SeeBeyond ICAN Suite

eGate Integrator User's Guide

Release 5.0.4

Version 20040709094807.

Contents

Figures 11

Tables 20

Chapter 1

Introduction 23

Purpose and Scope 23

Intended Audience 23

Organization of Information 24

Writing Conventions 25

Additional Conventions 25

Supporting Documents 25

The SeeBeyond Web Site 26

Chapter 2

System Overview 27

Introduction 27

Integration Model 28

System Architecture 30

Repository 31

Run-Time Environments 31

User Interfaces 32

Enterprise Manager 32

Enterprise Designer 32

Chapter 3

Enterprise Manager 35

Overview 35

Installing and Updating eGate 35

Monitoring and Managing eGate 35
Starting Enterprise Manager 36
The Enterprise Manager Interface 37
Home 38
Documentation 39
The ICAN Monitor 40
The SRE Monitor 41
Starting the SRE Monitor 42
Controlling Components 44
Viewing Message Destinations 47
Viewing Log Files 50
Viewing Alerts 52

Chapter 4
Enterprise Designer 54
Overview 54
User Interface 55
Editors 55
Analysis and Archiving tools 55
Using a Proxy Server 56
Starting Enterprise Designer 58
Interface Features 60
Menus 60
File Menu 60
Tools Menu 60
View Menu 60
Window Menu 61
Help Menu 61
Options Setup 62
Toolbar 63
Browser Buttons 63
Enterprise Explorer 64
Project Explorer 64
Environment Explorer 65
Enterprise Designer Editors 66
Connectivity Map Editor 66
OTD Editor 67
Collaboration Editor (Java) 68
Collaboration Editor (XSLT) 69
Environment Editor 70
Deployment Editor 71
Additional Tools and Features 72
Project/Environment Import 72
Importing a Project Using Enterprise Designer 72
Importing a Project Using the Command Line 75

Project/Environment Export 76
Exporting a Project Using Enterprise Designer 76
Exporting a Project Using the Command Line 80
Impact Analyzer 81
Version Control 83
Viewing a Component's Version History 84
Checking a Component In 85
Checking a Component In Without Revisions 86
Checking a Component Out 87
Retrieving a Component to Your Workspace 88
Checking In a Previous Version as the Latest Version 90
Command-line Utilities 91

Chapter 5
eGate Projects 92
Overview 92
Project Components 92
The Project Explorer 93
Project Explorer Icons 94
Context Menus 95
Repository Menu 95
Project Menu 96
Connectivity Map Menu 98
Object Type Definition Menu 99
Collaboration Definition Menu 100
Deployment Profile 102
Using the Connectivity Map Editor 104
Services 106
Collaborations 106
Message Destinations 107
External Applications 107
Schedulers 108
Component Connections 109
Configuring a Connection 110
Defining Constants and Variables 112

Chapter 6
Object Type Definitions 114
Overview 114
OTD Types 115
Externally-Defined OTDs 115
User-Defined OTDs 115
JMS OTDs 115
OTD Libraries 115

OTD Wizards 116
OTD Editor 118
OTD Tester 119
Using the OTD Tester 123
Creating Externally-Defined OTDs 126
Using the DTD Wizard 126
Using the WSDL Wizard 132
WSDL OTD Structure 135
WSDL Operation Elements 135
Using the XSD Wizard 136
Creating User-Defined OTDs 142
Using the User-Defined OTD Wizard 142
Creating and Managing Nodes 143
Creating and Modifying Elements 144
Creating and Modifying Fields 145
Editing OTD Properties 146
Root Node Properties 147
Element Node Properties 148
Field Node Properties 149
Specifying the Node Type 151
Specifying Delimiters 152
Delimiter List 152
Delimiter Properties 154
Delimiter Levels 155
Multiple Delimiters 157
Delimiter Type 158
Precedence 159
Optional Mode 160
Terminator Mode 161
Escape Sequences 161
Creating a Delimiter List 162
OTD Libraries 167
CBO OTD Library 167
Add-on OTD Libraries 167

Chapter 7
Collaboration Definitions (Java) 168
Overview 168
Using the Collaboration Definition Wizard (Java) 169
Creating a Java-based Collaboration Definition 170
New Web Service 171
Existing Web Service 175
Editing Collaboration Definition Properties 177
Using the Collaboration Editor (Java) 179
Java Toolbar Icons 180
Business Rules Editor 181

Commands 181
Business Rules Toolbar Icons 182
Business Rules Tree 183
Java Source Editor 184
Java Source Editor Toolbar Icons 184
Collaboration Tester 186
Business Rules Designer 187
Method-Access Nodes 188
Business Rules Designer Toolbar Icons 189
Collaboration Method Palette (Java) 192
Collaboration Method Boxes (Java) 192
Collaboration Methods (Java) 193
Boolean Methods 193
Comparison Methods 194
Flow Control Methods 196
Math Methods 197
Object Methods 199
String Methods 202
Array Operation Methods 205
Operators 207
Version Control 209
Creating a Modified Collaboration Definition (Java) 209
Merging Two Versions of a Collaboration Definition (Java) 210
Setting Up Collaboration Definition Variables 211
Creating a Variable 211
Invoking the Variable Constructor 214
Displaying Method Classes 216
Using *Try-Catch* *217*
Adding and Using Third-Party Java Classes 220
Adding Class Instances to a Collaboration 224
Validating Java-based Collaboration Definitions 226
Debugging Java-based Collaboration Definitions 227
Enabling the Debugger 227
Invoking the Java Debugger 229
Setting Breakpoints 231
Inspecting and Editing the Source Code 233
Stepping Into, Over, or Out 234
Inspecting Java Threads and Methods 235
Inspecting a Local Variable or Method 236
Saving and Resuming Debug Sessions 239
Creating Alerts 240
Creating Log Entries 243
Using Byte-Array Converters 245
Defining Encoding Converter Methods 246

Chapter 8

Collaboration Definitions (XSLT) 259

Overview 259

Using the Collaboration Definition Wizard (XSLT) 260
Creating a Collaboration Definition (XSLT) 261
New Web Service 262
Existing Web Service 265

Using the Collaboration Editor (XSLT) 266
XSLT Toolbar Icons 267
Collaboration Method Palette (XSLT) 268
Collaboration Method Boxes (XSLT) 268

Collaboration Methods (XSLT) 269
Operator Methods 269
String Methods 272
Number Methods 275
Boolean Methods 277
Nodes Methods 279
UAN Extension Methods 282

Version Control 288
Creating a Modified Collaboration Definition (XSLT) 288
Merging Two Versions of a Collaboration Definition (XSLT) 288

Chapter 9

Environments 290

Overview 290
Environment Components 290

Environment Explorer 291
Environment Explorer Icons 292
Context Menus 293
Repository Menu 293
Environment Menu 294
Logical Host Menu 295
Integration Server 297
SeeBeyond JMS IQ Manager 298

Environment Editor 299
Defining Environmental Constants 299

Logical Hosts 301
Overview 301
Bootstrapping 301
Configuring a Logical Host 302
Configuring the Base Port Number 303

Integration Servers 305
Configuring an Integration Server 305

Web Container 307
Web Server Configurations 307
Default Web Server 308
Performance Monitoring (Profiling) 310
Security Realm 311
eInsight Engine 312
Application Manager 313
Integration Server 314
JDBC DataSource Connection Pools 316
Oracle JDBC Connection Pool 316
Deploying User-Defined Stateless Session Beans 318
Message Servers 321
SeeBeyond JMS IQ Manager Configuration 321
General Configuration 321
Performance 323
Messaging Behavior 325
Sun Java System 326
Active Directory Service 327
Diagnostics 328
Stable Storage 330
JNDI 331

Chapter 10
Project Deployment 332
Deployment Profiles 332
The Deployment Editor 333
Creating a Deployment Profile 334
Activating and Deactivating Deployment Profiles 337
Using Enterprise Designer 337
Using a Command-line Script 339
Mapping Variables 340
Deploying Projects to Third-Party Servers 341
BEA WebLogic 341
IBM WebSphere 344

Chapter 11
Web Services 347
Overview 347
SeeBeyond Web Services 348
UDDI Registry 349
Using UDDI Browsers 351
Web Services Application 352

Web Services External System 353

Load Balancing 354
- Configuring the Apache Server 354
- Creating the Mapping File 355
- Adding the Rewrite Rule 355
- Debugging 356

Building a Web Client 357
- Object Type Definition 357
- eInsight Business Process 359
- eGate Project 361

Building a Web Server 364
- Object Type Definition 364
- eInsight Business Process 367
- eGate Project 367

Glossary 369

Index 377

Figures

Figure 1	eGate Integrator	27
Figure 2	eGate Integrator Implementation Model	29
Figure 3	Distributed eGate Integrator System	30
Figure 4	SeeBeyond Enterprise Manager	32
Figure 5	Enterprise Designer	33
Figure 6	Connectivity Map Editor	34
Figure 7	Enterprise Manager Login	36
Figure 8	Enterprise Manager GUI	37
Figure 9	ICAN Monitor Launch Icon	38
Figure 10	ICAN Monitor/SRE Monitor Launch Icons	38
Figure 11	Documentation Page	39
Figure 12	ICAN Monitor Interface - Initial	40
Figure 13	ICAN Monitor Interface - Environment	40
Figure 14	SRE Monitor	41
Figure 15	Enterprise Manager Home Page	42
Figure 16	SRE Monitor Initial Page	42
Figure 17	Add Registry/Repository Dialog Box	43
Figure 18	Viewing SRE Components	44
Figure 19	FromExternal e*Way Status (Not Running)	45
Figure 20	FromExternal e*Way Status (Running)	45
Figure 21	Refreshing the Explorer Tree	46
Figure 22	Viewing JMS Topics and Queues	47
Figure 23	Message Payload Dialog	48
Figure 24	Viewing Logs	50
Figure 25	Viewing Alerts	52
Figure 26	SeeBeyond Enterprise Designer	54
Figure 27	Update Center Wizard	56
Figure 28	Proxy Configuration Dialog Box	57
Figure 29	Login Dialog Box	58
Figure 30	Options Setup - Heap Size Dialog	62
Figure 31	Options Setup - Language Dialog	62
Figure 32	Enterprise Explorer: Project Explorer View	64

Figure 33	Enterprise Explorer: Environment Explorer View	65
Figure 34	Connectivity Map Editor	66
Figure 35	OTD Editor	67
Figure 36	Collaboration Editor (Java)	68
Figure 37	Collaboration Editor (XSLT)	69
Figure 38	Environment Editor	70
Figure 39	Deployment Editor	71
Figure 40	Import Message Box	72
Figure 41	Import Manager Dialog Box (1)	73
Figure 42	Open File Dialog Box	73
Figure 43	Import Manager Dialog Box	74
Figure 44	Import Status Message Box	74
Figure 45	Export Manager Dialog Box (1a)	76
Figure 46	Export Manager Dialog Box (1b)	77
Figure 47	Export Manager Dialog Box (2)	78
Figure 48	Save As Dialog Box	78
Figure 49	Enter File Name Dialog Box (2)	79
Figure 50	Export Status Message Box	79
Figure 51	Impact Analyzer Dialog Box	81
Figure 52	*Checked In* Icon (OTD Example)	83
Figure 53	*Checked Out* Icon	83
Figure 54	*Retrieved* Icon	83
Figure 55	Version Control - History Dialog Box	84
Figure 56	Version Control - Check In Dialog Box	85
Figure 57	Version Control - Undo Check Out Dialog Box	86
Figure 58	Version Control - Check Out Dialog Box	87
Figure 59	Version Control - History Dialog Box	88
Figure 60	Access File Dialog Box	89
Figure 61	Confirm Version Replace Dialog Box	89
Figure 62	Make Latest Dialog Box	90
Figure 63	Confirm Latest Version Override Dialog Box	91
Figure 64	Project Explorer	93
Figure 65	Repository Menu	95
Figure 66	Project Menu	96
Figure 67	Connectivity Map Menu	98
Figure 68	OTD Menu	99
Figure 69	Collaboration Definition Menu	100
Figure 70	Deployment Profile Menu	102

Figure 71	Connectivity Map Editor	104
Figure 72	Linking JMS Topics	105
Figure 73	Service Component	106
Figure 74	External Application Drop-Down Menu	107
Figure 75	Scheduler Properties Dialog Box	108
Figure 76	Connection Icons in a Connectivity Map	109
Figure 77	Default Configuration Dialog Box	110
Figure 78	Project Variable Creation	112
Figure 79	Project Constant Creation	113
Figure 80	Variables and Constants Object Group	113
Figure 81	OTD Wizard Selection Dialog	116
Figure 82	OTD Editor	118
Figure 83	OTD Tester	119
Figure 84	OTD Tester - Data Encoding Enabled	121
Figure 85	Specify Encoding Dialog Box	121
Figure 86	Select Data File	123
Figure 87	Object Elements and Values	123
Figure 88	Data Display - Status Panel	124
Figure 89	Data Display - Verbose Panel	124
Figure 90	OTD Tester Node Table	125
Figure 91	Serialized Data in Output Panel	125
Figure 92	OTD Wizard Selection: DTD Wizard	126
Figure 93	Select DTD File(s) Dialog Box	127
Figure 94	Cannot Create OTD Warning Box	127
Figure 95	Select Encoding Dialog Box	128
Figure 96	Select Document Elements Dialog Box	129
Figure 97	Select OTD Options Dialog Box	130
Figure 98	OTD Wizard Selection: WSDL Wizard	132
Figure 99	WSDL Wizard: Select WSDL Location	133
Figure 100	WSDL Wizard: Select WSDL File	133
Figure 101	WSDL Wizard: Select OTD Options	134
Figure 102	OTD Wizard Selection: XSD Wizard	136
Figure 103	XSD Wizard: Select XSD File(s)	137
Figure 104	Cannot Create OTD Warning Box	137
Figure 105	Select Encoding Dialog Box	138
Figure 106	Select Document Elements Dialog Box	139
Figure 107	Select OTD Options Dialog Box	140
Figure 108	OTD Wizard Selection: User-Defined OTD	142

