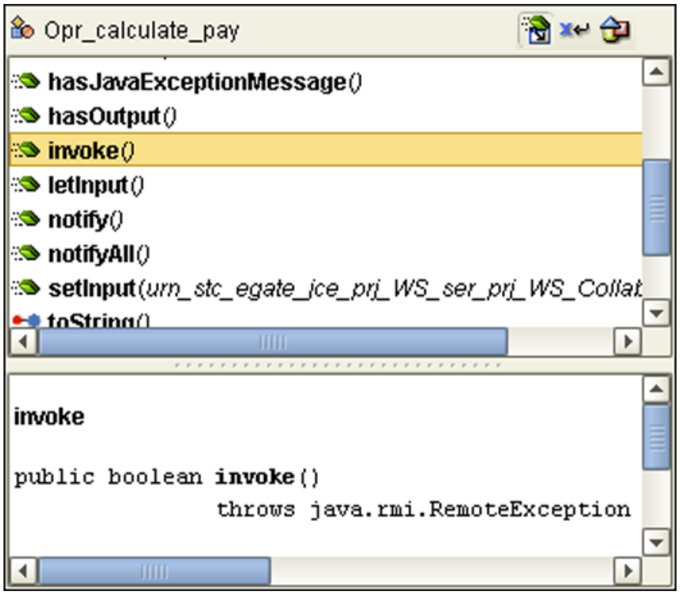
- You can use any Java operator when assigning values to, or extracting values from, web service messages (see Figure 460).

Figure 460 Example Java Operator - unmarshalFromString



- You can arbitrarily invoke one or more web service operations from the available WSD Objects, by selecting the **invoke()** method from the Method Browser (see Figure 461). At run time, these operations are invoked as modeled in the Collaboration Definition.

Figure 461 Method Browser - Invoke Method

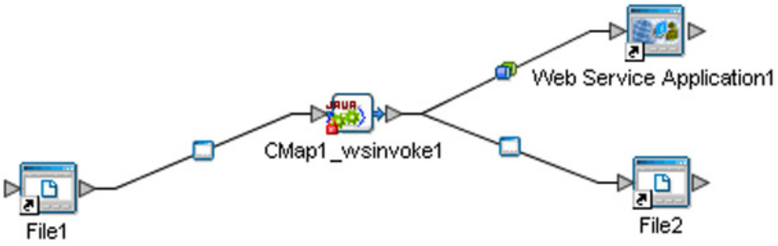

Opr_calculate_pay
hasJavaExceptionMessage()
hasOutput()
invoke()
letInput()
notify()
notifyAll()
setInput(urn_stc_egate_jce_prj_WS_ser_prj_WS_Collat
toString()

invoke

```
public boolean invoke()
                 throws java.rmi.RemoteException
```

- You can connect the WSD Objects to Web Service External Applications via the Collaboration Definition in the Connectivity Map Editor (see Figure 462).

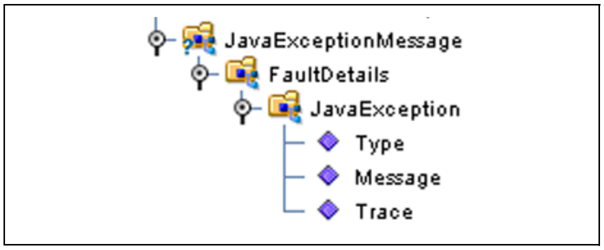
Figure 462 Connecting WSD Objects to Web Service External Applications



- If the WSD Object for the web services operation being used contains a Binding section, then that binding information is assumed automatically.
- If the WSD Object for the web services operation being used does not contain a Binding section, then then the default binding configuration is used and the WSDL Binding Editor is displayed for customizing the configuration (see **Configuring WSDL Binding Properties** on page 504).

- You can convert SOAP faults to Java exceptions containing information identifying the operations that failed, along with the attributes of the SOAP faults (see Figure 463). At run time, SOAP faults redirect control to the appropriate Java exception catch blocks.

Figure 463 Java Exception



Chapter 13

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 516
- **Using the Environment Editor** on page 517
- **Using Environment Explorer** on page 518
- **Environment Components** on page 520
- **Environmental Constants** on page 523
- **Keystores** on page 525
- **Logical Hosts and Domains** on page 526
- **Integration Servers** on page 529
- **Message Servers** on page 532
- **Schedulers** on page 535
- **SOAP/HTTP Web Service External Systems** on page 536
- **UDDI External Systems** on page 550

13.1 Overview

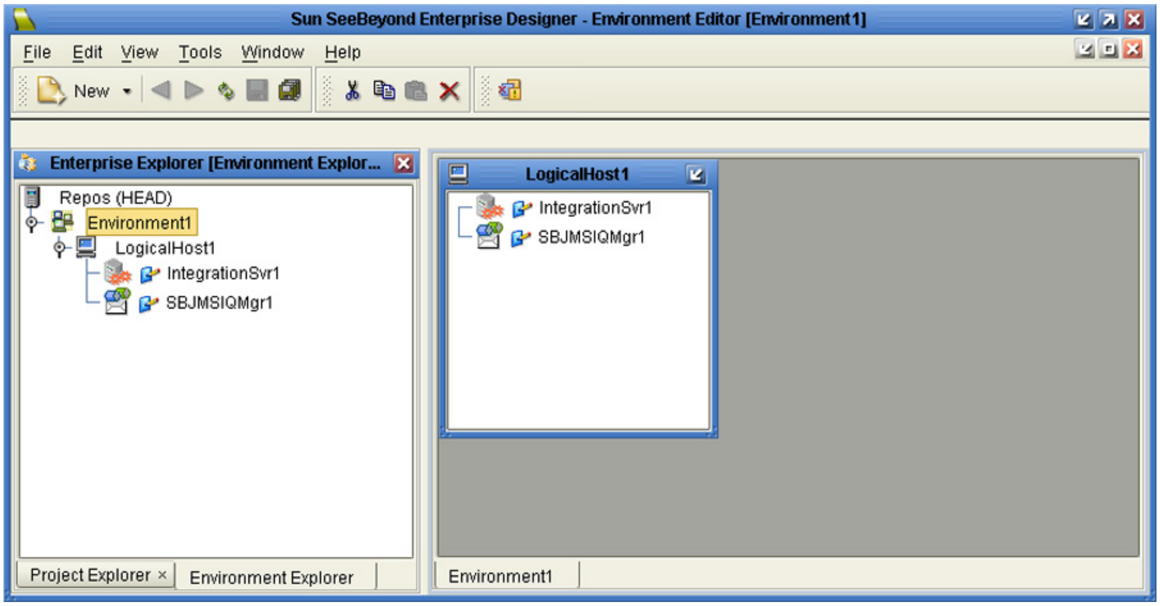
Projects are run within *domains*, which contain the logical resources required by the Project at run time. Each domain contains a single instance of an application server and a single instance of a message server. The domains, in turn, are defined within run-time *Environments*, which represent the 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 464).

Figure 464 Environment Editor

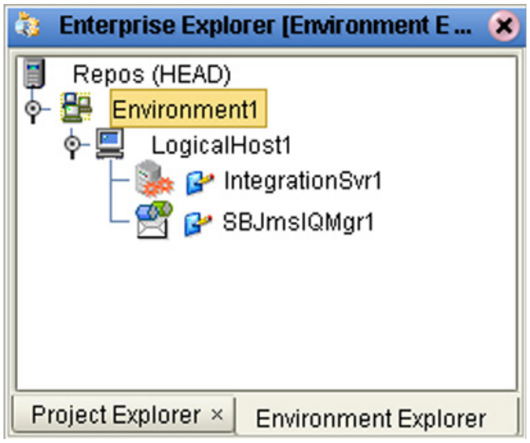


Here you can see the various components (Logical Hosts, servers, and external systems) included in the selected Environment. New Environments are added through the use of the Repository context menu (see **Using the Repository Context Menu** on page 62). Components are added to the Environment by selecting options in the Environment and Logical Host context menus (see **Environment Components** on page 520 and **Logical Hosts and Domains** on page 526, respectively).

13.3 Using Environment Explorer

Environment Explorer **deals with physical resources, including** integration and message servers (see Figure 465). 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 465 Enterprise Explorer: Environment Explorer View



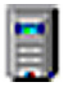
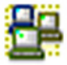
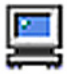
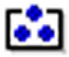
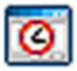
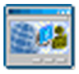
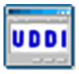
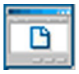
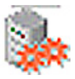
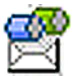
Each component in Environment Explorer has an icon to identify the component type (see **Environment Explorer Icons** on page 519).

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 148 appear in the Environment Explorer.

Table 148 Environment Icons

Icon	Function
	Represents the **Repository**, 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 **Environment**, which contains Logical Hosts and information about external systems.
	Represents a **Logical Host** or **Domain**, which contains the various logical components and files that are required at run time. See **Logical Hosts and Domains** on page 526.
	Represents an **Environmental Constant**, which you can use to automate eWay and message destination configuration changes. See **Environmental Constants** on page 523.
	Represents a **Scheduler** external system, which is an Environmental container for a Scheduler adapter. See **Schedulers** on page 123.
	Represents a **SOAP/HTTP Web Service External System**, which is a web service or web services client. See **SOAP/HTTP Web Service External Systems** on page 536.
	Represents a **UDDI External System**, which represents a UDDI Registry server. See **UDDI External Systems** on page 550.
	Represents an **External System** (such as a file) that is accessed through an eWay. See the *Sun SeeBeyond eGate™ Integrator System Administration Guide* for information on keystores, and the individual eWay User's Guides regarding the external systems.
	Represents a **Sun SeeBeyond Integration Server** or third-party **application server**, which manages the Collaborations and other process interactions of a Project. The integration server is deployed to a Logical Host. See **Integration Servers** on page 529.
	Represents a **Sun SeeBeyond JMS IQ Manager** or third-party **message server**, which is used to store and forward eGate Integrator system messages. The message server is deployed to a Logical Host. See **Message Servers** on page 532.