Figure 109 Enter OTD Name 143
Figure 110 Add Element Options 144
Figure 111 Input Dialog Box 144
Figure 112 Add Field Options 145
Figure 113 User-Defined OTD Root Node Properties 147
Figure 114 User-Defined OTD Element Node Properties 148
Figure 115 User-Defined OTD Field Node Properties 149
Figure 116 Node Type Property Options (Root Node Example) 151
Figure 117 Delimiter List Editor 152
Figure 118 Popup Menu 153
Figure 119 OTD Hierarchical and Delimiter Levels 155
Figure 120 Multiple Delimiter Example 157
Figure 121 Escape Type Delim 158
Figure 122 Optional Mode Property (Example 1) 160
Figure 123 Optional Mode Property (Example 2) 160
Figure 124 Terminal Mode Property Example 161
Figure 125 Demo OTD Tree 162
Figure 126 Initial Root Node Properties 162
Figure 127 Activated *delim* Value Field 162
Figure 128 Delimiter List Editor - Delimiters Not Set 163
Figure 129 Delimiter List Editor - Insert Level 163
Figure 130 Delimiter List Editor - Add Delimiter 164
Figure 131 Delimiter List Editor - Add Levels and Delimiters 164
Figure 132 Root Node - Delimiter Specified 165
Figure 133 Element_1 Node Properties 165
Figure 134 Element_2 Node Properties 166
Figure 135 Field_1 Node Properties 166
Figure 136 Initial Wizard Dialog 170
Figure 137 New Web Service: Operation Name 171
Figure 138 New Web Service: Input Message 172
Figure 139 New Web Service: Output Message 173
Figure 140 New Web Service: Auxiliary OTD 174
Figure 141 Existing Web Service: Select Operation 175
Figure 142 Existing Web Service: Select OTD 176
Figure 143 Collaboration Definition (Java) Properties 177
Figure 144 Collaboration Definition (Java) Properties 178
Figure 145 Collaboration Editor (Java) 179
Figure 146 Business Rules Editor 181

Figure 147 Java Source Editor 184
Figure 148 JCE Tester 186
Figure 149 Business Rules Designer: Addition Method 187
Figure 150 Method-Access Nodes 188
Figure 151 Java Collaboration Method Palette Dialog Box 192
Figure 152 Expanded Method Box 192
Figure 153 Collapsed Method Box 192
Figure 154 Method Palette: Boolean Methods 193
Figure 155 Method Palette: Comparison Methods 194
Figure 156 Method Palette: Flow Control Methods 196
Figure 157 Method Palette: Math Methods 197
Figure 158 Method Palette: Object Methods 199
Figure 159 Select Java Class Browser 201
Figure 160 Method Palette: String Methods 202
Figure 161 Method Palette: Array Operation Methods 205
Figure 162 Method Palette: Operators 207
Figure 163 Version Control - Create Diff Dialog Box 209
Figure 164 Version Control - Merge Changes Dialog Box 210
Figure 165 Create Variable Dialog Box 211
Figure 166 Find Class Dialog Box 212
Figure 167 Collaboration Definition (Java) with Variable 213
Figure 168 Call New Constructor Dialog Box 214
Figure 169 Collaboration Definition (Java) with Constructor 215
Figure 170 Variable Context Menu 216
Figure 171 Method List Box 216
Figure 172 Try Icon 217
Figure 173 Try Option Menu (Partial) 217
Figure 174 Create a New Exception Variable Dialog Box 217
Figure 175 Find Class Dialog Box 218
Figure 176 Catch SQLException Rule 218
Figure 177 Exception Message 219
Figure 178 Project Context Menu: New File 220
Figure 179 File Import Dialog Box 220
Figure 180 Import JAR File Icon 221
Figure 181 Add/Remove Jar File Dialog Box (1) 221
Figure 182 Add/Remove Jar File Dialog Box (2) 222
Figure 183 Call Java Method Dialog Box 222
Figure 184 Using the Third-Party Java Method 223

Figure 185 Call New Constructor Dialog Box 224
Figure 186 Constructor Example 1 225
Figure 187 Constructor Example 2 225
Figure 188 Validating a Collaboration Definition 226
Figure 189 Integration Server Properties Dialog Box 227
Figure 190 Logical Host Context Menu 228
Figure 191 Integration Server Context Menu 228
Figure 192 Java Debugger 229
Figure 193 File Menu 230
Figure 194 Attach to JVM Dialog Box 230
Figure 195 Collaboration Source Code Display 230
Figure 196 Breakpoint Example 231
Figure 197 Debug Menu 231
Figure 198 Stop in Method Dialog Box 232
Figure 199 Choose Exception Dialog Box 232
Figure 200 Break on Exception Dialog Box 232
Figure 201 Breakpoint Indicator 233
Figure 202 Stepping Into, Over, and Out Commands 234
Figure 203 Java Thread and Method Display 235
Figure 204 Local Variables Tab 236
Figure 205 Evaluate Local Variable 237
Figure 206 Evaluate Method 238
Figure 207 Save Debugger Session Dialog Box 239
Figure 208 Resume Debugger Session Dialog Box 239
Figure 209 Empty File Test 240
Figure 210 Alert Menu 241
Figure 211 Alert Severity Selection Window 241
Figure 212 Create Literal Dialog Box 241
Figure 213 Pass Alert Message to Object Argument 242
Figure 214 Logging Menu 243
Figure 215 Logging Level/Method List 243
Figure 216 Create Literal Dialog Box 244
Figure 217 Pass Log Message to Object Argument 244
Figure 218 Encoding Converter Methods Dialog Box 246
Figure 219 Using *unmarshalFromBytes* Method 249
Figure 220 Create a Variable Dialog Box 250
Figure 221 Find Class Dialog Box 251
Figure 222 Local Variable *MyConverter* Added 252

Figure 223 Encoding Converter Icon 252
Figure 224 Encoding Converter Method Dialog Box 253
Figure 225 Encoding Converter Rule Created 254
Figure 226 Link Rule Result to Variable 254
Figure 227 Select a Method to Call 255
Figure 228 Select Method List 255
Figure 229 Convert Method Box 256
Figure 230 Data Mapping 256
Figure 231 Marshal Output Data 257
Figure 232 Output Data 258
Figure 233 Collaboration Wizard (XSLT) Dialog Box 261
Figure 234 New Web Service: Operation Name 262
Figure 235 New Web Service: Input Message 263
Figure 236 New Web Service: Output Message 264
Figure 237 Existing Web Service: Select Operation 265
Figure 238 Collaboration Editor (XSLT) 266
Figure 239 XSLT Collaboration Method Palette Dialog Box 268
Figure 240 Expanded Method Box 268
Figure 241 Collapsed Method Box 268
Figure 242 Method Palette: Operator Methods 269
Figure 243 Method Palette: String Methods 272
Figure 244 Method Palette: Number Methods 275
Figure 245 Method Palette: Boolean Methods 277
Figure 246 Method Palette: Nodes Methods 279
Figure 247 Method Palette: UAN Extension Methods 282
Figure 248 Collaboration Definition (XSLT) Context Menu 288
Figure 249 Version Control - Create Diff Dialog Box 288
Figure 250 Version Control - Merge Changes Dialog Box 289
Figure 251 Enterprise Explorer: Environment Explorer View 291
Figure 252 Repository Menu 293
Figure 253 Environment Menu 294
Figure 254 Logical Host Menu 295
Figure 255 Logical Host Menu with Third-Party Servers 296
Figure 256 Integration Server Menu 297
Figure 257 JMS IQ Manager Menu 298
Figure 258 Environment Editor 299
Figure 259 Environmental Constants Panel 300
Figure 260 Logical Hosts 301

Figure 261 Startup Sequence 302
Figure 262 Logical Host Configuration Properties 303
Figure 263 Management Agent Configuration Properties 304
Figure 264 Top-level IS Configuration Properties 305
Figure 265 Web Container Configuration Properties 307
Figure 266 Default Web Server Properties 308
Figure 267 Profiling Configuration Properties 310
Figure 268 Security Configuration Properties 311
Figure 269 eInsight Engine Configuration Properties 312
Figure 270 Application Manager Configuration Properties 313
Figure 271 Integration Server Configuration Properties 314
Figure 272 Oracle JDBC Connection Pool Properties 316
Figure 273 JMS IQ Manager - General Configuration Properties 321
Figure 274 Performance Configuration Properties 323
Figure 275 Messaging Behavior Configuration Properties 325
Figure 276 Sun Java System Configuration Properties 326
Figure 277 Active Directory Service Configuration Properties 327
Figure 278 Diagnostics Configuration Properties 328
Figure 279 Stable Storage Configuration Properties 330
Figure 280 eGate Integrator Implementation Model 332
Figure 281 Deployment Editor Window 333
Figure 282 Web Client Example Project 334
Figure 283 Web Client Example Environment 335
Figure 284 Example Deployment Profile (1) 335
Figure 285 Example Deployment Profile (2) 336
Figure 286 Example Deployment Profile (3) 336
Figure 287 Activate Dialog Box 337
Figure 288 Success Information Box 337
Figure 289 Logical Host Context Menu - Apply 338
Figure 290 Deactivate Dialog Box 338
Figure 291 Deployment Profile Mappings 340
Figure 292 Project Variable Value Entry 340
Figure 293 WebLogic Deployment (1) 342
Figure 294 WebLogic Deployment (2) 342
Figure 295 WebLogic Deployment Verification 343
Figure 296 WebSphere Deployment (1) 344
Figure 297 WebSphere Deployment (2) 345
Figure 298 WebSphere Deployment Verification 346

Figure 299 SeeBeyond Web Services Page 349
Figure 300 Example Web Service WSDL File 349
Figure 301 Microsoft Visual Studio Example 350
Figure 302 Enterprise Designer - Connectivity Map Editor 352
Figure 303 Web Service External System Properties Dialog Box 353
Figure 304 Load Balancing Example 354
Figure 305 Select WSDL Wizard 357
Figure 306 Select File Location 358
Figure 307 Select WSDL File 358
Figure 308 Select External Server 359
Figure 309 Web Client Business Process 359
Figure 310 Web Client Business Process *Receive* Rule 360
Figure 311 Web Client Business Process *Write* Rule 360
Figure 312 Sample WSDL File 360
Figure 313 Map Business Process 361
Figure 314 Web Client Connectivity Map 361
Figure 315 Web Client Example Project 362
Figure 316 Web Client Deployment (1) 362
Figure 317 Web Client Deployment (2) 363
Figure 318 Select WSDL Wizard 364
Figure 319 Select File Location 365
Figure 320 Select WSDL File 365
Figure 321 Select External Client 366
Figure 322 Web Server Business Process 367
Figure 323 Connectivity Map 367
Figure 324 Web Server Example Project 367
Figure 325 Web Server Deployment (1) 368
Figure 326 Web Server Deployment (2) 368

Tables

Table 1	Writing Conventions	25
Table 2	Enterprise Manager - Pages	37
Table 3	Enterprise Manager - Control Tabs	37
Table 4	Document Categories	39
Table 5	ICAN Monitor Interface - Details Tabs	40
Table 6	SRE Monitor Explorer Icons	41
Table 7	SRE Monitor Topic / Queue Icons	47
Table 8	SRE Monitor Logging Icons	50
Table 9	SRE Monitor Alerts Icons	53
Table 10	File Menu Options	60
Table 11	Tools Menu Options	60
Table 12	View Menu Options	60
Table 13	Window Menu Options	61
Table 14	Help Menu Options	61
Table 15	Enterprise Designer Toolbar Icons	63
Table 16	Browser Buttons	63
Table 17	Impact Analyzer Command Buttons	82
Table 18	Project Icons	94
Table 19	Repository Menu Options	95
Table 20	Project Menu Options	96
Table 21	Connectivity Map Menu Options	98
Table 22	OTD Menu Options	99
Table 23	Collaboration Definition Menu Options	100
Table 24	Deployment Profile Menu Options	102
Table 25	Connectivity Map Toolbar Icons	105
Table 26	Configuration Dialog Box Toolbar Buttons	111
Table 27	OTD Wizard Navigation Buttons	117
Table 28	OTD Editor Toolbar Icons	119
Table 29	OTD Tester Buttons	120
Table 30	OTD Tester Icons	120
Table 31	Encoding Options	122
Table 32	Encoding Options	128

Table 33	DTD OTD Options	130
Table 34	Encoding Options	138
Table 35	XSD OTD Options	140
Table 36	Add Element Options	144
Table 37	Add Field Options	145
Table 38	Root Node Properties	147
Table 39	Element Node Properties	148
Table 40	Field Node Properties	149
Table 41	Encoding Options	150
Table 42	Node Type Property Options	151
Table 43	Delimiter List Editor Command Buttons	153
Table 44	Delimiter Properties and Value Options	154
Table 45	Escape Sequences	161
Table 46	Wizard Navigation Buttons	169
Table 47	Java Toolbar Icons	180
Table 48	Business Rules Toolbar Buttons	182
Table 49	Rules for Placement of Subnodes	183
Table 50	Java Source Editor Toolbar Icons	184
Table 51	Method Access Nodes	188
Table 52	Business Rules Designer Toolbar Icons	189
Table 53	Boolean Collaboration Methods (Java)	193
Table 54	Comparison Collaboration Methods (Java)	194
Table 55	Math Collaboration Methods (Java)	196
Table 56	Math Collaboration Methods (Java)	197
Table 57	Object Collaboration Methods (Java)	199
Table 58	String Collaboration Methods (Java)	202
Table 59	Array Operation Collaboration Methods (Java)	205
Table 60	Operators (Java)	207
Table 61	Byte-Array Encoding Converters - Japanese	245
Table 62	Byte-Array Encoding Converters - Korean	245
Table 63	Encoding Converter Methods Dialog Box Options	247
Table 64	Wizard Navigation Buttons	260
Table 65	XSLT Toolbar Icons	267
Table 66	Operator Collaboration Methods (XSLT)	269
Table 67	String Collaboration Methods (XSLT)	272
Table 68	Number Collaboration Methods (XSLT)	275
Table 69	Boolean Collaboration Methods (XSLT)	277
Table 70	Nodes Collaboration Methods (XSLT)	279

Table 71	UAN Extension Collaboration Methods (XSLT)	282
Table 72	UAN Extension Collaboration Method Parameter Definitions	287
Table 73	Environment Icons	292
Table 74	Repository Menu Options	293
Table 75	Environment Menu Options	294
Table 76	Logical Host Menu Options	295
Table 77	Integration Server Menu Options	297
Table 78	Integration Server Menu Options	298
Table 79	Environmental Constants Panel Icons	300
Table 80	Logical Host Configuration Properties List	303
Table 81	Management Agent Configuration Properties List	304
Table 82	Top-level IS Configuration Properties List	305
Table 83	Web Container Configuration Properties List	307
Table 84	Default Web Server Properties List	308
Table 85	Profiling Configuration Properties List	310
Table 86	Security Realm Configuration Properties List	311
Table 87	Application Manager Configuration Properties List	313
Table 88	Integration Server Configuration Properties List	314
Table 89	Oracle JDBC Connection Pool Properties List	316
Table 90	JMS IQ Manager - General Configuration Properties List	322
Table 91	Performance Configuration Properties List	323
Table 92	Messaging Behavior Configuration Properties List	325
Table 93	Sun Java System Configuration Properties List	326
Table 94	Active Directory Service Configuration Properties List	327
Table 95	Diagnostics Configuration Properties List	328
Table 96	Stable Storage Configuration Properties List	330
Table 97	Deployment Toolbar Buttons	333
Table 98	UDDI Registry Information	351
Table 99	Web Service Application Properties	353
Table 100	Terminology Cross-Reference	376

Chapter 1

Introduction

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

1.1 Purpose and Scope

This User's Guide provides general information about the features and operation of SeeBeyond® eGate Integrator in creating and deploying eGate Projects. For information on eGate Integrator system management, see the *eGate Integrator System Administration Guide*.

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 eGate Projects.*

1.2 Intended Audience

This User's Guide is intended for personnel who are involved in integrating software applications using eGate Integrator. To a large extent, these are individuals who will be using the eGate Enterprise Designer to build eGate Projects to accomplish this task. This guide also provides a basic overview of the eGate product for those attempting to gain a general understanding of how eGate Integrator works.

This guide assumes that the reader is an experienced computer user, familiar with Windows-style GUI operations, and also has an in-depth understanding of the operating system(s) on which eGate Integrator will be installed.

Note: *The eGate Integrator graphical user interface (GUI) runs only on Windows. Refer to the SeeBeyond ICAN Suite Installation Guide for a list of operating systems on which eGate Integrator itself can run.*

1.3 Organization of Information

This document provides information about eGate Integrator 5.0 and includes the following chapters and appendices:

- **Chapter 1 "Introduction"** describes the purpose of this User's Guide, including writing conventions and a list of related documents.
- **Chapter 2 "System Overview"** provides an overview of the general structure, architecture, and operation of eGate Integrator, and it's place within the SeeBeyond ICAN Suite.
- **Chapter 3 "Enterprise Manager"** provides a detailed overview of the Enterprise Manager, including its structure and operation.
- **Chapter 4 "Enterprise Designer"** provides a detailed overview of the Enterprise Designer, including its structure and operation.
- **Chapter 5 "eGate Projects"** explains how to create a Connectivity Map and use the Configuration Editor to modify eWay and JMS connections between Connectivity Map components.
- **Chapter 6 "Object Type Definitions"** describes how to create Object Type Definitions (OTDs).
- **Chapter 7 "Collaboration Definitions (Java)"** describes how to build Collaboration Definitions described in Java.
- **Chapter 8 "Collaboration Definitions (XSLT)"** describes how to build Collaboration Definitions described in XSLT.
- **Chapter 9"Environments"**explains how to create and populate eGate Environments, and how to configure and start Logical Hosts.
- **Chapter 10 "Project Deployment"** explains how to create and activate Deployment Profiles.
- **Chapter 11 "Web Services"** describes how to use eGate Integrator in concert with other ICAN Suite components to create Web services.

In addition, the **Glossary** on page 369 lists various terms used in this User's Guide.

1.4 Writing Conventions

The following writing conventions are observed throughout this document.

Table 1 Writing Conventions

Text	Convention	Example
Button, file, icon, parameter, variable, method, menu, and object names.	**Bold** text	▪ Click **OK** to save and close. ▪ From the **File** menu, select **Exit**. ▪ Select the **logicalhost.exe** file. ▪ Enter the **timeout** value. ▪ Use the **getClassName()** method. ▪ Configure the **Inbound** File eWay.
Command line arguments and code samples	`Fixed` font. Variables are shown in ***`bold italic`***.	`bootstrap -p` ***`password`***
Hypertext links	**Blue** text	**http://www.seebeyond.com**

Additional Conventions

Windows Systems

For the purposes of this guide, all references to **Windows** apply to Microsoft Windows Server 2003, Windows XP, and Windows 2000.

Path Name Separator

This guide uses a backslash (\) as the separator within path names. If you are working on a UNIX system, please substitute a forward slash (/).

1.5 Supporting Documents

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

- *eGate Integrator JMS Reference Guide*
- *eGate Integrator Release Notes*
- *eGate Integrator System Administration Guide*
- *eGate Integrator Tutorial*
- *SeeBeyond ICAN Suite Deployment Guide*
- *SeeBeyond ICAN Suite Installation Guide*
- *SeeBeyond ICAN Suite Primer*

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 SeeBeyond documentation is included in the *SeeBeyond ICAN Suite Primer*.

The documentation for the SeeBeyond ICAN Suite is distributed as a collection of online documents, which can be accessed through the Enterprise Manager (see **Documentation** on page 39). 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

1.6 The SeeBeyond Web Site

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

http://www.seebeyond.com

Chapter 2

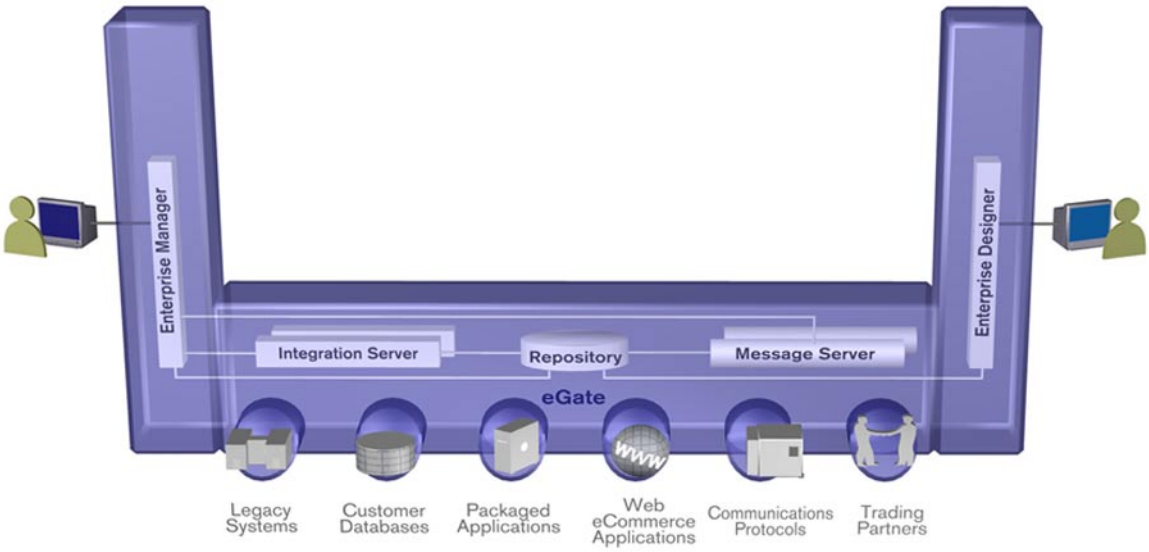
System Overview

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

2.1 Introduction

SeeBeyond® eGate™ Integrator is a fully J2EE certified and Web services-based, distributed integration platform that serves as the foundation of the SeeBeyond Integrated Composite Application Network™ (ICAN™) Suite. eGate Integrator provides the core integration platform, comprehensive systems connectivity, guaranteed messaging and robust transformation capabilities while providing a unified, single sign-on environment for integration development, deployment, monitoring and management. eGate Integrator supports portability of integrations across common J2EE application servers through a completely open, J2EE-certified and Web services-based architecture.

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 enterprise. An integrated UDDI-compliant server allows publication and discovery of Web services. The run-time

environment employs J2EE-compliant integration servers as operational engines and JMS-compliant message servers for the propagation of messages. The flexibility of the eGate system allows the option of deployment to a SeeBeyond run-time environment or to third-party application servers, 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 ICAN Suite components. Enterprise Designer provides a unified, graphical development environment for integrating systems and developing composite applications using Web services.

eGate Integrator can communicate with and link multiple applications and databases across a variety of different operating systems. eGate 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

SeeBeyond addresses application integration by means of an eGate 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.

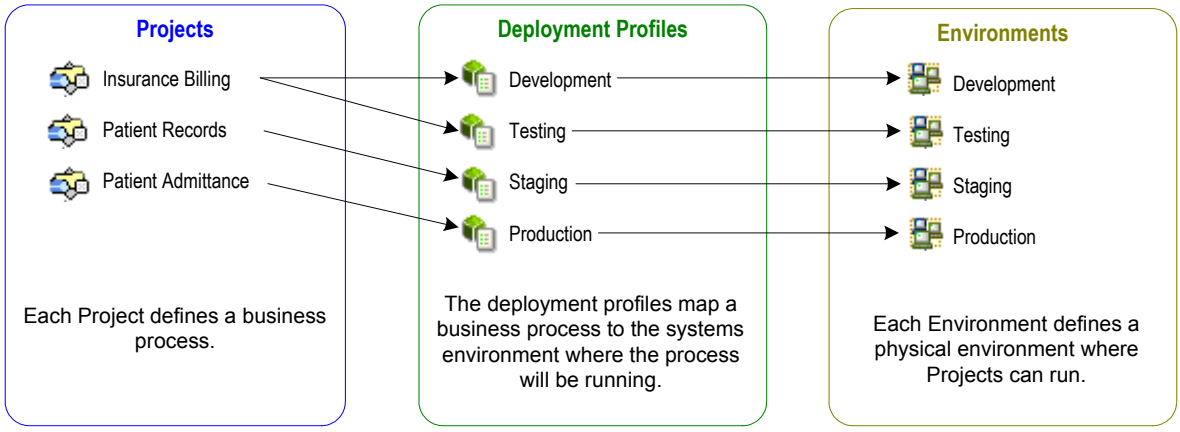
Projects are created using tools contained within Enterprise Designer and, once deployed, can be run and monitored using Enterprise Manager. Projects can also be set up to be run from the business process level using the SeeBeyond eInsight Business Process Manager, if that product is also installed.

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 the 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 components. This provides you with extensive flexibility and efficiency in designing eGate Integrator implementations. For example, once you build your Projects and Environments, you have the flexibility to change each component without having to make changes to the other component.

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



In the figure above, any of the Projects 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.

In broad outline, implementing an integration Project using eGate Integrator includes the following steps:

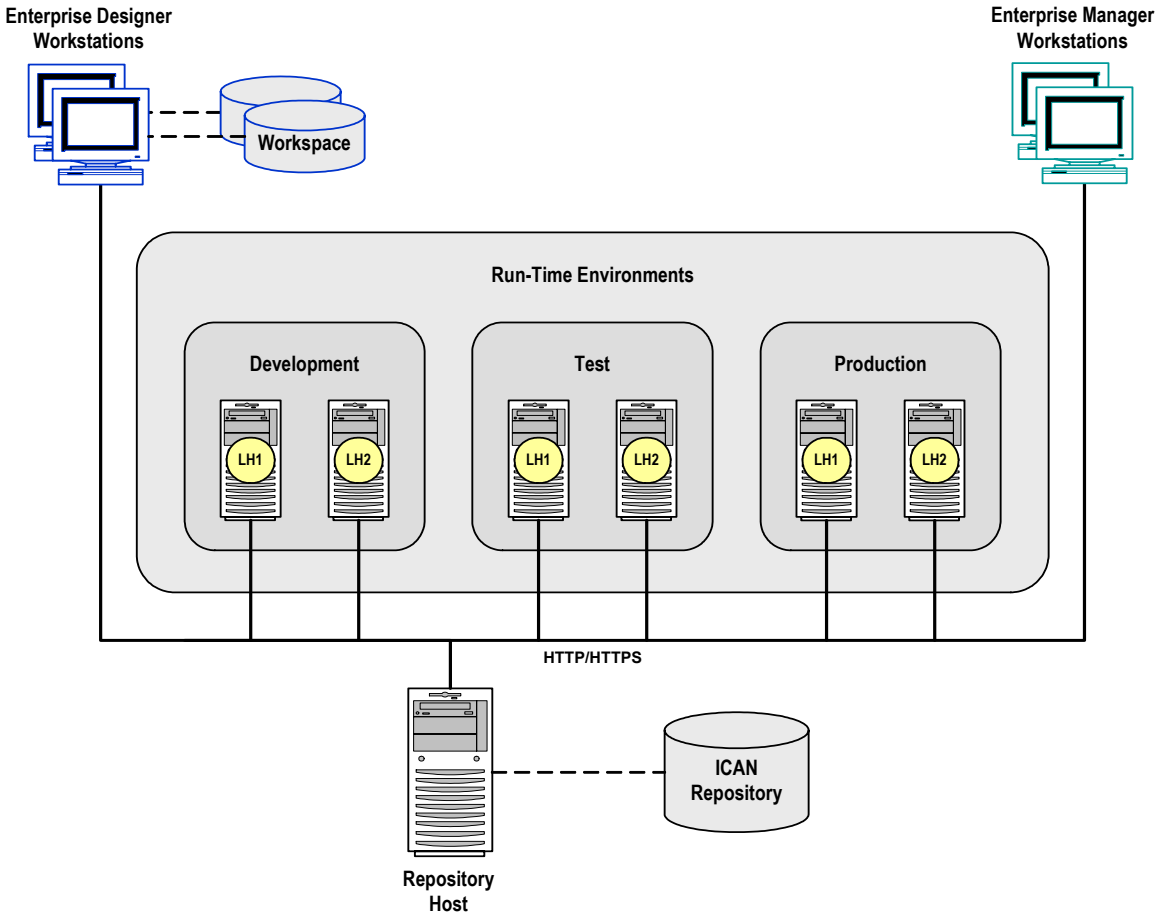
1. **Design your Project.**
2. **Define your Environments.**
3. **Create your Deployment Profiles.**
4. **Deploy the eGate Project.**

These implementation steps are all accomplished using Enterprise Designer, which is introduced in **Enterprise Designer** on page 32 and developed further in subsequent chapters.