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 62). 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 466, 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 466 Environment Context Menu

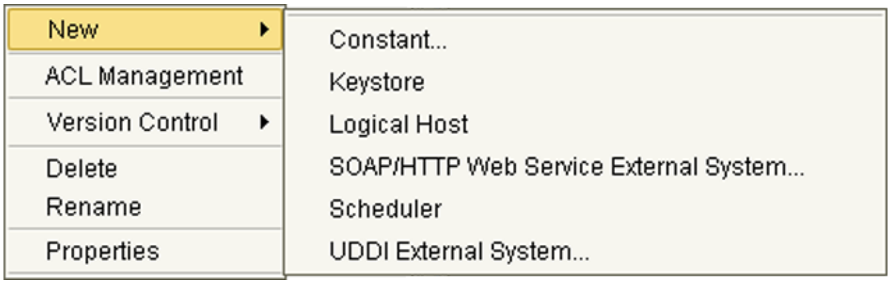


Table 149 Environment Context Menu Options

Option		Function
New	Constant	Presents a dialog with which you can add a constant to the selected Environment. See **Environmental Constants** on page 523.
	Keystore	Adds a new keystore to the selected Environment. See **Keystores** on page 525
	Logical Host	Adds a new Logical Host to the selected Environment. See **Logical Hosts and Domains** on page 526.
	SOAP/HTTP Web Service External System	Adds a SOAP/HTTP Web Service External System to the selected Environment. See **SOAP/HTTP Web Service External Systems** on page 536.
	Scheduler	Presents a dialog with which you can add a new scheduling component to the selected Environment. See **Schedulers** on page 535.
	UDDI External System	Adds a UDDI External System to the selected Environment. See **UDDI External Systems** on page 550.

Table 149 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 *Sun SeeBeyond eGate™ Integrator System Administration Guide*.
Version Control	Check In	Presents a dialog with which you can check in a new version of the selected Environment. Refer to **Checking a Component In** on page 83 for more details.
	Check Out	Presents a dialog with which you can check out the current version of the selected Environment. See **Checking a Component Out** on page 85 for more information.
	Tag	Presents a dialog with which you can specify a tag to attach to the selected Environment.
Delete		Deletes the selected Environment, subject to the following conditions: ▪ You have *write* privileges for Environments (see ACL Management, above). ▪ The Environment is not checked out by anyone other than yourself. If these conditions are met, a dialog is presented in which you confirm that you want to delete the selected Environment. Clicking **Yes** then deletes the Environment.
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 467. Those context menus that are different are described in the sections for the specific components.

Figure 467 Environment Component Context Menu

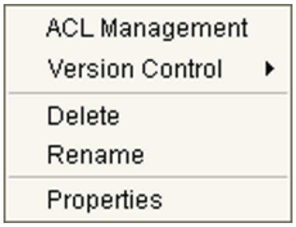


Table 150 Environment Component Context Menu Options

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 *Sun SeeBeyond eGate™ Integrator System Administration Guide*.
Version Control	Accesses the version control features for the selected component. See **Accessing Component Version Control Features** on page 82 for additional information.
Delete	Deletes the selected component, subject to the following conditions: ▪ You have *write* privileges for the component (see ACL Management, above). ▪ You have the component checked out to your workspace. ▪ The component is not checked out by anyone other than yourself. If these conditions are met, a dialog is presented in which you confirm that you want to delete the selected component. Clicking **Yes** 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 468) displays the *Environmental Constants* panel shown in Figure 469.

Figure 468 Environment Context Menu - New Constant

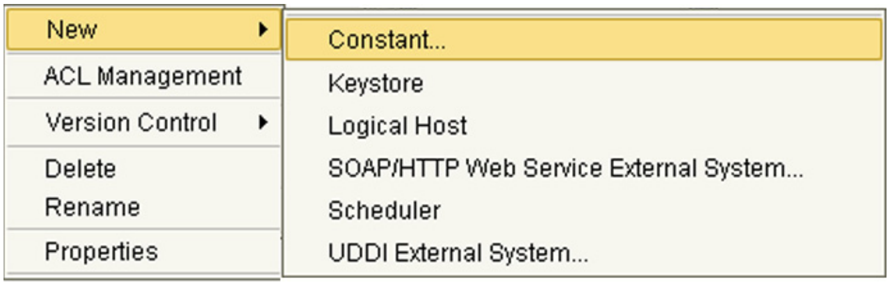
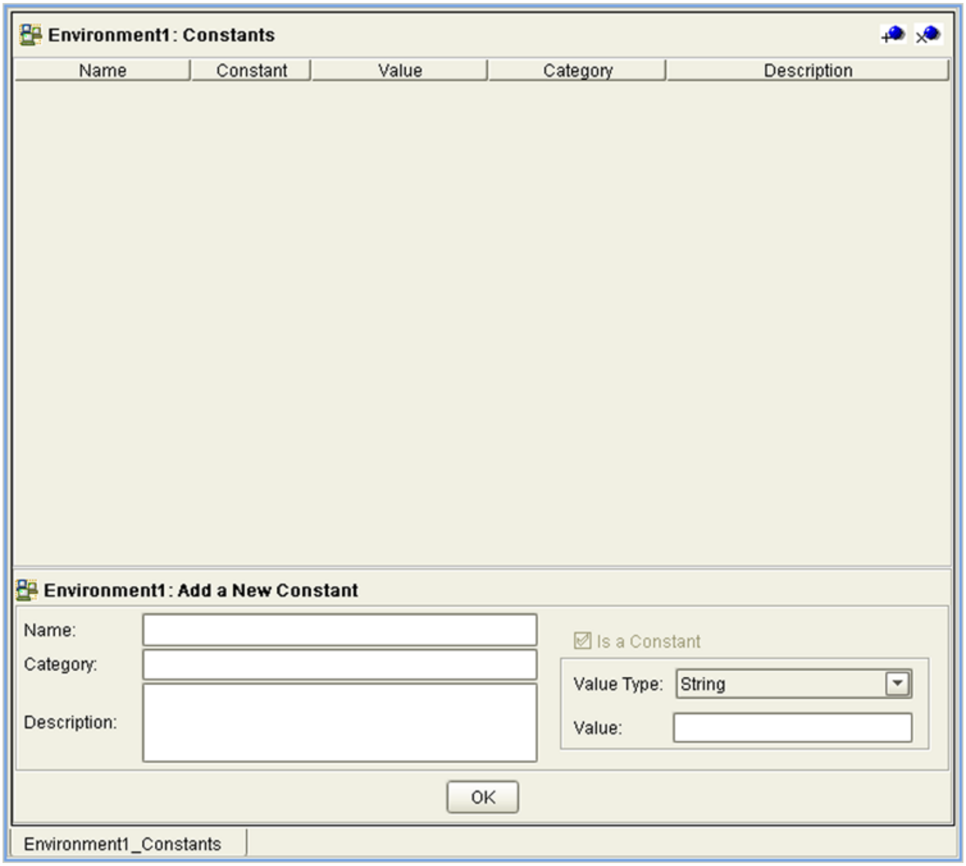


Figure 469 Environmental Constants Panel



☑ Is a Constant

Value Type: String

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 151 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 522).

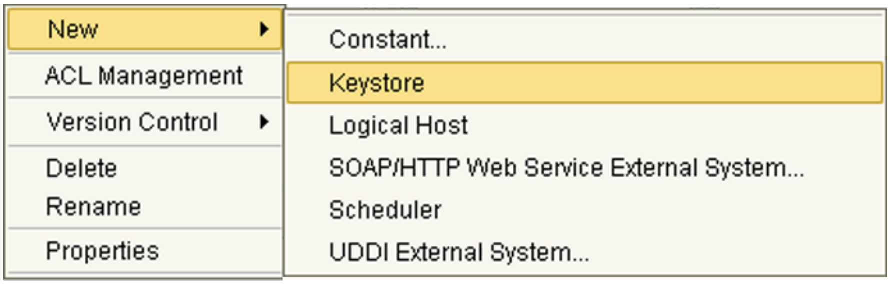
13.6 Keystores

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 470) 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 470 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 471).

Figure 471 Keystore Context Menu

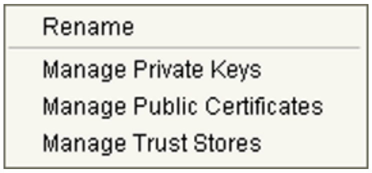


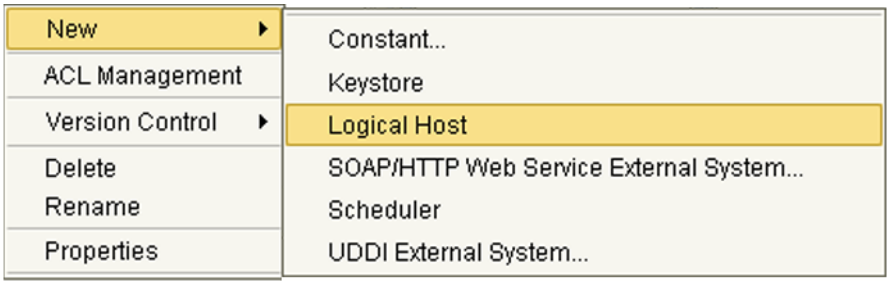
Table 152 Keystore Context Menu Options

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 472).

Figure 472 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 473.

Figure 473 Logical Host Context Menu

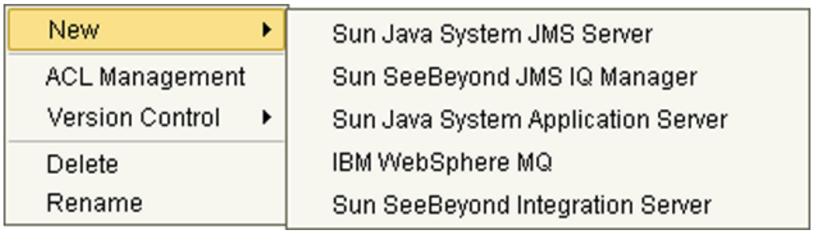

New
ACL Management
Version Control
Delete
Rename
Sun Java System JMS Server
Sun SeeBeyond JMS IQ Manager
Sun Java System Application Server
IBM WebSphere MQ
Sun SeeBeyond Integration Server

Table 153 Logical Host Context Menu Options

Option		Function
New	Sun Java System JMS Server	Adds a new Sun Java System Message Queue to the selected Logical Host. See the *Sun SeeBeyond eGate™ Integrator JMS Reference Guide*.
	Sun SeeBeyond JMS IQ Manager	Adds a new Sun SeeBeyond JMS IQ Manager to the selected Logical Host. See **Message Servers** on page 532.
	Sun Java System Application Server	Adds a new Sun Java System Application Server to the selected Logical Host. See the *Sun SeeBeyond eGate™ Integrator JMS Reference Guide*.
	WebSphere MQ	Adds a new IBM WebSphere MQ message server to the selected Logical Host. See **Message Servers** on page 532.
	Sun SeeBeyond Integration Server	Adds a new Sun SeeBeyond Integration Server to the selected Logical Host. See **Integration Servers** on page 529.
ACL Management		Presents a dialog with which an Administrator can assign read/write privileges to users for the selected Logical Host. See the *Sun SeeBeyond eGate™ Integrator System Administration Guide*.
Version Control		Accesses the version control features for the selected component. See **Accessing Component Version Control Features** on page 82 for additional information.
Delete		Deletes the selected Logical Host, subject to the following conditions: ▪ You have *write* privileges for the component (see ACL Management, above). ▪ You have the component checked out to your workspace. ▪ The component is not checked out by anyone other than yourself. If these conditions are met, a dialog is presented in which you confirm that you want to delete the selected component. Clicking **Yes** then deletes the component.
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*.