2.3 System Architecture

SeeBeyond's 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 system implementation that is highly distributed.

Figure 3 Distributed eGate Integrator System



Note: *In this scenario it is assumed that all instances of eGate 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. The Repository also stores all of the product binary files that are required at run time by the Logical Hosts. The components and configurations are downloaded to the Logical Host during the initial bootstrap process and as needed after design-time configuration changes are made.

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 ICAN 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. The Repository makes Web Services available via a UDDI registry.

2.3.2 Run-Time Environments

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

- **Logical Hosts**
  Each Environment contains one or more system definitions. Each definition must include one or more **integration servers** such as the SeeBeyond Integration Server, which are the engines that run eGate Collaborations and eWays, and one or more **message servers** such as the 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*.
- **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, each supporting a single Logical Host. 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 eGate 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 (CPU, memory and disk) are sufficient to support concurrent usage.

2.4 User Interfaces

eGate Integrator provides two basic graphical user interfaces (GUIs), each of which addresses a different set of users. Enterprise Manager is an interface used by the entire ICAN Suite, the primary users of which are system administrators. Enterprise Designer is used by personnel who are involved in defining a software system for integrating the various enterprise applications using eGate Integrator and other ICAN Suite products.

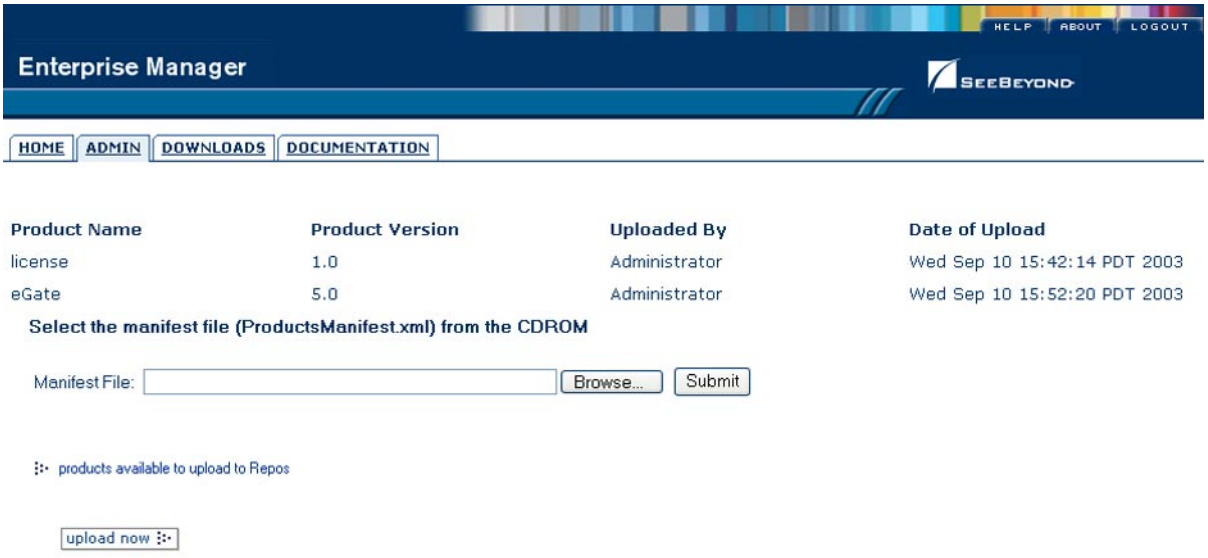
2.4.1 Enterprise Manager

Enterprise Manager is a Web-based application that works within Microsoft Internet Explorer. It is used throughout the SeeBeyond ICAN Suite for:

- Installing and updating ICAN Suite products
- Accessing ICAN Suite product documentation
- Managing and monitoring runtime components

The Enterprise Manager is described in **Enterprise Manager** *on page 35*. Figure 4 shows the Enterprise Manager **Admin** page, used in product installation.

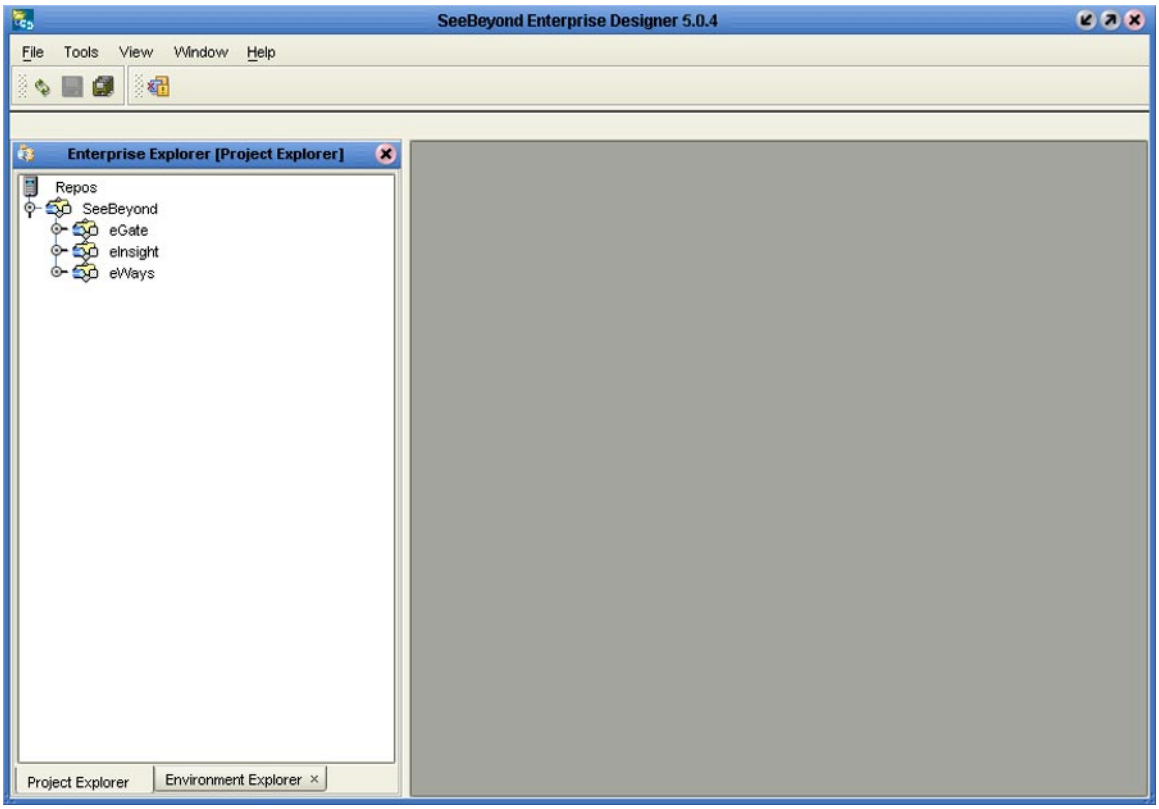
Figure 4 SeeBeyond Enterprise Manager



2.4.2 Enterprise Designer

The SeeBeyond Enterprise Designer is used to create and configure the logical components and physical resources of an eGate Project. Through this GUI (see Figure 5), you can develop Projects to process and route data through an eGate Integrator system. Enterprise Designer is also used by other components of the ICAN Suite, including eInsight Business Process Manager and eXchange.

Figure 5 Enterprise Designer

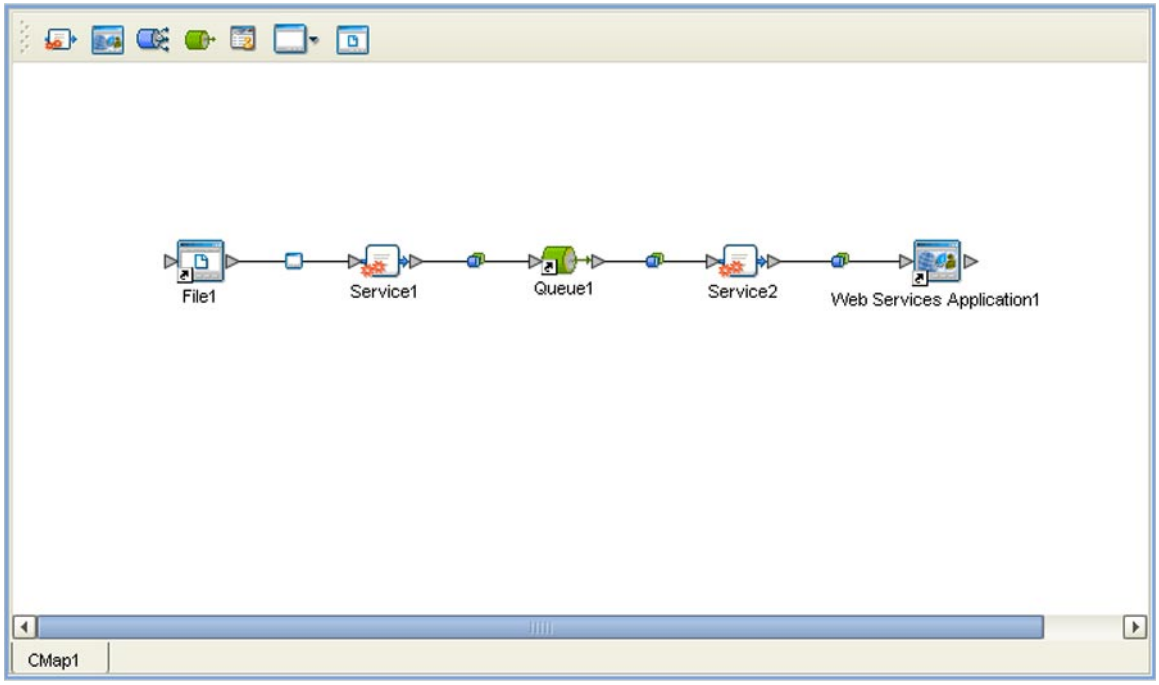


The major features of the Enterprise Designer are the Enterprise Explorer on the left, and an editor panel on the right—which is initially blank. The Enterprise Explorer follows the familiar Windows Explorer format, displaying a tree structure. The editor panel displays a variety of editors, depending upon what component is selected in the Enterprise Explorer. These editors include, for example:

- Connectivity Map Editor
- OTD Editor
- Java Collaboration Editor
- XSLT Collaboration Editor
- Environment Editor
- Deployment Editor

The Connectivity Map Editor (see Figure 6) provides a graphic example of one of these editors, in which logical components of a Project can be created and connected. eGate uses Connectivity Maps to intuitively configure the end-to-end flow of messages within an integration. The integration developer can to drag and drop the various Collaborations, Intelligent Queues and external-system eWay adapters 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 **eGate Projects** on page 92.

Figure 6 Connectivity Map Editor



The Enterprise Designer also includes the design-time functionality for other ICAN products, such as eInsight and eXchange. For more information on using other ICAN products in the Enterprise Designer, see the product documentation for those products.

For more information on the Enterprise Designer, see **Enterprise Designer** *on page 54*.

Chapter 3

Enterprise Manager

This chapter provides an introduction to the ICAN Suite Enterprise Manager.

3.1 Overview

Enterprise Manager is a Web-based interface with which you can install and update eGate Integrator, and monitor and manage deployed eGate components.

Important: *Enterprise Manager works only with Microsoft Internet Explorer.*

3.1.1 Installing and Updating eGate

eGate Integrator components are uploaded from the installation media (CD-ROMs) to the Repository server via the Enterprise Manager. These products are then available to be downloaded and installed from the Repository server. For information on installing and updating eGate components, see the *SeeBeyond ICAN Suite Installation Guide*.

3.1.2 Monitoring and Managing eGate

The Enterprise Manager allows you to monitor and manage deployed eGate components in real-time.

- **The ICAN Monitor** on page 40 describes the basic features of the ICAN Monitor interface. ICAN Monitor usage for specific tasks is described in the *eGate Integrator System Administration Guide*.
- **The SRE Monitor** on page 41 describes an optional facility that allows you to monitor and manage e*Gate 4.x schemas in eGate 5.0, using the Schema Runtime Environment.