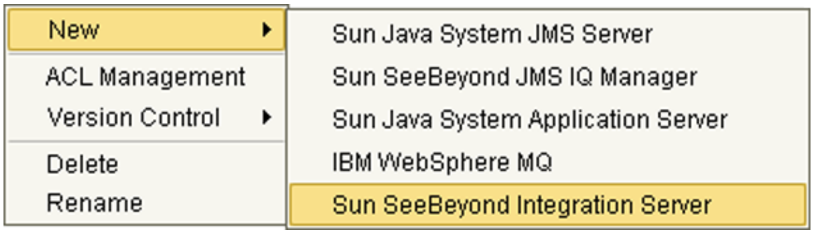
13.8 Integration Servers

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 474).

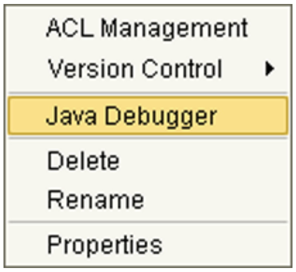
Figure 474 Logical Host Menu - New Integration Server


New
ACL Management
Version Control
Delete
Rename
Sun Java System JMS Server
Sun SeeBeyond JMS IQ Manager
Sun Java System Application Server
IBM WebSphere MQ
Sun SeeBeyond 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 522). 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 475. 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 313). This menu is used for all supported integration servers.

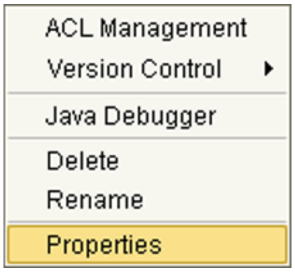
Figure 475 Integration Server Context Menu - Java Debugger



13.8.2 Configuring an Integration Server

Selecting **Properties** from the integration server context menu (see Figure 476) displays the dialog shown in Figure 477. This basic dialog is used for all supported integration servers. The configuration property values, however, are different for different servers. The dialog for a Sun SeeBeyond Integration Server is used as an example.

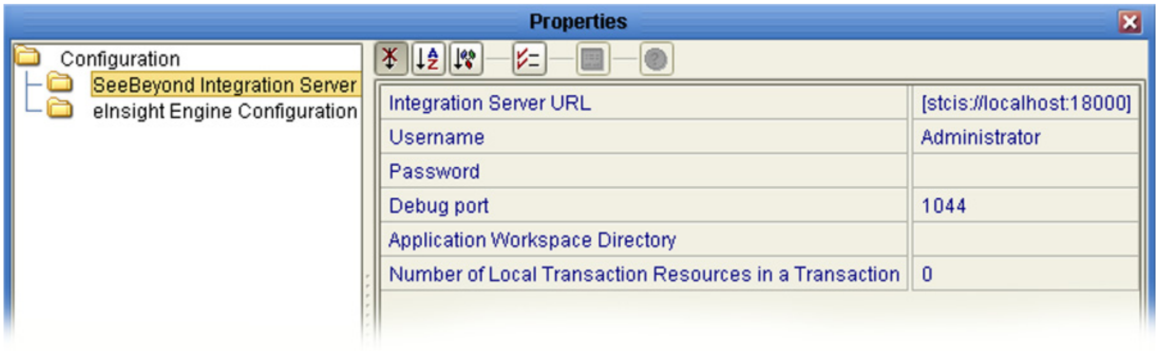
Figure 476 Integration Server Context Menu - Properties



Sun SeeBeyond Integration Server Configuration

The properties shown in this dialog are used only for generating application files (see **Building an Application File** on page 148); they do not affect operation of the integration server itself. Configuration of a Sun SeeBeyond Integration Server for runtime operation is performed from Enterprise Manager, and is described in the *Sun SeeBeyond eGate™ Integrator System Administration Guide*.

Figure 477 Sun SeeBeyond Integration Server Configuration Properties



Integration Server URL	[stcis://localhost:18000]
Username	Administrator
Password	
Debug port	1044
Application Workspace Directory	
Number of Local Transaction Resources in a Transaction	0

Table 154 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 **stcis://localhost:18000**.
Username	User name for accessing the Integration Server. The default value is **Administrator**.

Table 154 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 *Sun SeeBeyond eGate™ Integrator System Administration Guide*).
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 implements the Java Messaging Service (JMS). By default, Enterprise Designer offers three built-in message server options:

- Sun SeeBeyond JMS IQ Manager
- Sun Java™ System JMS Server (Sun Java System Message Queue)
- IBM WebSphere MQ (version 6.0 only)

Sun SeeBeyond JMS IQ Manager is eGate Integrator's native JMS message server implementation, and is described further in the *Sun SeeBeyond eGate™ Integrator JMS Reference Guide*. Information regarding the Sun Java System Message Queue and the IBM WebSphere MQ is found in the documentation supplied with those products.

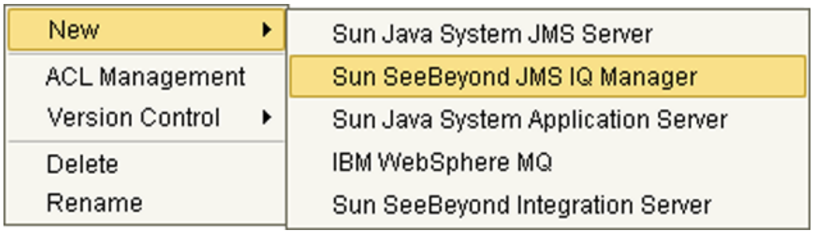
Another option, which must be installed separately, is the Sun Java Message Service Grid (JMS Grid). Procedures for installing and configuring JMS Grid for use with Enterprise Designer are summarized in the *Sun SeeBeyond eGate™ Integrator JMS Reference Guide*. For information on JMS Grid itself, see the *Sun Java™ Message Service Grid User's Guide*.

To support legacy SeeBeyond Projects, eGate Integrator includes the Schema Run-time Environment (SRE) JMS IQ Manager, which also must be installed separately. See the *Sun SeeBeyond eGate™ Integrator JMS Reference Guide* for information.

eGate Integrator also supports selected third-party application servers, which contain their own message servers (see the *Sun SeeBeyond eGate™ Integrator System Administration Guide* for details).

You add the message server component to the Environment 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 482).

Figure 478 Logical Host Menu - New Message Server


New
ACL Management
Version Control
Delete
Rename
Sun Java System JMS Server
Sun SeeBeyond JMS IQ Manager
Sun Java System Application Server
IBM WebSphere MQ
Sun SeeBeyond Integration Server

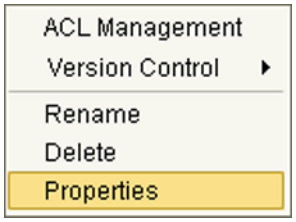
Entering a name and clicking **OK** adds the message server to the selected Environment. Right-clicking the message server in Environment Explorer displays its context menu (see Figure 479).

Note: *If you do not see the option for the message server you need, you have not installed the **.sar** file for that message server. For information about installing **.sar** files, refer to the Sun Java™ Composite Application Platform Suite Installation Guide.*

13.9.1 Configuring a Message Server

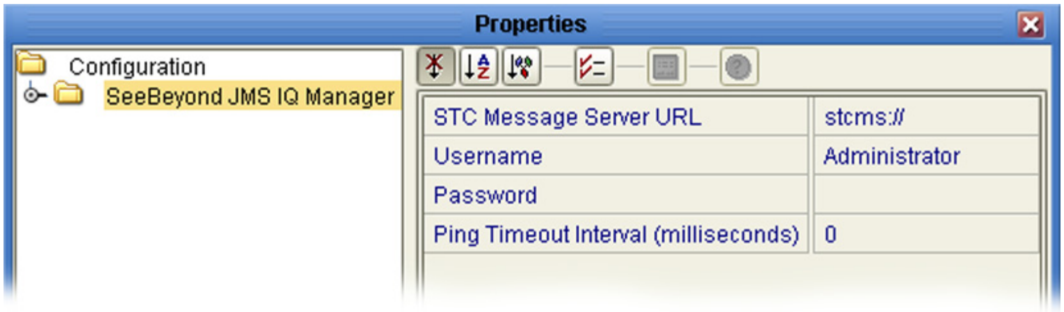
Selecting **Properties** from the message server context menu (see Figure 479) displays the dialog shown in Figure 480. This basic dialog is used for all supported message servers. The configuration property values, however, are different for different servers. The dialog for a Sun SeeBeyond JMS IQ Manager is used as an example.

Figure 479 Message Server Context Menu - Properties



Sun SeeBeyond JMS IQ Manager Configuration

Figure 480 Sun SeeBeyond JMS IQ Manager Configuration Properties



STC Message Server URL	stcms://
Username	Administrator
Password	
Ping Timeout Interval (milliseconds)	0

The properties shown in this dialog are used only for generating application files (see **Building an Application File** on page 148); they do not affect operation of the message server itself. Configuration of a Sun SeeBeyond JMS IQ Manager for runtime operation is performed from Enterprise Manager, and is currently described in the *Sun SeeBeyond eGate™ Integrator JMS Reference Guide*.

Table 155 JMS Message Server Configuration Properties List

Property	Description
<Message Server> URL	URL of the message server to be used for the current Project deployment, showing the host and port on which the message server is running. The format is: stcms://*host*:*port* By default, Projects connect to the message server associated with the application server specified for the Project. Specify the URL if you want the Project to use a different message server. The URL can be appended with a string that specifies the re-delivery characteristics after messages are rolled back. See the *Sun SeeBeyond eGate™ Integrator JMS Reference Guide*.

Table 155 JMS Message Server Configuration Properties List

Property	Description
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.

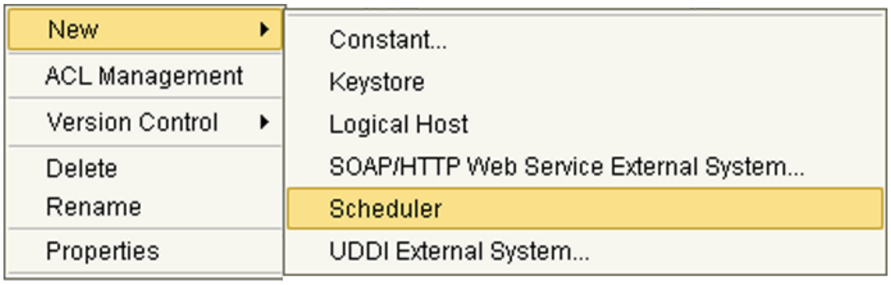
Note: *For the Sun SeeBeyond JMS IQ Manager, leave the STC Message Server URL set to the default value (**stcms://**). The associated integration server will automatically change this URL to **stcms://localhost:port**. This allows you to deploy an EAR file to any Logical Host (domain).*

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 123). You create this component by selecting the **New > Scheduler** option from the *Environment* context menu in Environment Explorer (see Figure 481).

Figure 481 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 522).

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 482), which displays the dialog shown in Figure 483.

Figure 482 Environment Context Menu - New SOAP/HTTP Web Service External System

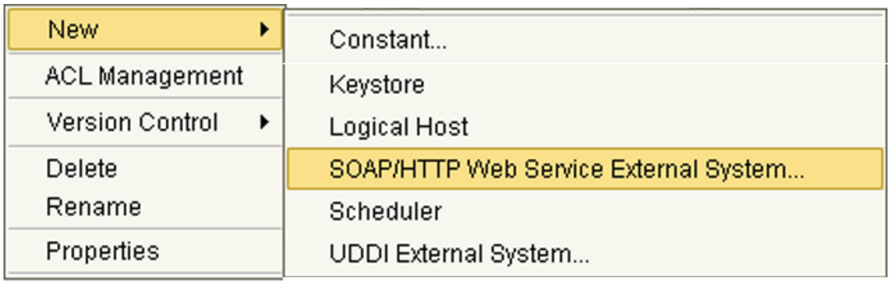
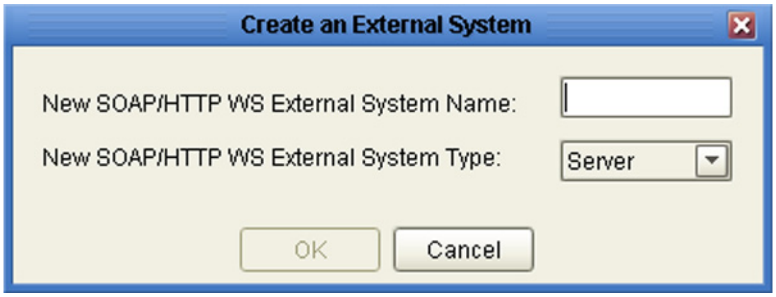


Figure 483 Create External System Dialog



New SOAP/HTTP WS External System Name:

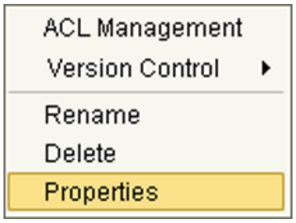
New SOAP/HTTP WS External System Type: Server

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 522).

13.11.1 Configuring a SOAP/HTTP Client

Selecting **Properties** from the **SOAP/HTTP Web Service External System (Client)** context menu (see Figure 484) displays the dialog set shown in this section, in which you assign the desired values for the listed properties.

Figure 484 SOAP/HTTP Web Service External System Context Menu - Properties



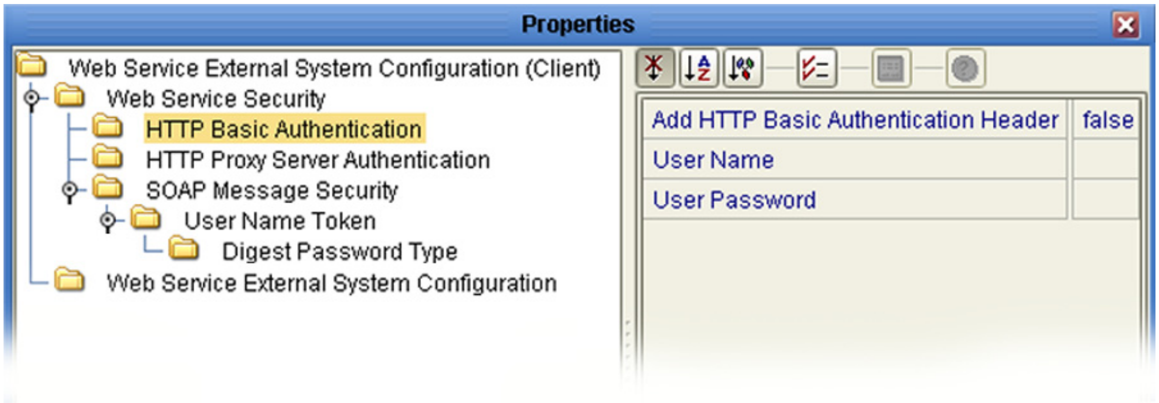
Web Service Security: HTTP Basic Authentication

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 485 HTTP Basic Authentication Configuration


Properties
Web Service External System Configuration (Client)
Web Service Security
HTTP Basic Authentication
HTTP Proxy Server Authentication
SOAP Message Security
User Name Token
Digest Password Type
Web Service External System Configuration

Add HTTP Basic Authentication Header	false
User Name	
User Password	

Table 156 HTTP Basic Authentication Configuration Properties

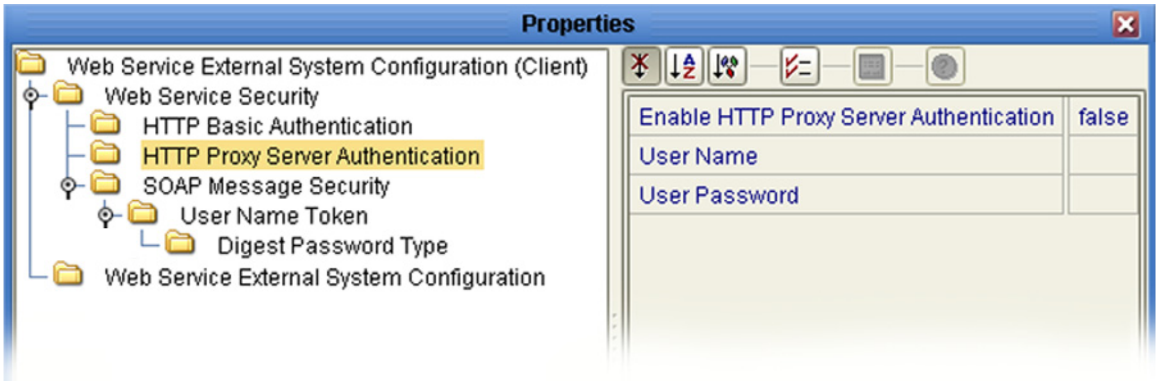
Property	Description
Add HTTP Basic Authentication Header	Specifies whether or not you want to use HTTP Basic Authentication for this web service. The default value is **false**.
User Name	Specifies the name of the Logical Host user to whom you want to grant access to the web service. This information 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 486 HTTP Proxy Server Configuration Dialog


Properties
Web Service External System Configuration (Client)
Web Service Security
HTTP Basic Authentication
HTTP Proxy Server Authentication
SOAP Message Security
User Name Token
Digest Password Type
Web Service External System Configuration

Enable HTTP Proxy Server Authentication	false
User Name	
User Password	

Table 157 HTTP Proxy Server Configuration Properties

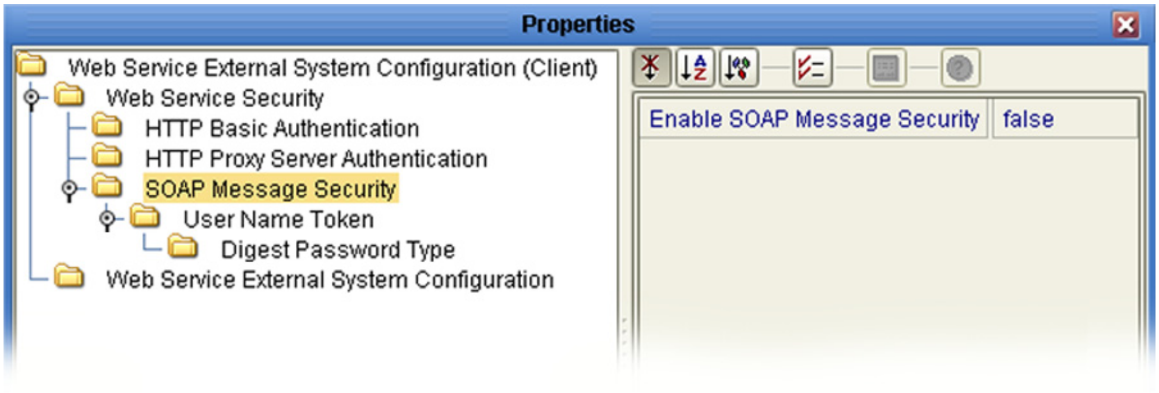
Property	Description
Enable HTTP Proxy Server Authentication	Specifies whether or not HTTP Proxy Server Authentication is to be used for this web service. The default value is **false**.
User Name	Specifies the name of the proxy server user.
User Password	Specifies the password for the proxy server user.

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 487 SOAP Message Security Configuration Dialog



Properties
Web Service External System Configuration (Client)
Web Service Security
HTTP Basic Authentication
HTTP Proxy Server Authentication
SOAP Message Security
User Name Token
Digest Password Type
Web Service External System Configuration

Enable SOAP Message Security false

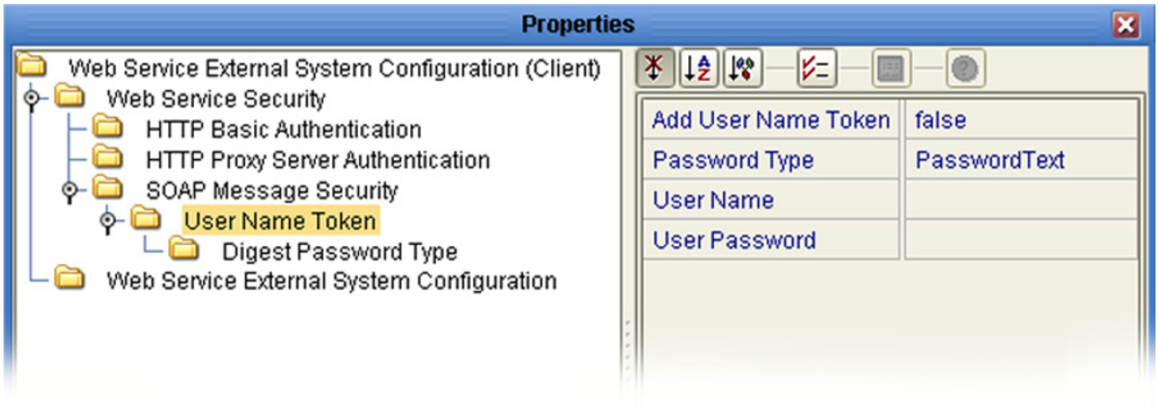
Table 158 SOAP Message Security Configuration Properties

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 **false**.