3.2 Starting Enterprise Manager

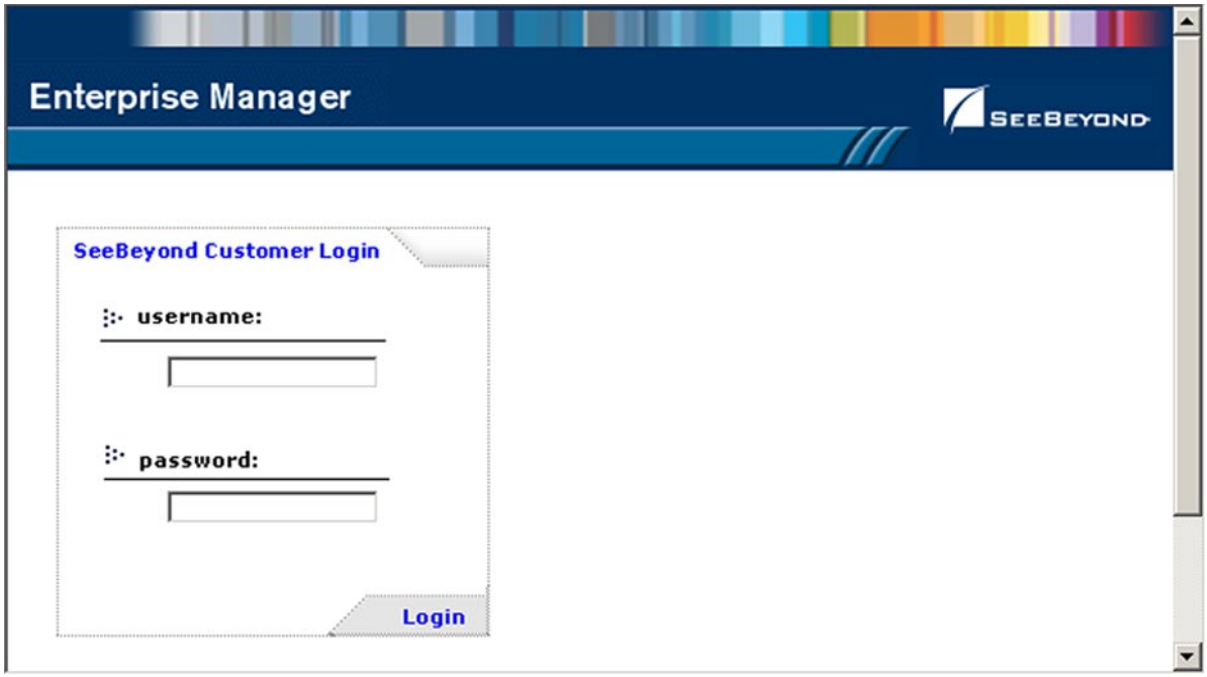
To start the Enterprise Manager

1. Launch Internet Explorer.
2. Enter **http://*hostname:portnumber*** in the **Address** box to display the SeeBeyond Customer Login window shown in Figure 7.

Note: *The **hostname** is the fully-qualified, network-addressable host name of the server where you installed the Repository. The **portnumber** is the number of the port you entered during installation of the Repository. See the SeeBeyond ICAN Suite Installation Guide.*

Important: *The TCP/IP host name must be alphanumeric.*

Figure 7 Enterprise Manager Login

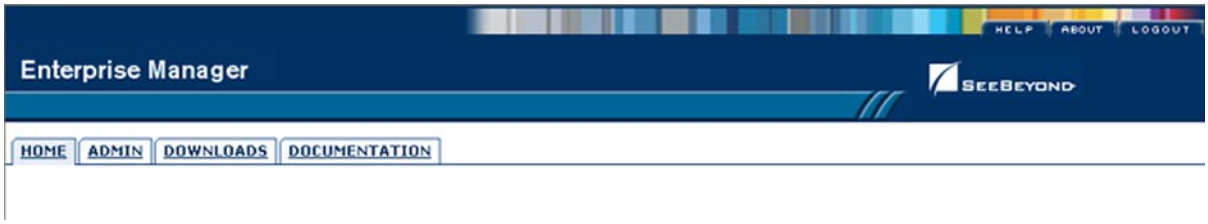


3. Enter your login ID and password in the **Username** and **Password** boxes and click **Login**.

3.3 The Enterprise Manager Interface

Once you have logged in, you see the full Enterprise Manager user interface (see Figure 8).

Figure 8 Enterprise Manager GUI



The Enterprise Manager is organized into four pages, as described in the following table. Each page is accessed by clicking the appropriate tab.

Table 2 Enterprise Manager - Pages

Page	Function
Home	The Home page is used for accessing the ICAN Monitor, which is the main subject of this chapter. See **Home** on page 38.
Admin	The Administration page is used for installing and updating ICAN components. See the *SeeBeyond ICAN Suite Installation Guide* for information.
Downloads	The Downloads page is used in installing and updating ICAN components. See the *SeeBeyond ICAN Suite Installation Guide* for information.
Documentation	The Documentation page is used for accessing ICAN Suite documentation. See **Documentation** on page 39, and the following *Note*.

Note: *You must download the documentation SAR files from the installation disk before you can access any documents using the Documentation page (see the SeeBeyond ICAN Suite Installation Guide).*

There are also three small tabs in the upper-right corner of the Enterprise Manager, which are described in the following table.

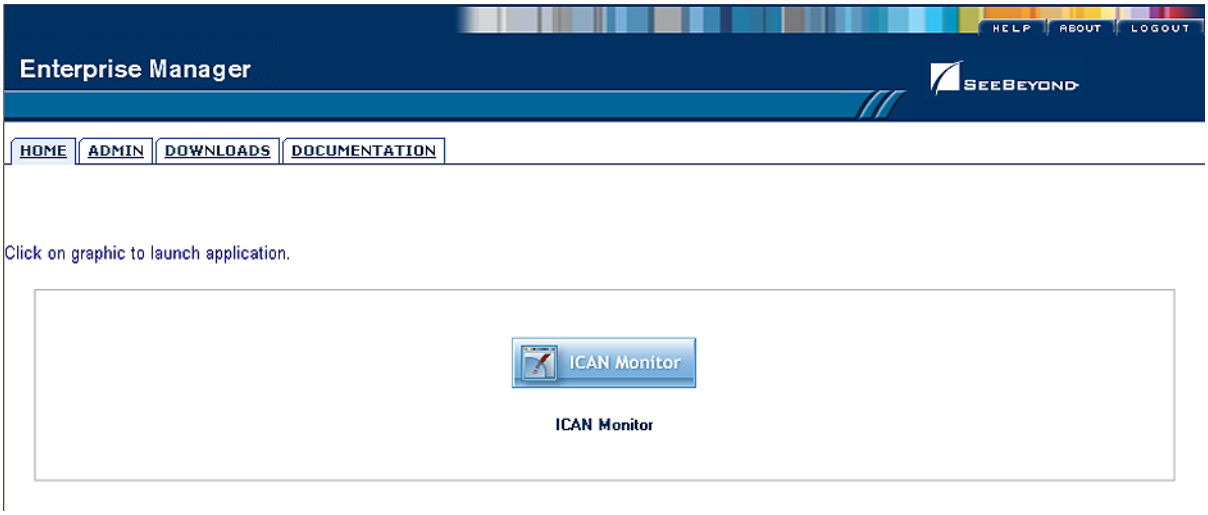
Table 3 Enterprise Manager - Control Tabs

Tab	Function
Help	The Help tab provides access to the online help system.
About	The About tab displays the installed version of the product (this tab is not present on the Documentation page).
Logout	The Logout tab logs you out of the Enterprise Manager and returns you to the Login page.

3.3.1 Home

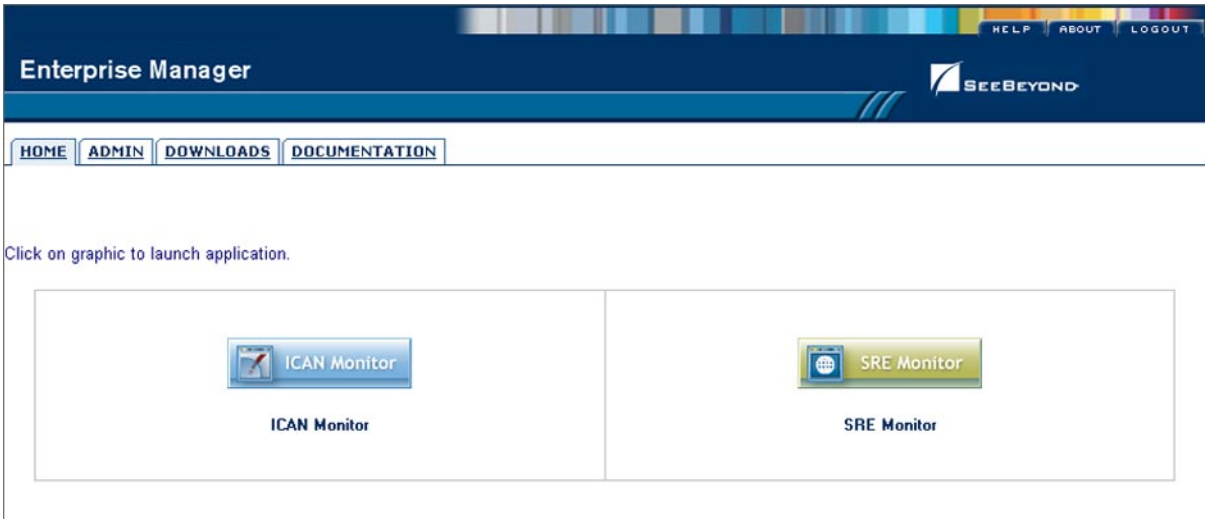
The Enterprise Manager's **Home** page (see Figure 9) contains a link to the ICAN Monitor. Click the **Monitor** icon to launch the ICAN Monitor (see **The ICAN Monitor** on page 40).

Figure 9 ICAN Monitor Launch Icon



If the SRE Monitor (see **The SRE Monitor** on page 41) is installed, its icon is also displayed on this page, as shown in Figure 10.

Figure 10 ICAN Monitor/SRE Monitor Launch Icons

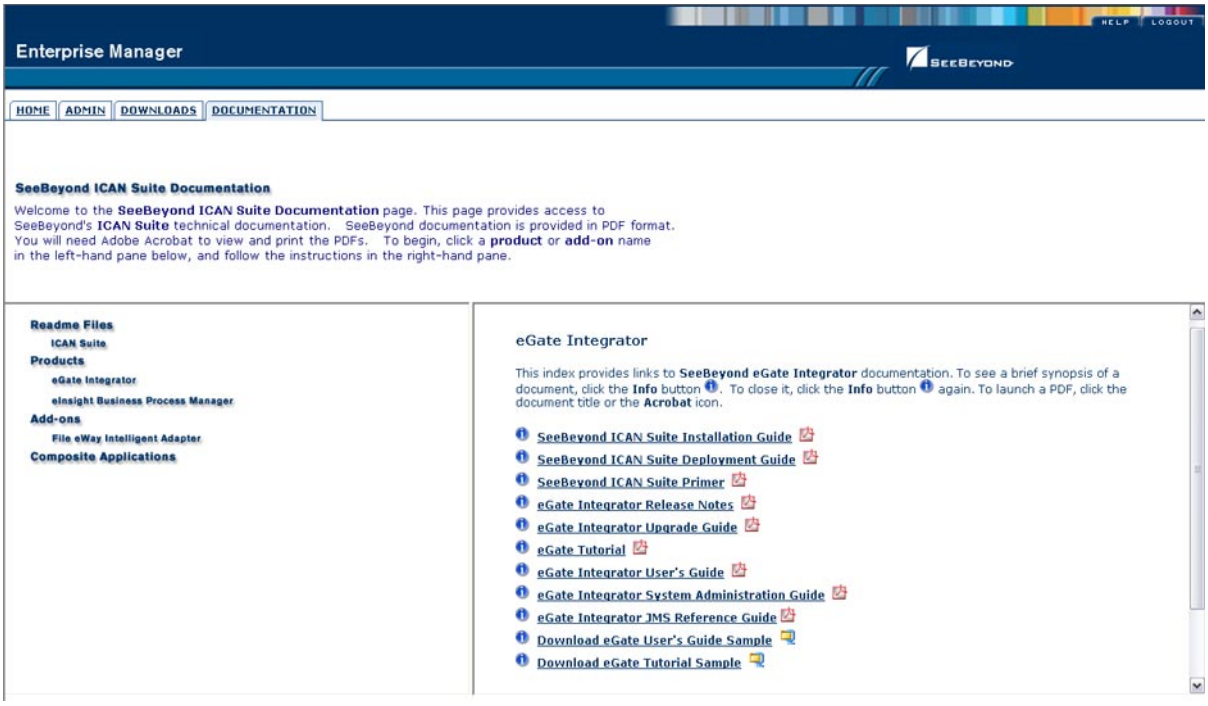


Note: *If connection problems are encountered, close all Internet Explorer windows and retry.*

3.3.2 Documentation

The **Documentation** page (see Figure 11) contains links to the latest versions of the SeeBeyond ICAN documentation in Adobe Acrobat (PDF) format, and also any sample Project files (in ZIP format). Shown is the current set for eGate Integrator.

Figure 11 Documentation Page



The provided documentation is organized into the major categories listed in Table 4.