User Name Token

A user name token contains a user name and password, which can be of two types: *text* and *digest*.

Figure 488 User Name Token Configuration Dialog


Properties
Web Service External System Configuration (Client)
Web Service Security
HTTP Basic Authentication
HTTP Proxy Server Authentication
SOAP Message Security
User Name Token
Digest Password Type
Web Service External System Configuration

Add User Name Token	false
Password Type	PasswordText
User Name	
User Password	

Table 159 User Name Token Configuration Properties

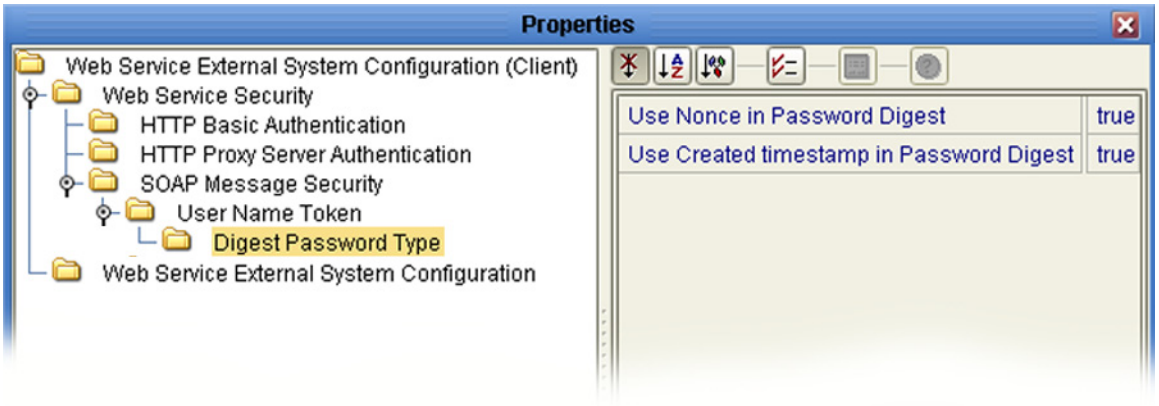
Property	Description
Add User Name Token	Specifies whether or not a User Name Token is to be used in the SOAP Message Security for this web service. The default value is **false**.
Password Type	Specifies whether you want to use a **digest** or **text** password in the token. In a **digest** type, the 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 **PasswordText**.
User Name	Specifies the name of the Logical Host user you want to use for the token.
User Password	If you specified **text** 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 *clear text* password is the original, unmodified password. A *password equivalent* 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 489 Digest Password Type Configuration Dialog


Properties
Web Service External System Configuration (Client)
Web Service Security
HTTP Basic Authentication
HTTP Proxy Server Authentication
SOAP Message Security
User Name Token
Digest Password Type
Web Service External System Configuration

Use Nonce in Password Digest	true
Use Created timestamp in Password Digest	true

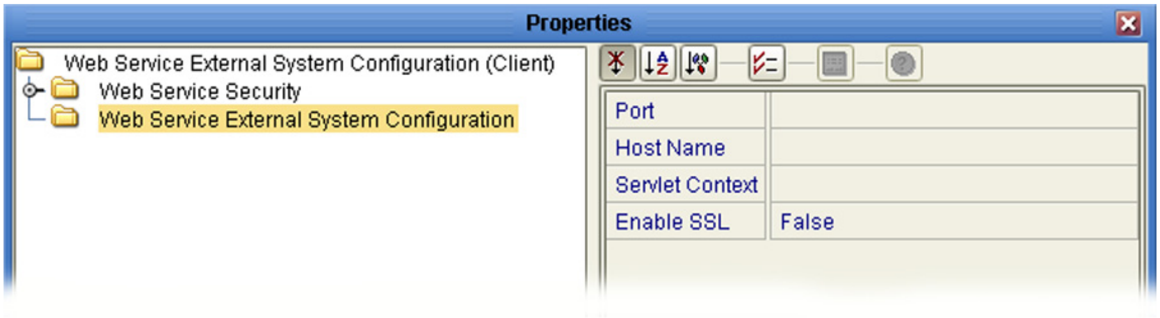
Table 160 Digest Password Type Configuration Properties

Property	Description
Use Nonce in Password Digest	If you specified **PasswordDigest** for the User Name Token password type, this property specifies whether or not you want to use a nonce in the password digest. A *nonce* is a random value that is used only once. The default value is **true**.
Use Created Timestamp in Password Digest	If you specified **PasswordDigest** 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 **true**.

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 490 Web Service External System Configuration Dialog



Properties
Web Service External System Configuration (Client)
Web Service Security
Web Service External System Configuration

Port	
Host Name	
Servlet Context	
Enable SSL	False

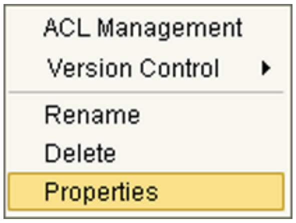
Table 161 Web Service External System Configuration Properties

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. **Note:** 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 **True**. The default is **False**.

13.11.2Configuring a SOAP/HTTP Server

Selecting **Properties** from the **SOAP/HTTP Web Service External System (Server)** context menu (see Figure 491) displays the dialog set shown in this section, in which you assign the appropriate values for the listed properties.

Figure 491 SOAP/HTTP Web Service External System Context Menu - Properties



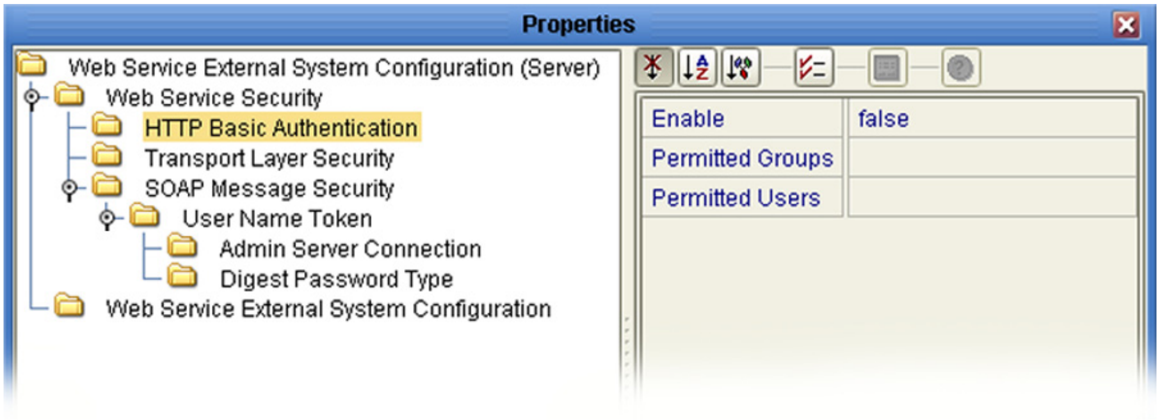
Web Service Security: HTTP Basic Authentication

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 492 HTTP Basic Authentication Configuration Dialog



Enable	false
Permitted Groups	
Permitted Users	

Table 162 HTTP Basic Authentication Configuration Properties

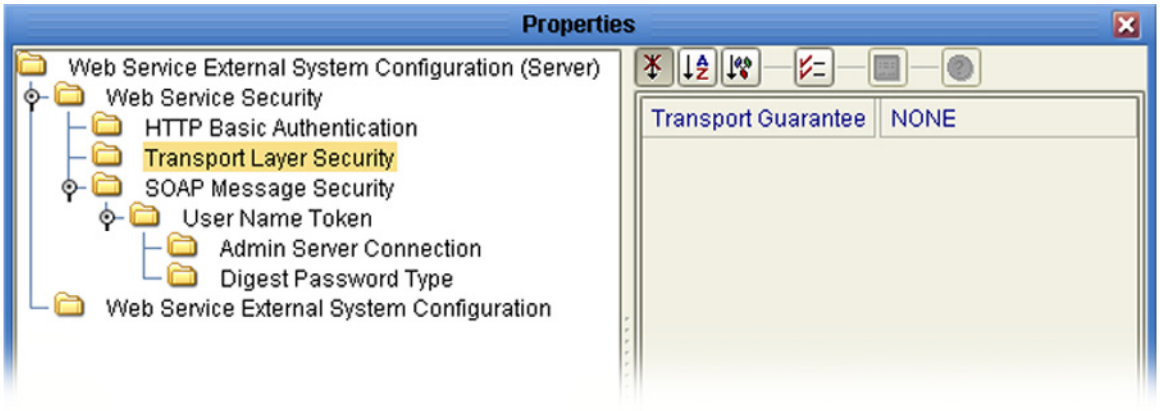
Property	Description
Enable	Specifies whether or not you want to use HTTP Basic Authentication for this web service. The default value is **false**.
Permitted Groups	Specifies the groups to which you want to grant access to the web service. Use commas to separate multiple groups.
Permitted Users	Specifies 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 493 Transport Layer Security Configuration Dialog



Transport Guarantee NONE

Table 163 Transport Layer Security Configuration Properties

Property	Description
Transport Guarantee	Specifies how strongly messages are protected in the Transport Layer. ▪ If you want messages to be visible but not modifiable, then set the value to **INTEGRAL**. ▪ If you want messages to be neither visible nor modifiable, then set the value to **CONFIDENTIAL**. ▪ The default value is **NONE**.

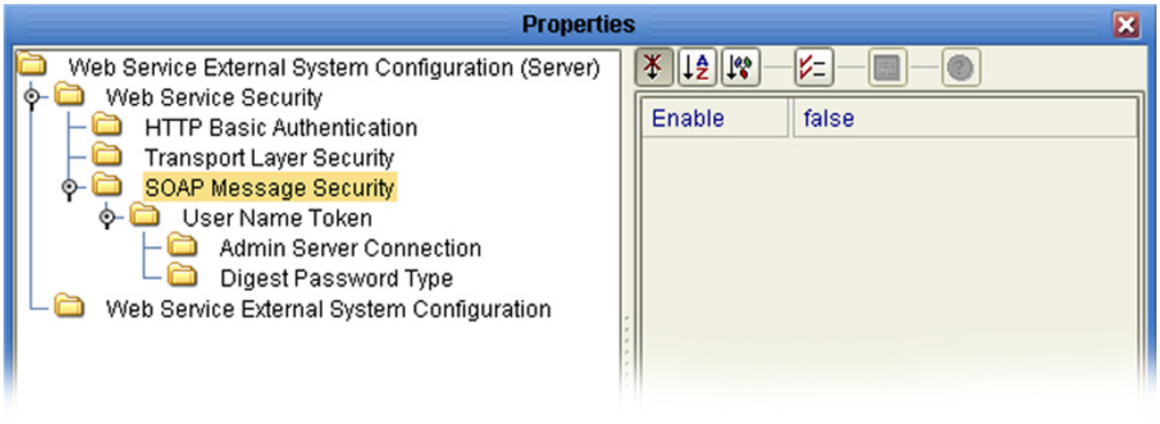
Web Service Security: SOAP Message Security

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 494 SOAP Message Security Configuration Dialog



Enable	false

Table 164 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 **false**.

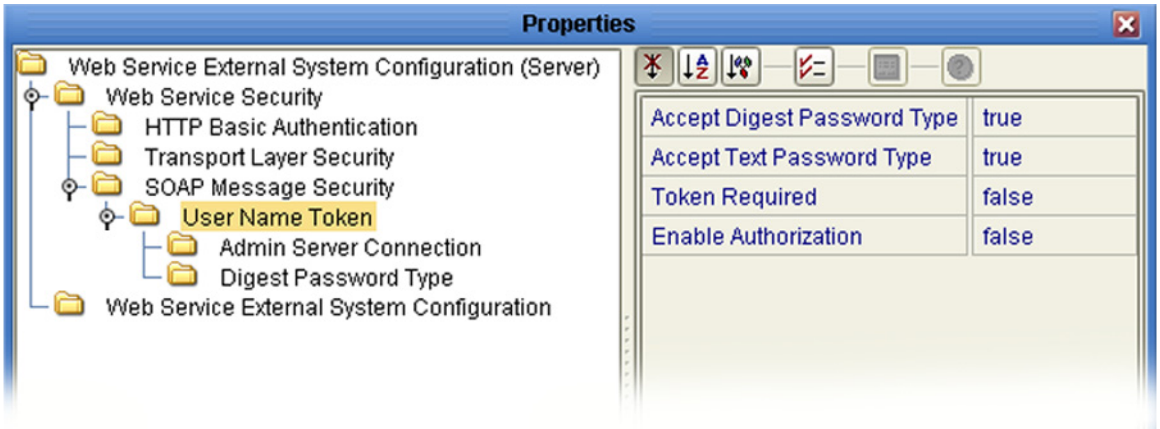
User Name Token

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 495 User Name Token Configuration Dialog



Accept Digest Password Type	true
Accept Text Password Type	true
Token Required	false
Enable Authorization	false

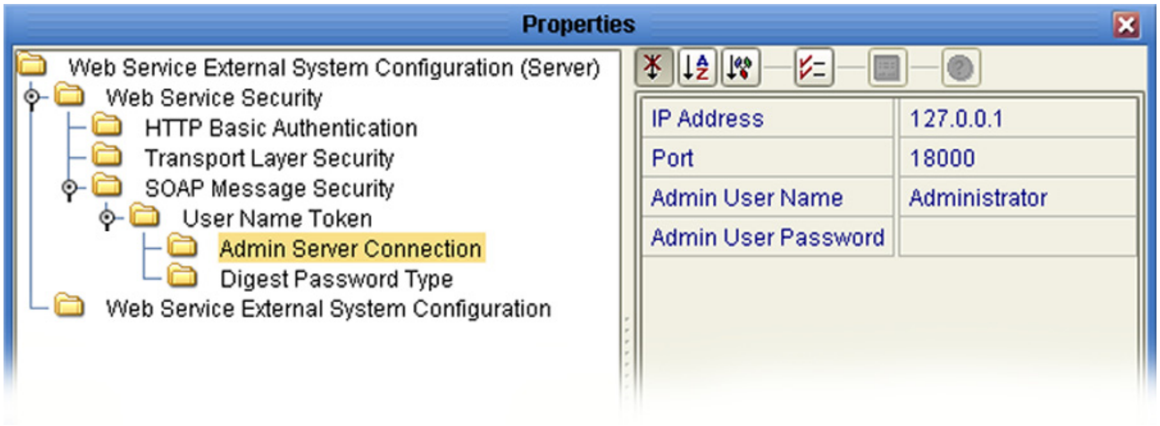
Table 165 User Name Token Configuration Properties

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 **true**.
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 **true**.
Token Required	Specifies whether or not the security header must contain a user name token. A value of **true** indicates that if the security header does not contain a user name token, then a SOAP header fault is thrown. The default value is **false**.
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 **false**.

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 496 Admin Server Connection Configuration Dialog



IP Address	127.0.0.1
Port	18000
Admin User Name	Administrator
Admin User Password	

Table 166 Admin Server Connection Configuration Properties

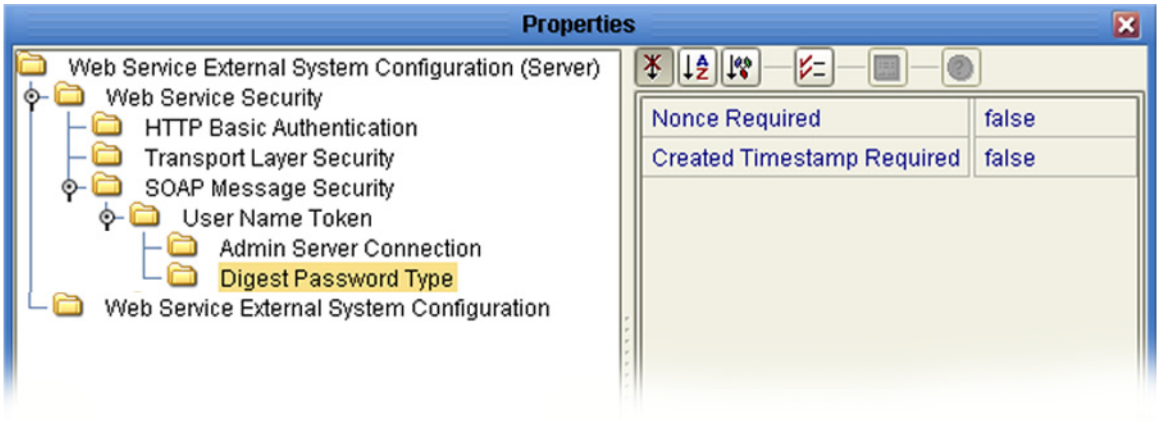
Property	Description
IP Address	Specifies the IP address or hostname for your domain. The default is **127.0.0.1**.
Port	Specifies the admin port for the integration server. The default is **18000**.
Admin User Name	Specifies the user name of the desired administrator. The default is **Administrator**.
Admin User Password	Specifies the password of the desired administrator. The default is **STC**.

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 497 Digest Password Type Configuration Dialog



Nonce Required	false
Created Timestamp Required	false

Table 167 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 **false**.
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 **false**.

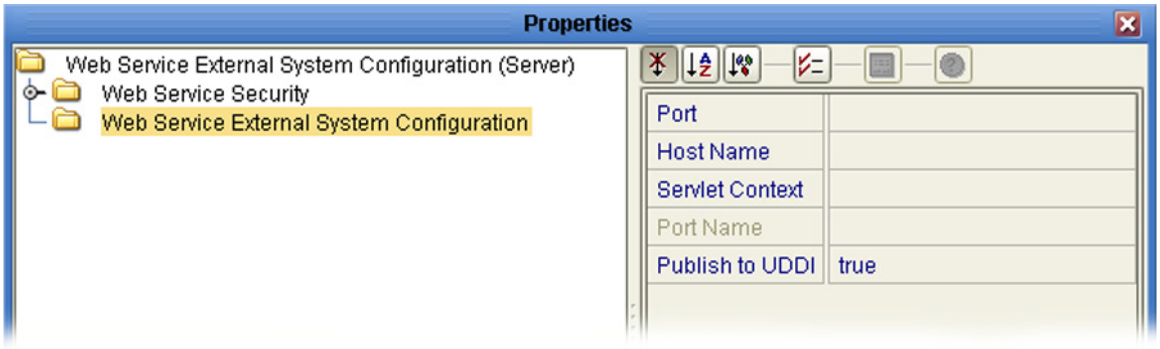
Web Service External System Configuration

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 498 Web Service External System Configuration Dialog


Properties
Web Service External System Configuration (Server)
Web Service Security
Web Service External System Configuration

Port	
Host Name	
Servlet Context	
Port Name	
Publish to UDDI	true

Table 168 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 **Note** following table).
Host Name	Name or IP address of the computer hosting the Web Services Application. There is no default. **Note:** 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. **Note:** 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 499), which displays the dialog shown in Figure 500.

Note: *You can create only one UDDI External System per run-time Environment.*

Figure 499 Environment Context Menu - New UDDI External System

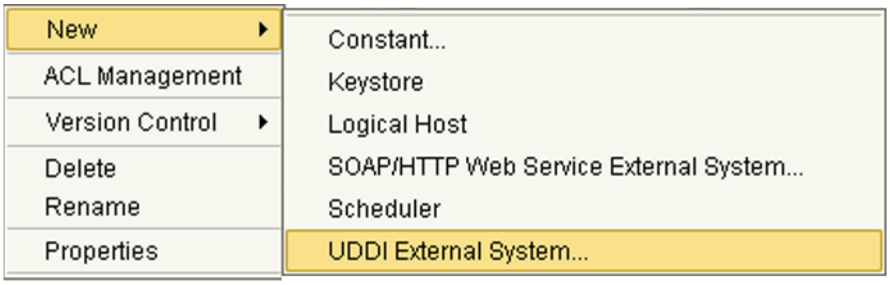
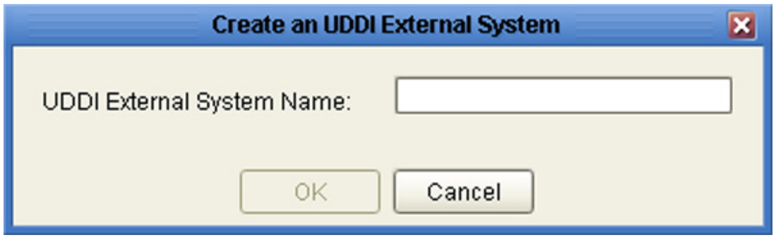


Figure 500 Create UDDI External System Dialog



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 522).