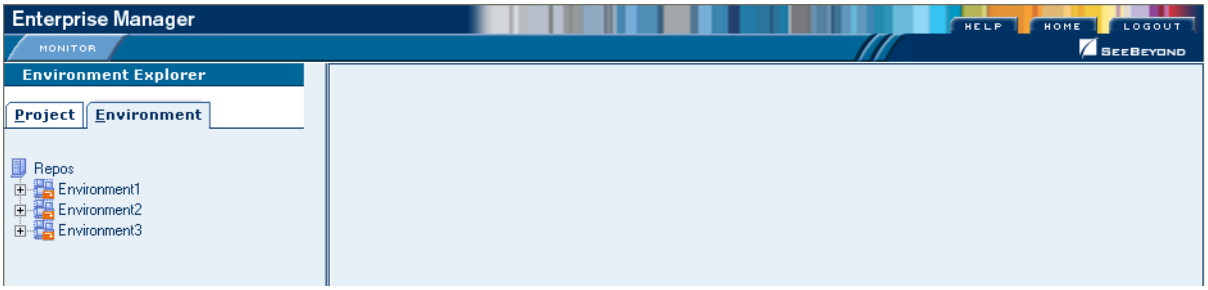
Table 4 Document Categories

Category	Contents
Readme Files	Includes information regarding the latest operating system and hardware requirements, cautions and caveats regarding known issues, and supplementary information arising after the publication of other documentation.
Products	Documentation regarding SeeBeyond core products, such as eGate Integrator and eInsight Business Process Manager. Also includes example Project files, if available.
Add-ons	Documentation regarding optional, ancillary products such as eWays and OTD Libraries.

3.4 The ICAN Monitor

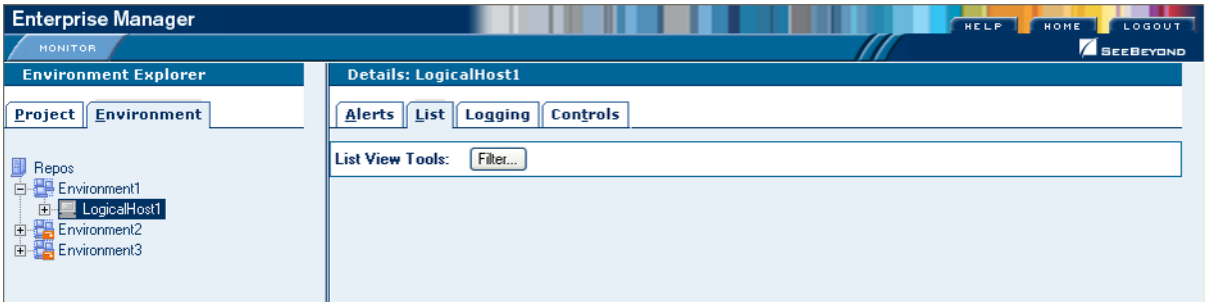
The ICAN Monitor has structure similar to that of the Enterprise Designer, with an Explorer panel on the left and a Details panel on the right. Initially, the Details panel is blank as shown in Figure 12.

Figure 12 ICAN Monitor Interface - Initial



Like the Enterprise Manager itself, the ICAN Monitor's **Details** area is organized into sections represented by tabs (see Table 5). Which tabs are present depends upon the component selected in the Explorer. For example, selecting the Logical Host displays the Monitor page shown in Figure 13.

Figure 13 ICAN Monitor Interface - Environment



At times, the Details panel will have two parts, to display an additional level of information. In this case, different tabs will be displayed in the upper and lower panels.

Table 5 ICAN Monitor Interface - Details Tabs

Tab	Function
Alerts	Displays all alerts for the component selected in the Explorer.
List	Displays a list presenting information about the component selected in the Explorer.
Logging	Displays all log messages for the component selected in the Explorer.
Controls	Displays controls that allow an Administrator to intervene in the run-time process and perform tasks such as starting and stopping components.

Note: *See the eGate Integrator System Administration Guide for detailed information regarding Monitor usage.*

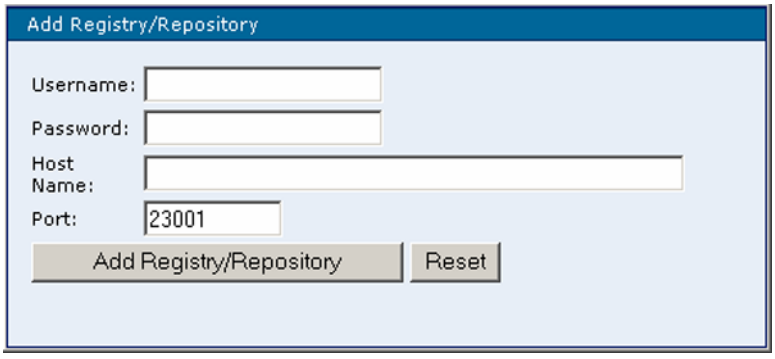
3.5 The SRE Monitor

eGate 5.0 provides a completely different operating environment from earlier versions of the product (e*Gate). The Schema Runtime Environment (SRE) allows you to use schemas developed for e*Gate 4.x with eGate 5.0 by providing the necessary environmental components. Instructions for installing and using the SRE are contained in the SeeBeyond documentation for the SRE.

The SRE Monitor enables you to manage e*Gate 4.x schemas running in the Schema Runtime Environment from within eGate 5.0. The SRE Monitor interface generally resembles the ICAN Monitor, but differs somewhat in detail (see Figure 14). Only the Environment Explorer is displayed, which has two additional icons in the upper left corner; these are described in Table 6.

Figure 14 SRE Monitor

Table 6 SRE Monitor Explorer Icons

Icon	Function
	The **Add Registry/Repository** icon displays the **Add Registry/Repository** dialog box, in which you specify the desired Registry or Repository's name and port, and your user name and password. 
	The **Refresh Registry** icon refreshes the SRE Registry and the Explorer tree following changes to component status.

Instructions for installing the SRE Monitor are contained in the *SeeBeyond ICAN Suite Installation Guide*.

3.5.1 Starting the SRE Monitor

To start the SRE Monitor

1. Start the SRE Monitor server, as described in the *SeeBeyond ICAN Suite Installation Guide*.
2. Launch Internet Explorer and access Enterprise Manager, as described in **Starting Enterprise Manager** on page 36.
3. On the Enterprise Manager Home page, click the **SRE Monitor** icon shown in Figure 15 to display the initial page of the SRE Monitor (see Figure 16).

Figure 15 Enterprise Manager Home Page

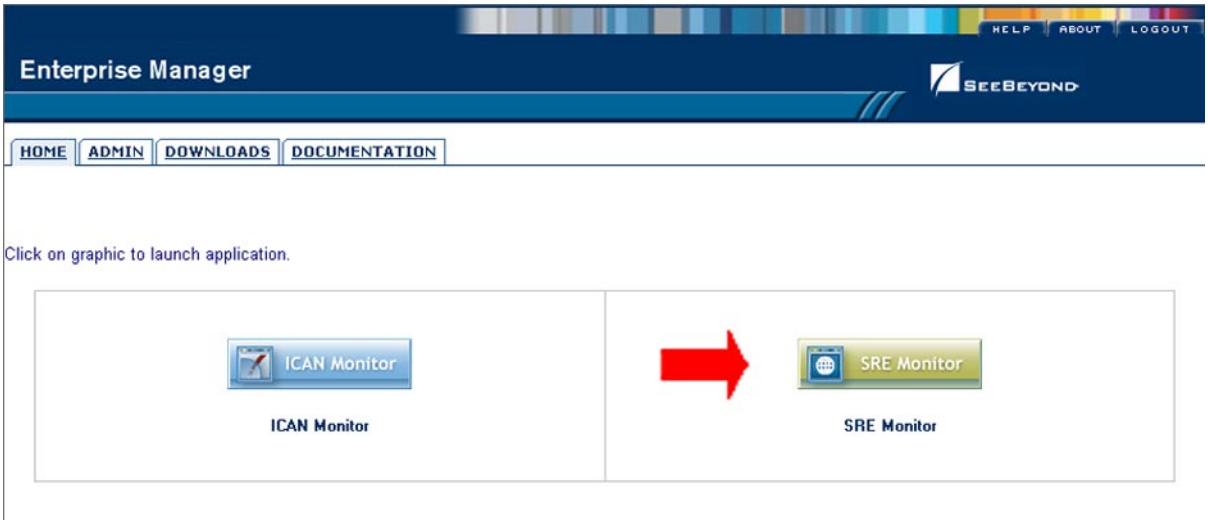
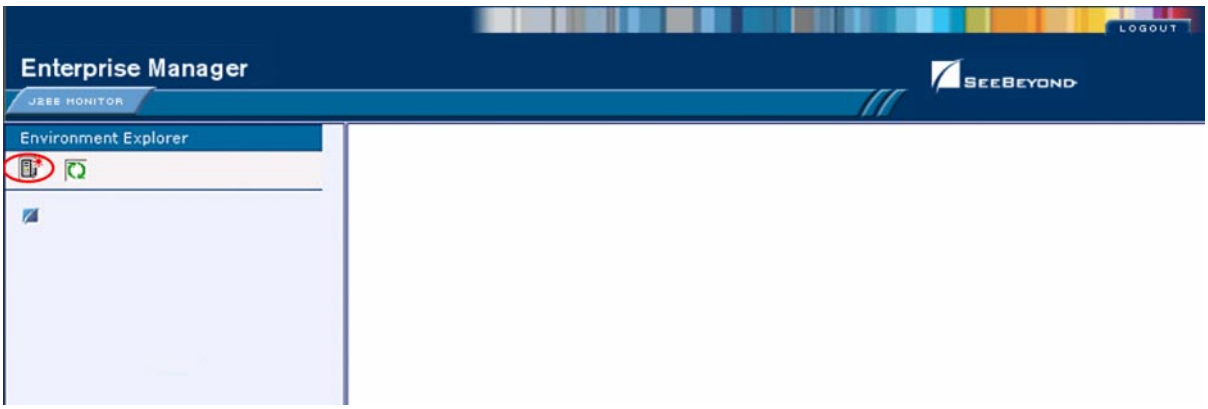
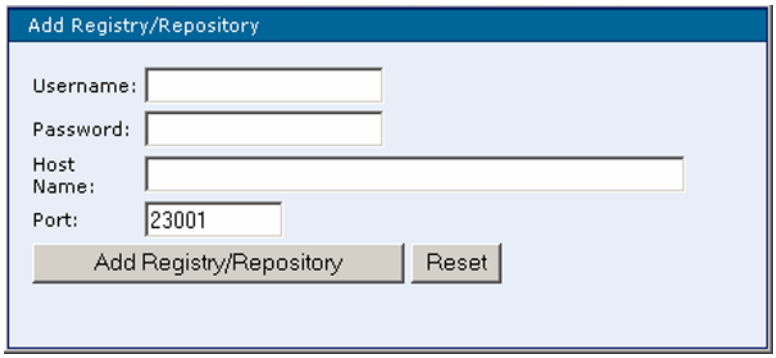


Figure 16 SRE Monitor Initial Page



4. Click the **Add Registry/Repository** icon in the upper left corner of the Explorer. This displays the dialog box shown in Figure 17.

Figure 17 Add Registry/Repository Dialog Box



5 Enter your login ID and password, and the Repository Host Name and Port, and click **Add Registry/Repository**.

Note: *The **Host Name** is the host name of the server where you installed the e*Gate 4.x Registry, and the **Port** is the number of the port you entered during installation of the e*Gate 4.x Registry. See the SeeBeyond ICAN Suite Installation Guide.*

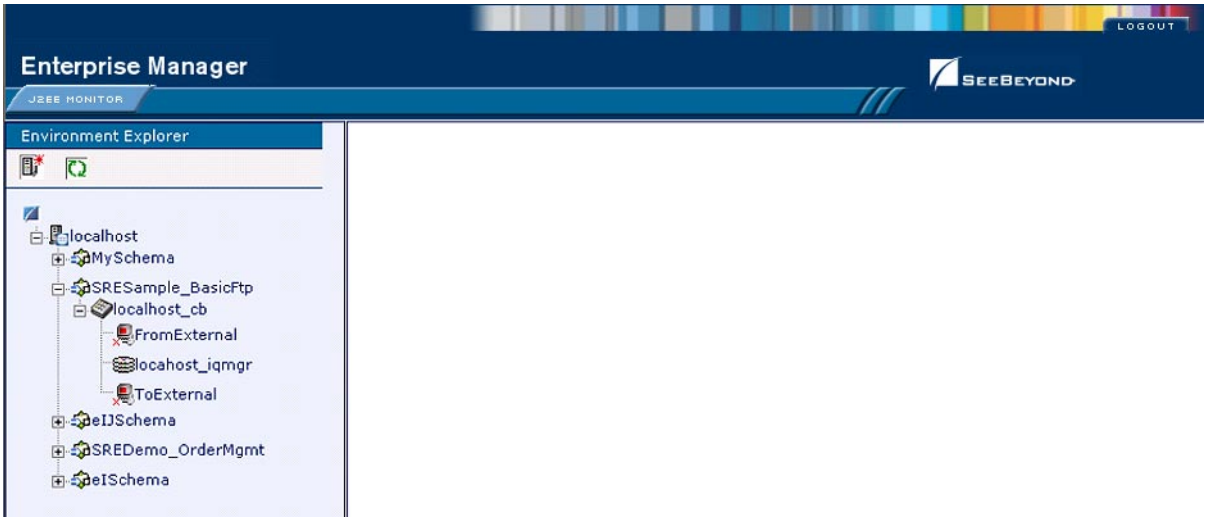
Important: *The Host Name must be composed of alphanumeric characters only.*