13.12.1Configuring the UDDI External System

Selecting **Properties** from the **UDDI External System** context menu (see Figure 501) displays the dialog shown in Figure 502. You should assign the appropriate values for the listed properties.

Figure 501 UDDI External System Context Menu - Properties

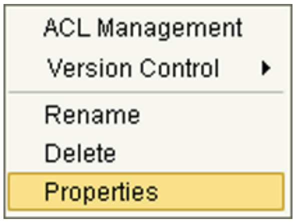
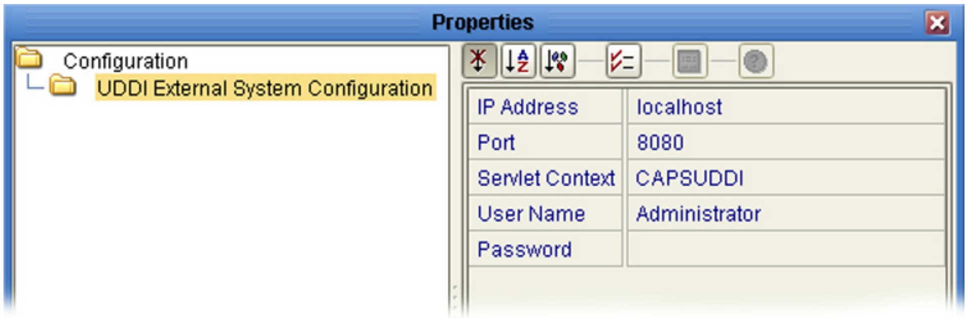


Figure 502 UDDI System Configuration Properties Dialog



IP Address	localhost
Port	8080
Servlet Context	CAPSUDDI
User Name	Administrator
Password	

Table 169 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.*

Appendix A

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 552
- **CodeGen Validation Errors** on page 553
- **Unexpected Behavior** on page 554
- **Java Method Usage** on page 555

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 55). 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 481 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 199 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 334 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
```

and set it to **true** (it is near the end of the file, and 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 199.

A.4 Java Method Usage

A.4.1 System.getProperties() Causes Exception

Projects developed prior to release 5.1.1 were allowed to use the System.getProperties() method. In release 5.1.1, this method was replaced with System.getProperty(*String propertyName*). If you are want to run a Project developed in eGate Integrator 5.0.x that incorporates the System.getProperties() method, you have two options:

1. Change all instances of System.getProperties() to System.getProperty(*String propertyName*).
2. Change the security policy of the integration server to allow the use of System.getProperties(). See the *Sun SeeBeyond eGate™ Integrator System Administration Guide* for information regarding security policy modifications.

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**, **Topic**s, **Queue**s, and **eWay**s. 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 Definition**s 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 **Project**s. 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 **Collaboration**s 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 **Topic**s, **Queue**s, **eWay**s, 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. See **Project Nesting** on page 116 regarding limitations.

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 Definition**s coded in XSLT.

Index

A

Access Collaboration method **273**
ACL properties **110**, **113**, **212**, **254**, **388**, **522**
Add and Assign Collaboration method **275**
Add Collaboration method **282**, **289**
Add Element as Child command **170**
Add Field command **170**
Add Instance command **175**
adding instances
  classes **295**
  methods **299**
addition Collaboration method **409**
Advanced Mode command **259**
ancestor node **157**
AND and Assign Collaboration method **276**
AND Collaboration method **278**, **409**
antecoding property **225**
application file
  command-line generation **149**
Assign Collaboration method **274**
Auto Layout command **267**
automap feature **144**

B

boolean Collaboration method **402**
BPEL
  definition **556**
branch, repository **79**
Break command **263**
business rules
  tree **263**
Business Rules on Left command **259**
Business Rules on Top command **259**
byte-array encoding **371–385**

C

Call Java Method command **267**
Cast Collaboration method **285**
Catch command **263**
ceiling Collaboration method **407**
CharAt Collaboration method **289**
check in **83**
  without revisions **92**
check out **85**, **90**
child node **157**
Choice node **158**
Class command **261**
classes
  adding instances **295**
Collaboration
  definition **556**
  derived **557**
Collaboration Definition
  definition **556**
Collaboration definition
  Java **325**
  XSLT **387**
Collaboration method box
  Java **271**
  XSLT **401**
Collaboration method palette
  Java **270**
  XSLT **401**
Collapse All Methods command **267**
Collapse All Rules command **261**
command-line tools
  commandline codegen **149**
  export **69**
  import **77**
  UDDIPublish **425**
Comment command **262**
Commit Changes command **264**
Commit Code Changes command **398**
Complement operator **287**
component version
  checked in **81**
  checked out **81**
  history **87**
  latest version **93**
  retrieved **81**
components
  copying and pasting **100**, **114**
  cutting and pasting **101**
Concat Collaboration method **290**
concat Collaboration method **412**
connection **556**
Connectivity Map
  definition **556**
  Editor **130**
constants **556**
  Environmental **523**
Constructor command **262**
contains Collaboration method **402**
Continue command **263**
conventions, document **36**
convert Collaboration method **382**, **383**

copying and pasting components **100**, **114**
copying and pasting Projects **100**
count Collaboration method **404**
Create Literal command **298**
current Collaboration method **404**
customizer **54**
cutting and pasting components **101**
cutting and pasting Projects **101**

D

data
  antecoding **225**
  cleansing **556**
  decoding **225**, **229**
  encoding **185**, **192**, **225**, **229**
  mapping **556**
  normalization **557**
  postcoding **225**
  transformation **557**
Data Encoding **160**
data encoding **371–385**
Data Type Definition (DTD) **183**
database, relational **559**
debugger
  Collaboration (Java) **313**
decoding property **225**, **229**
Decrement method **283**
Delete Element or Field command **170**
Delete Instance command **175**
deployment
  automapping **144**
Deployment Editor **138**
Deployment Profile **138**
  creating **140**
  definition **557**
  map variables **145**
  snapshot **147**
derived Collaboration **557**
descendant node **157**
diff file
  create **97**
  generate **97**
  merge **98**
digest password **421**
Divide and Assign Collaboration method **276**
Divide Collaboration method **283**
division Collaboration method **409**
document Collaboration method **405**
document conventions **36**
Domain
  definition **557**
Do-While command **262**
DTD
  definition **557**
  Wizard **183**

E

EAR file
  generation **149**
Editor
  Connectivity Map **130**
  Deployment **138**
  Java Collaboration **256**
  XSLT Collaboration **396**
editor
  OTD **168**
element-available Collaboration method **403**
encoding
  byte-array **371–385**
Encoding Converter method **372**
encoding property **225**, **229**
EndsWith Collaboration method **290**
Enterprise Designer
  definition **557**
  font size **60**
  menu bar **48**
  starting **47**
Enterprise Explorer
  Environments **518**
  Projects **107**
Enterprise Manager **44**
Enterprise Service Bus (ESB)
  definition **557**
Environment **42**
  constants **523**
  definition **557**
Environment Explorer **518**
Equal Collaboration method **280**
equal Collaboration method **410**
Equals Collaboration method **285**, **290**
error messages
  OutOfMemory **552**
eWay
  definition **557**
Expand All Methods command **267**
Expand All Rules command **261**
Export Java Rule command **258**
external application **121**
  definition **558**
external system **42**
  definition **558**

F

False Collaboration method **278**
false Collaboration method **403**

Field command **262**
Find command **261**, **265**
Find Next command **265**
Find Previous command **265**
floor Collaboration method **407**
font size (Enterprise Designer)
    changing **60**
For command **262**
format_number Collaboration method **412**
function_available Collaboration method **403**

G

generate-id Collaboration method **405**
Go command **176**
Greater or equal Collaboration method **280**
Greater than Collaboration method **280**
greater_or_equal Collaboration method **410**
greater_than Collaboration method **410**
Group node **158**

H

HEAD branch **79**
heap size **55**
HTTP
    definition **558**
HTTP basic authentication
    described **420**

I

Icons
    JCE Business Rules Designer toolbar **267**
    JCE Business Rules toolbar **261**, **264**
    JCE Class Browser toolbar **293**
    JCE toolbar **258**
    OTD Editor toolbar **169**
    Web Service Definition Editor toolbar **478**
    XCE toolbar **398**, **399**
    XML Schema Editor toolbar **430**
id Collaboration method **405**
If-Then command **262**
Impact Analyzer
    definition **558**
    usage **102**
Import a Local File command **258**, **398**, **399**
Import JAR File command **260**
Import OTD command **169**
importing
    WSDL files **499**
    XSD files **470**
Increment method **283**
IndexOf Collaboration method **290**
Instanceof Collaboration method **285**
Integration Server
    definition **558**
Intersection operator **279**, **287**

J

Java 2 Enterprise Edition (J2EE) **44**
Java Collaboration
    Debugger **313**
    definitions **325**
    Editor (JCE) **256**
    method box **271**
    method palette **270**
    Wizard **326**
Java Collaboration methods
    Access **273**
    Add **282**, **289**
    Add and Assign **275**
    AND **278**
    AND and Assign **276**
    Assign **274**
    Cast **285**
    CharAt **289**
    Concat **290**
    convert **382**, **383**
    Decrement **283**
    Divide **283**
    Divide and Assign **276**
    EndsWith **290**
    Equal **280**
    Equals **285**, **290**
    False **278**
    Greater or equal **280**
    Greater than **280**
    Increment **283**
    IndexOf **290**
    Instanceof **285**
    Length **274**, **290**
    Less or equal **280**
    Less than **281**
    Literal Character **290**
    Literal Number **283**
    Literal String **290**
    marshal **161**
    marshalToBytes **163**, **384**
    marshalToString **162**
    Multiply **282**
    Multiply and Assign **275**
    Negate **283**
    NOT **278**
    Not equal **281**
    Null **286**

OR **278**
OR and Assign **277**
Positive **283**
Predecrement **284**
Preincrement **283**
Remainder **283**
Remainder and Assign **276**
Replace **291**
reset **183**, **190**, **200**
Shift Left and Assign **276**
Shift Right and Assign **276**
Substring **291**
Subtract **282**
Subtract and Assign **275**
ToLowerCase **291**
ToString **286**
ToUpperCase **291**
Trim **291**
True **278**
Unassigned Shift Right and Assign **276**
unmarshal **164**
unmarshalFromBytes **166**, **375**
unmarshalFromString **165**
writeBytes **385**
XOR and Assign **277**

Java Collaboration operators
Complement **287**
Intersection **279**, **287**
Shift Left **288**
Shift Right **288**
Symmetric Difference **279**, **287**
Union **279**, **287**
Unsigned Shift Right **288**

JCE commands
Advanced Mode **259**
Auto Layout **267**
Break **263**
Business Rules on Left **259**
Business Rules on Top **259**
Call Java Method **267**
Catch **263**
Class **261**
Collapse All Methods **267**
Collapse All Rules **261**
Comment **262**
Commit Changes **264**
Constructor **262**
Continue **263**
Create Literal **298**
Do-While **262**
Expand All Methods **267**
Expand All Rules **261**
Export Java Rule **258**
Field **262**
Find **261**, **265**
Find Next **265**
Find Previous **265**
For **262**
If-Then **262**
Import a Local File **258**, **398**, **399**
Import JAR File **260**
Local Variable **262**
Method **261**
New Array **267**
Redo **265**
Refresh Collaboration **260**
Replace **265**
Return **262**, **298**
Roll Back Changes **264**
Rule **262**
Source Code Mode **259**
Standard Mode **259**
Switch **262**
Test Mode **259**
Throw **262**
Try **262**
Undo **265**
Validate **259**
While **262**

JMS client
reconfiguring **133**

JMS IQ Manager
definition **558**

K

key Collaboration method **405**

L

language Collaboration method **403**
last Collaboration method **405**
Leaf nodes **158**
Length Collaboration method **274**, **290**
Less or equal Collaboration method **280**
Less than Collaboration method **281**
lesser_or_equal Collaboration method **410**
lesser_than Collaboration method **410**
link **558**
Literal Character Collaboration method **290**
Literal Number Collaboration method **283**
Literal String Collaboration method **290**
Local Variable command **262**
local-name Collaboration method **406**
Logical Host **42**
definition **558**

M

map variables **145**
marshal Collaboration method **161**
Marshal command **175**
Marshal To Bytes command **175**
Marshal To String command **175**
marshalToBytes Collaboration method **163**, **384**
marshalToString Collaboration method **162**
menu bar **48**
message destination
  definition **558**
  linked **558**
metadata **558**
method box
  Java Collaboration **271**
  XSLT Collaboration **401**
Method command **261**
method palette
  Java Collaboration **270**
  XSLT Collaboration **401**
methods
  adding instances **270**, **299**, **400**
multiplication Collaboration method **410**
Multiply and Assign Collaboration method **275**
Multiply Collaboration method **282**

N

name Collaboration method **406**
namespace-uri Collaboration method **406**
Negate operator **283**
negative Collaboration method **411**
nesting of Projects **116**
New Array command **267**
node
  ancestor **157**
  child **157**
  choice **158**
  descendant **157**
  group **158**
  leaf **158**
  non-leaf **158**
  parent **157**
  repeating **266**
  root **157**
  sibling **157**
nonce **541**
non-leaf node **158**
normalize_space Collaboration method **412**
NOT Collaboration method **278**, **403**
Not equal Collaboration method **281**
not_equal Collaboration method **411**
Null Collaboration method **286**
number Collaboration method **407**
number-literal Collaboration method **407**

O

Object Type Definition **252**, **387**, **559**
  wizard **184**, **191**
Object Type Definition (OTD) **155**
Open File command **175**
OR and Assign Collaboration method **277**
OR Collaboration method **278**, **411**
OTD **559**
  editor **168**
  properties **224**
  tester **171**
OTD Editor commands
  Add Element as Child **170**
  Add Field **170**
  Delete Element or Field **170**
  Go **176**
  Import OTD **169**
  Open File **175**
  Redo **170**
  Save as New Name **169**
  Save File **175**
  Show as hex **175**
  Tester **169**
  Toggle Reference Tab Panel **170**
  Undo **170**
  Word wrap **176**
OTD Tester commands
  Add Instance **175**
  Delete Instance **175**
  Marshal **175**
  Marshal To Bytes **175**
  Marshal To String **175**
  Reset **175**
  Unmarshal **176**
  Unmarshal From Bytes **176**
  Unmarshal From String **176**
OTD Wizard
  DTD **183**
  User-Defined **200**, **202**
  XSD **190**
OutOfMemory error **552**

P

palette
  Java Collaboration methods **270**
  XSLT Collaboration methods **401**
parent node **157**
password equivalent **421**, **540**
passwords

WS-Security **421**
position Collaboration method **406**
Positive method **283**
postcoding property **225**
Predecrement method **284**
Preincrement method **283**
Profile, Deployment **138**
Project
copying and pasting **100**
cutting and pasting **101**
definition **559**
nesting **116**
Project Explorer **107**
properties
OTD **224**

Q

queue
definition **559**

R

rdbms **559**
reconfiguring
JMS client **133**
Redo command **170**, **265**
Refresh Collaboration command **260**
relational database **559**
Relaunch **189**, **199**
Remainder and Assign Collaboration method **276**
Remainder Collaboration method **283**
remainder Collaboration method **411**
repeating nodes
group **158**
mapping between **266**
Replace Collaboration method **291**
Replace command **265**
Repository **41**, **62**
definition **559**
repository
branches **79**
reset Collaboration method **183**, **190**, **200**
Reset command **175**
Return command **262**, **298**
Roll Back Changes command **264**
Roll Back Code Changes command **398**
root node **157**
round Collaboration method **408**
Rule command **262**

S

Save as New Name command **169**
Save File command **175**
Save XSL to a Local File command **398**
Scheduler **123**
Schema Runtime Environment (SRE) **44**
definition **559**
Screen Captures **36**
security **529**
service
definition **559**
servlet context property **549**
Shift Left and Assign Collaboration method **276**
Shift Left operator **288**
Shift Right and Assign Collaboration method **276**
Shift Right operator **288**
Show as hex command **175**
Show Mapping Only command **398**
Show Maps and Code command **398**
Show XSLT Code Only command **398**
Sibling node **157**
snapshot **147**
SOAP
definition **559**
encoding **499**
Source Code Mode command **259**
Standard Mode command **259**
starting
Enterprise Designer **47**
starts_with Collaboration method **403**
string Collaboration method **413**
string_length Collaboration method **413**
string_literal Collaboration method **413**
Subproject
definition **560**
nesting **116**
Substring Collaboration method **291**
substring Collaboration method **413**
substring-after Collaboration method **413**
substring-before Collaboration method **413**
Subtract and Assign Collaboration method **275**
Subtract Collaboration method **282**
subtraction Collaboration method **411**
sum Collaboration method **408**
supporting documents **37**
Switch command **262**
Symmetric Difference operator **279**, **287**
system-property Collaboration method **414**

T

table **560**
tagging components **95**

Test Mode command **259**
tester
- Collaboration (Java) **313**
- OTD **171**
- XSLT **396**

Tester command **169**
Throw command **262**
Toggle Reference Tab Panel command **170**
token
- defined **420**

ToLowerCase Collaboration method **291**
topic
- definition **560**

ToString Collaboration method **286**
ToUpperCase Collaboration method **291**
transformation **560**
translate Collaboration method **414**
Transport Layer Security
- described **420**

Trim Collaboration method **291**
True Collaboration method **278**
true Collaboration method **403**
Try command **262**

U

UDDI
- definition **560**
- registry **426**

Unassigned Shift Right and Assign Collaboration method **276**
undo check-out **92**
Undo command **170**, **265**
Union operator **279**, **287**
unmarshal Collaboration method **164**
Unmarshal command **176**
Unmarshal From Bytes command **176**
Unmarshal From String command **176**
unmarshalFromBytes Collaboration method **166**, **375**
unmarshalFromString Collaboration method **165**
unparsed-entity-uri Collaboration method **414**
Unsigned Shift Right operator **288**
User-Defined OTD Wizard **200**, **202**

V

Validate command **259**
variables
- mapping **145**

version control **81**
- branches **79**
- check in **81**, **83**
- check out **81**, **85**, **90**
- command-line utilities **99**
- create diff **97**
- definition **560**
- generate diff **97**
- history **87**
- latest version **93**
- merge diff **98**
- retrieve **81**
- tagging **95**
- undo check out **92**

W

Web Service Definition Editor **476**
web services
- invoke from Collaboration **513**

While command **262**
Wizard
- Java Collaboration **326**
- OTD (DTD) **183**
- OTD (User-Defined) **200**, **202**
- OTD (XSD) **190**
- WSDL import **499**
- XSD import **470**
- XSLT Collaboration **390**

Word wrap command **176**
writeBytes Collaboration method **385**
WSDL
- definition **560**
- editor **476**
- Import Wizard **499**

WS-Security
- described **420**

X

XCE commands
- Commit Code Changes **398**
- Roll Back Code Changes **398**
- Save XSL to a Local File **398**
- Show Mapping Only **398**
- Show Maps and Code **398**
- Show XSLT Code Only **398**

XML
- definition **560**

XML Schema Editor **428**
XOR and Assign Collaboration method **277**
XSD
- definition **560**
- editor **428**
- Import Wizard **470**
- OTD Wizard **190**

XSLT
- definition **560**

XSLT Code Tester **396**
XSLT Collaboration
  definitions **387**
  Editor **396**
  method box **401**
  method palette **401**
  Wizard **390**
XSLT Collaboration methods
  addition **409**
  AND **409**
  boolean **402**
  ceiling **407**
  concat **412**
  contains **402**
  count **404**
  current **404**
  division **409**
  document **405**
  element-available **403**
  equal **410**
  false **403**
  floor **407**
  format_number **412**
  function_available **403**
  generate-id **405**
  greater_or_equal **410**
  greater_than **410**
  id **405**
  key **405**
  language **403**
  last **405**
  lesser_or_equal **410**
  lesser_than **410**
  local-name **406**
  multiplication **410**
  name **406**
  namespace-uri **406**
  negative **411**
  normalize_space **412**
  NOT **403**
  not_equal **411**
  number **407**
  number-literal **407**
  OR **411**
  position **406**
  remainder **411**
  round **408**
  starts_with **403**
  string **413**
  string_length **413**
  string_literal **413**
  substring **413**
  substring-after **413**
  substring-before **413**
  subtraction **411**
  sum **408**
  system-property **414**
  translate **414**
  true **403**
  unparsed-entity-uri **414**