Controlling Components

To start and stop components, and view their properties

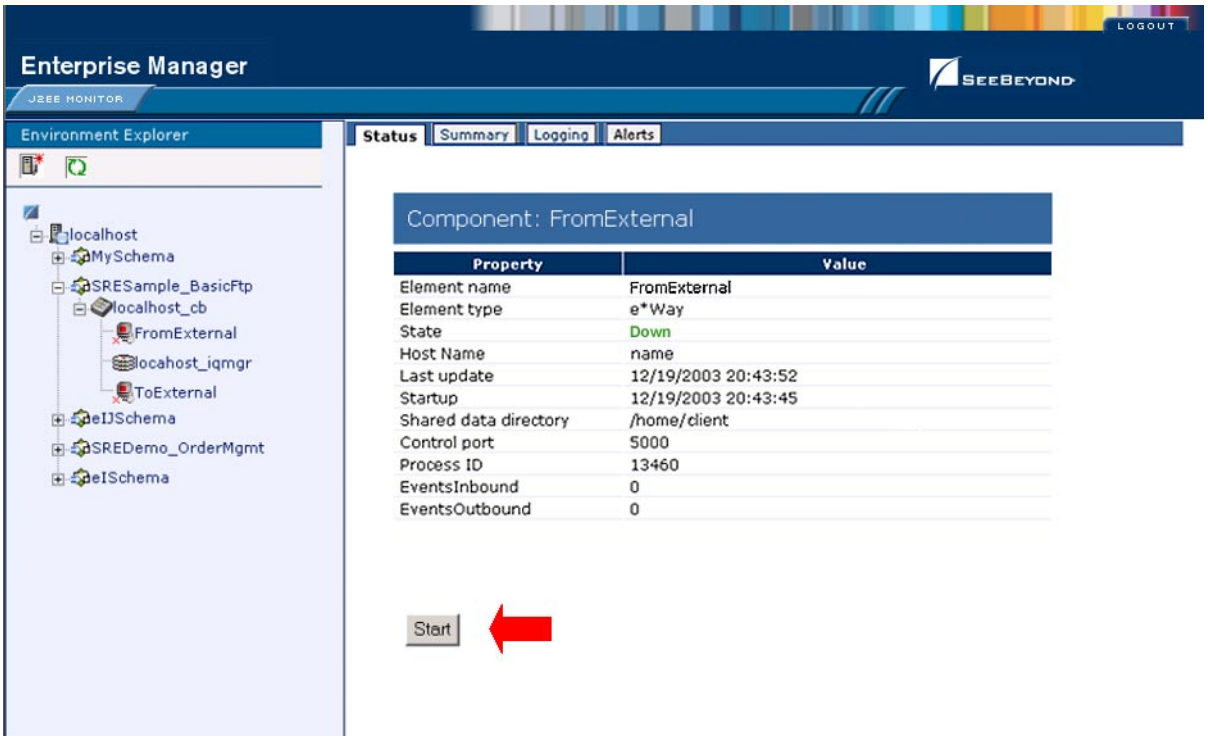
1 Expand the Control Broker in the Explorer tree to view the SRE components, as shown in Figure 18.

Figure 18 Viewing SRE Components



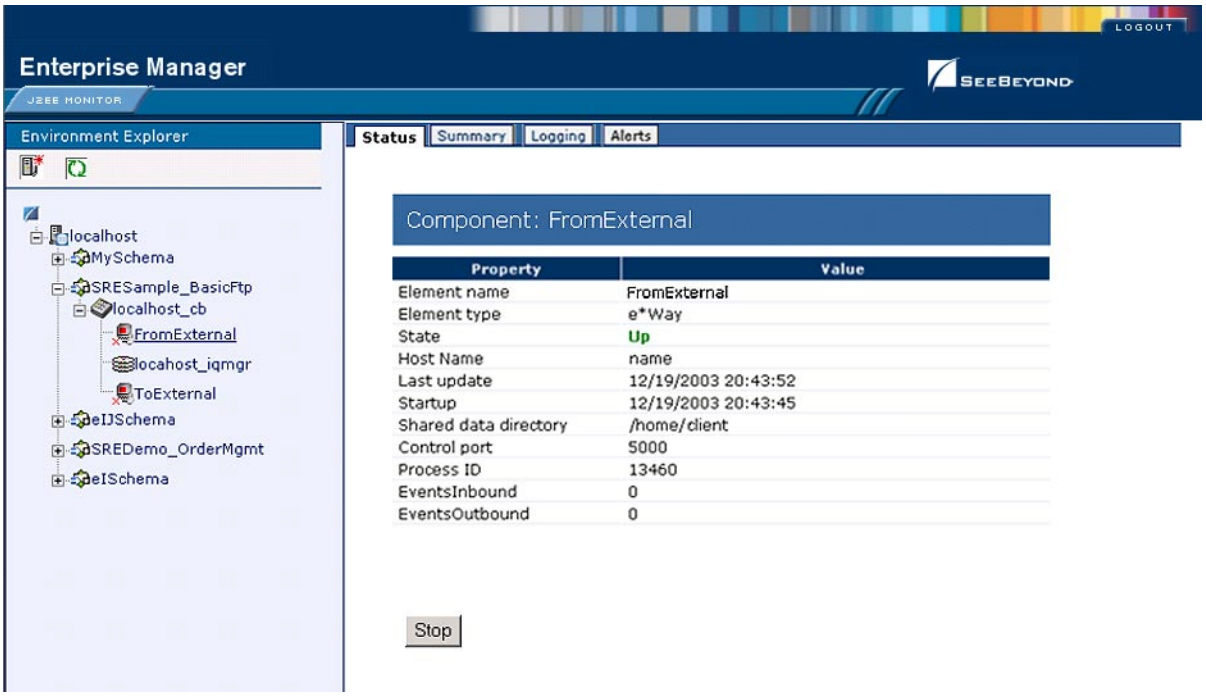
2 Click a component to display its status. As an example, the status of the *FromExternal* e*Way is shown in Figure 19. You can start the component by clicking the **Start** button, which becomes a **Stop** button when the component is running.

Figure 19 FromExternal e*Way Status (Not Running)



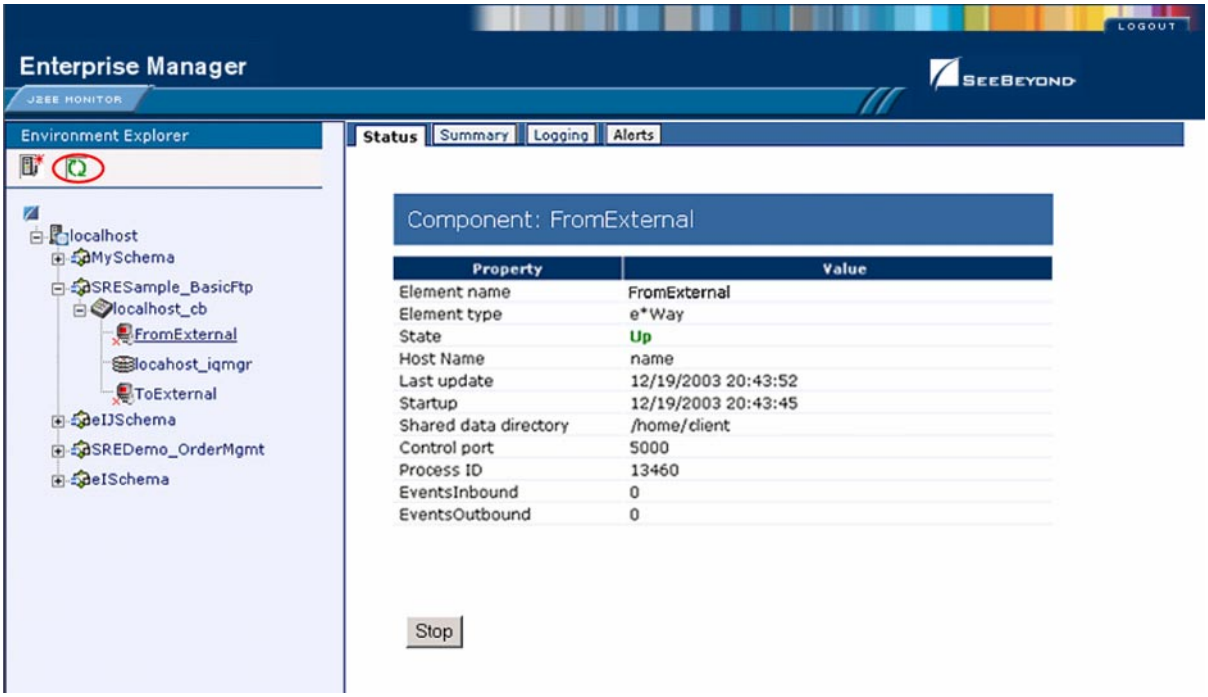
3 The status page is updated after you start or stop the component, as shown in Figure 20.

Figure 20 FromExternal e*Way Status (Running)



4 To refresh the Explorer tree, click the **Refresh Repository** icon, as shown in Figure 21.

Figure 21 Refreshing the Explorer Tree



Viewing Message Destinations

You can view, create, and delete message destinations (topics and queues), and view message journals, using the SRE Monitor (see Figure 22).

Figure 22 Viewing JMS Topics and Queues

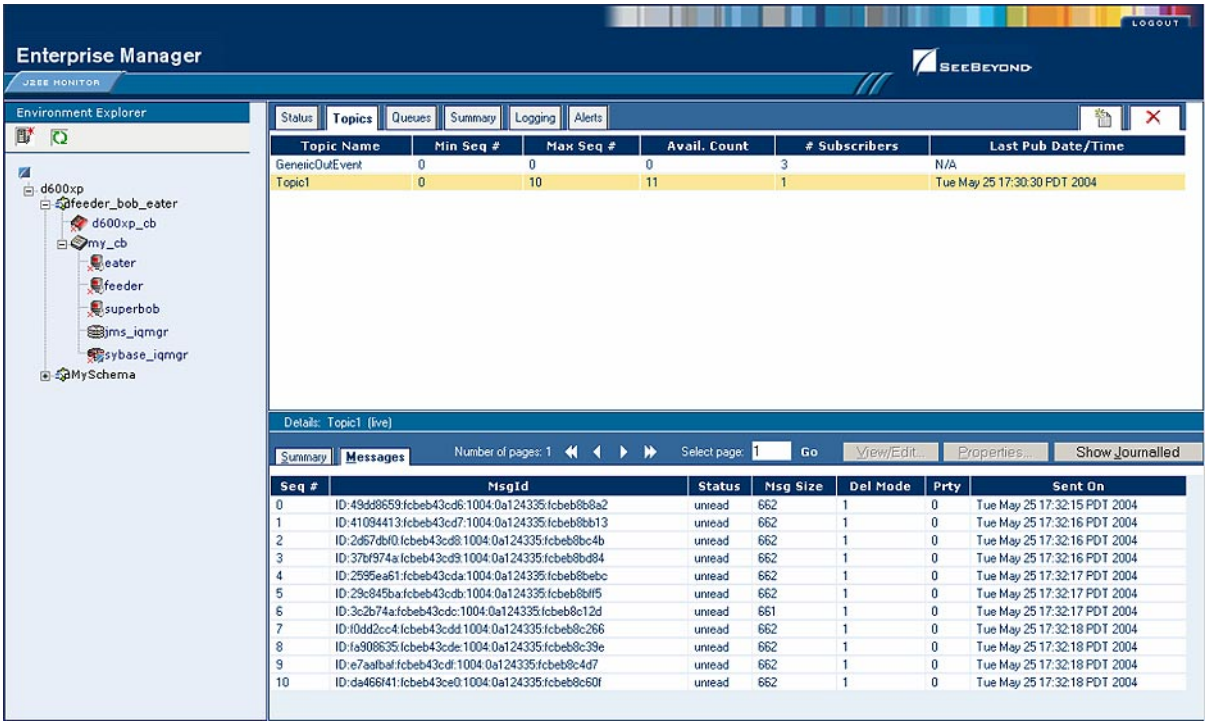


Table 7 SRE Monitor Topic / Queue Icons

Icon	Function
	**Create Topic or Queue** displays a dialog box in which you enter the name of the topic or queue.
	**Delete Topic or Queue** displays a warning message requesting confirmation that you want to delete the topic or queue.

To view message destination summaries

1. Find the topic or queue for which you want to view its summary.
2. Click the **Summary** tab, which displays the *Summary* page.
3. To select a message, you can enter the message number in the **View/Edit** box, or you can drag the message slider.

To view message properties

1. Select the topic or queue for which you want to view a message.
2. In the upper *Details* panel on the Topic or Queue page, click the topic or queue for which you want to view message properties.
3. In the lower *Details* panel, scroll to the desired message to see its properties.

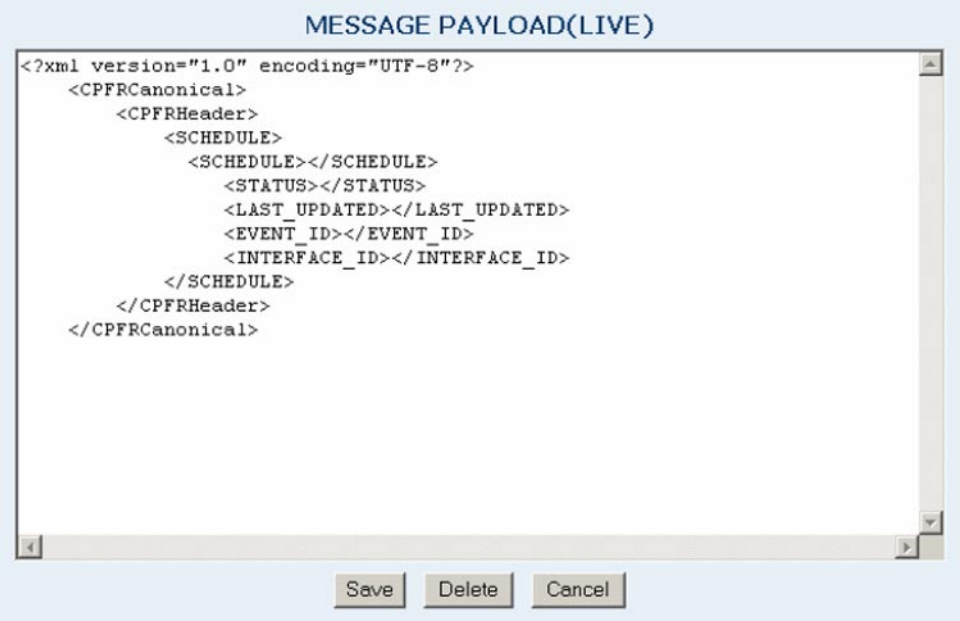
To view message details

1. Select the topic or queue for which you want to view a message.
2. In the lower *Details* panel (**Message** tab), click the message for view you want to view details.
3. Click **Properties** to display the *View Message Property* box.

To view a message payload

1. Find the message for which you want to view the payload.
2. In the lower *Details* panel, click the message for which you want to view the payload.
3. Click **View/Edit**, which displays the *Edit Payload for Message* dialog box (see Figure 23).

Figure 23 Message Payload Dialog



Note: *To edit or delete messages, you must change the JMS Server configuration (see the eGate Integrator User's Guide for SRE).*

To view a journaled message

1. Find the message for which you want to view its journal.
2. Click the **Show Journal** button, which displays the *Journal* page.

Note: *To show journaled messages, you must set the Journal flag in the JMS Server configuration (see the eGate Integrator User's Guide for SRE).*

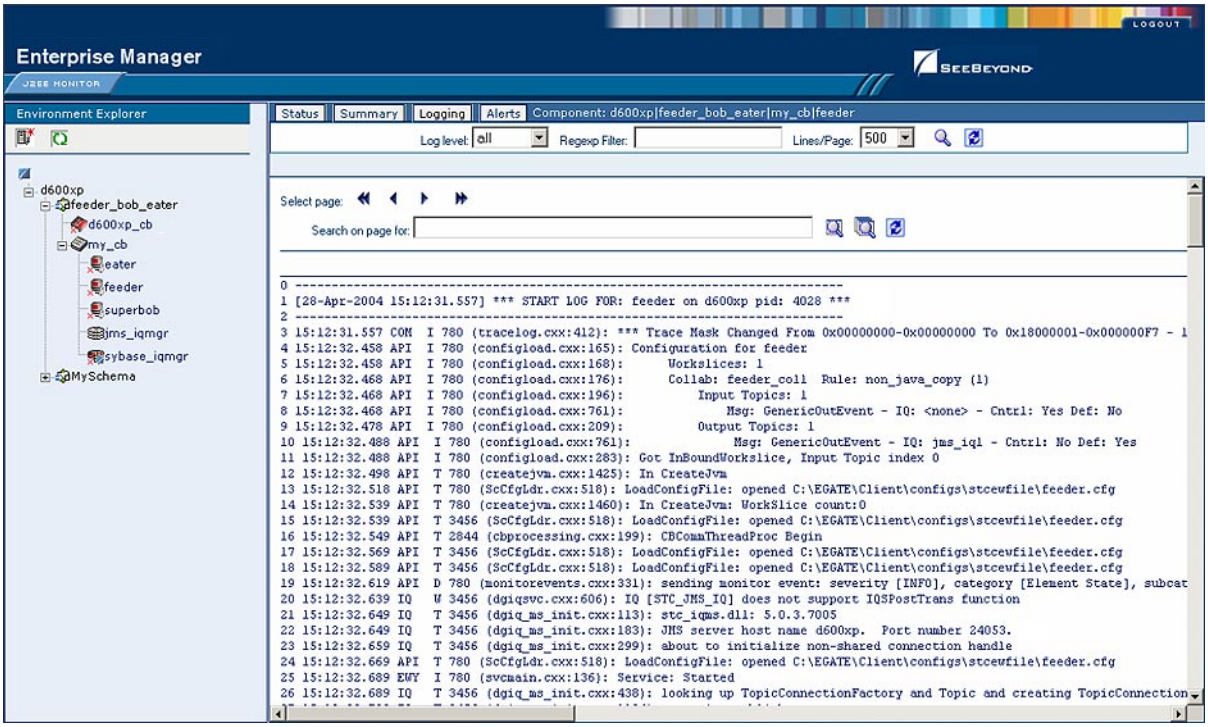
Viewing Log Files

To view a log file

1 In the Environment Explorer, select an Environment or a component.

2 Click the **Logging** tab in the upper Details panel to display the log for the selected item (see Figure 24).

Figure 24 Viewing Logs



The Alerts details panel contains a set of icons, shown in Table 9.

Table 8 SRE Monitor Logging Icons

Icon	Function
	**Search**
	**Find on Page**
	**Find All on Page**
	**Reset / Clear All**

- To filter the log messages for a specific log level, change the setting of the **Log level** drop-down list and click the **Search** icon.
- The **Regexp Filter** field allows you to perform a regular expression search.
- To change the number of lines that appear in each page, change the setting of the **Lines/Page** drop-down list and click the **Search** icon.
- To search for a string in the log file, enter a string and click the **Find on Page** or **Find All on Page** icon. The string must contain at least three characters.

Note: *To view a component's log file, the SRE Monitor must be running on the same computer on which the control broker is running.*

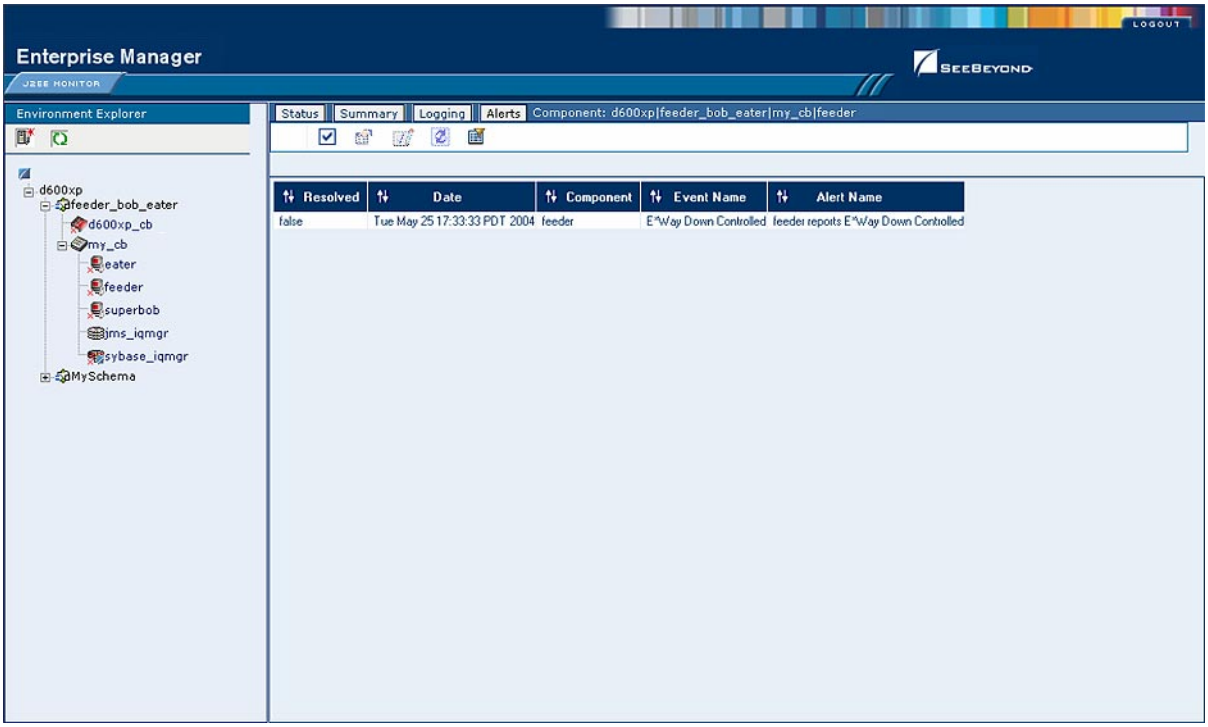
Viewing Alerts

To view an Alert

1 In the Environment Explorer, select an Environment or a component.

2 Click the **Alerts** tab in the upper Details panel to display the alerts for the selected item (see Figure 25).

Figure 25 Viewing Alerts



The Alerts details panel contains a set of icons, shown in Table 9.

Table 9 SRE Monitor Alerts Icons

Icon	Function
	**Select All**
	**View Details**
	**Set Resolved**
	**Reset**
	**Filter**

To view Alert details

1. Select the Alert.
2. Click the **View Details** icon.
3. An information box is displayed showing the details of the Alert.

Note: *You can also double-click the Alert to display the information box.*

To change the Alert status

1. Select the Alert.
2. Click the **Set Resolved** icon.
3. The Alert Status will be set to *Resolved*.

To filter the Alert status

1. Select the Alert.
2. Click the **Filter** icon.
3. A dialog box is displayed, in which you can set filtering parameters to control which Alerts appear in the monitor.

Chapter 4

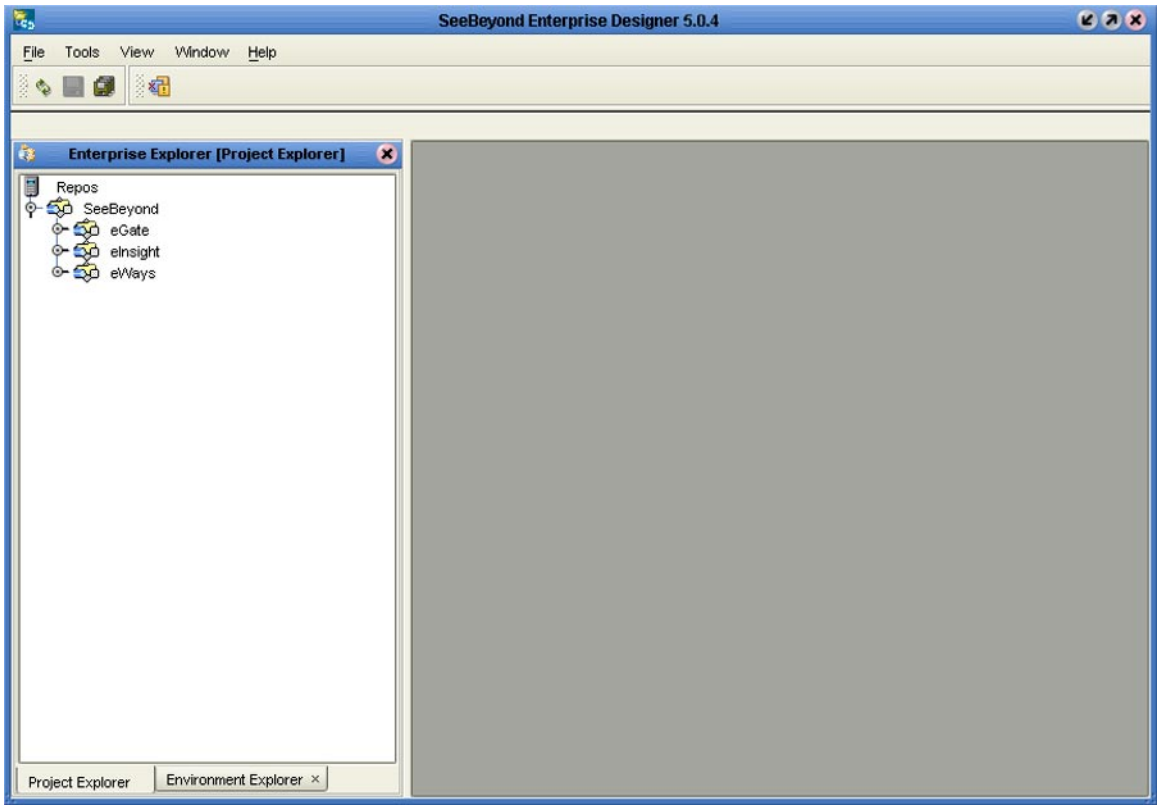
Enterprise Designer

This chapter presents an overview of the major features of the Enterprise Designer.

4.1 Overview

The Enterprise Designer graphical user interface (GUI) is used to create and configure the logical components and physical resources of an eGate Project. Through this GUI (see Figure 26), you can develop Projects to process and route data through an eGate Integrator system.

Figure 26 SeeBeyond Enterprise Designer



The procedure for invoking the Enterprise Designer is described in **Starting Enterprise Designer** on page 58.

4.1.1 User Interface

The Enterprise Designer contains the customary graphical interface features, which are described in the following sections of this chapter.

- **Menus** on page 60 describes the options contained in the individual menus.
- **Toolbar** on page 63 describes the functionality of the toolbar icons.
- **Browser Buttons** on page 63 describes the browser buttons that appear throughout the Enterprise Designer, in various wizards and dialog boxes.

The major features of the Enterprise Designer are the Enterprise Explorer on the left, and an editor panel on the right—which is initially blank. The Enterprise Explorer follows the familiar Windows Explorer format, displaying a tree structure. The Enterprise Explorer provides two views of the ICAN system, which are described in the following sections of this chapter:

- **Project Explorer** on page 64
- **Environment Explorer on page 65**

4.1.2 Editors

The editor panel displays a variety of editors, depending upon what component is selected in the Enterprise Explorer. These editors are described in the following sections of this chapter:

- **Connectivity Map Editor** on page 66
- **OTD Editor** on page 67
- **Collaboration Editor (Java)** on page 68
- **Collaboration Editor (XSLT)** on page 69
- **Environment Editor** on page 70
- **Deployment Editor** on page 71

4.1.3 Analysis and Archiving tools

The Enterprise Designer includes several analysis and archiving tools, which are described in the following sections of this chapter:

- **Project/Environment Import** on page 72, which allows you to import a Project that has been created elsewhere.
- **Project/Environment Export** on page 76, which allows you to export a Project to an external file so that it may be used elsewhere.
- **Impact Analyzer** on page 81, which helps you visualize how a change to one part of a Project would affect the rest of the Project.
- **Version Control** on page 83, which allows you to maintain multiple versions of Project components.

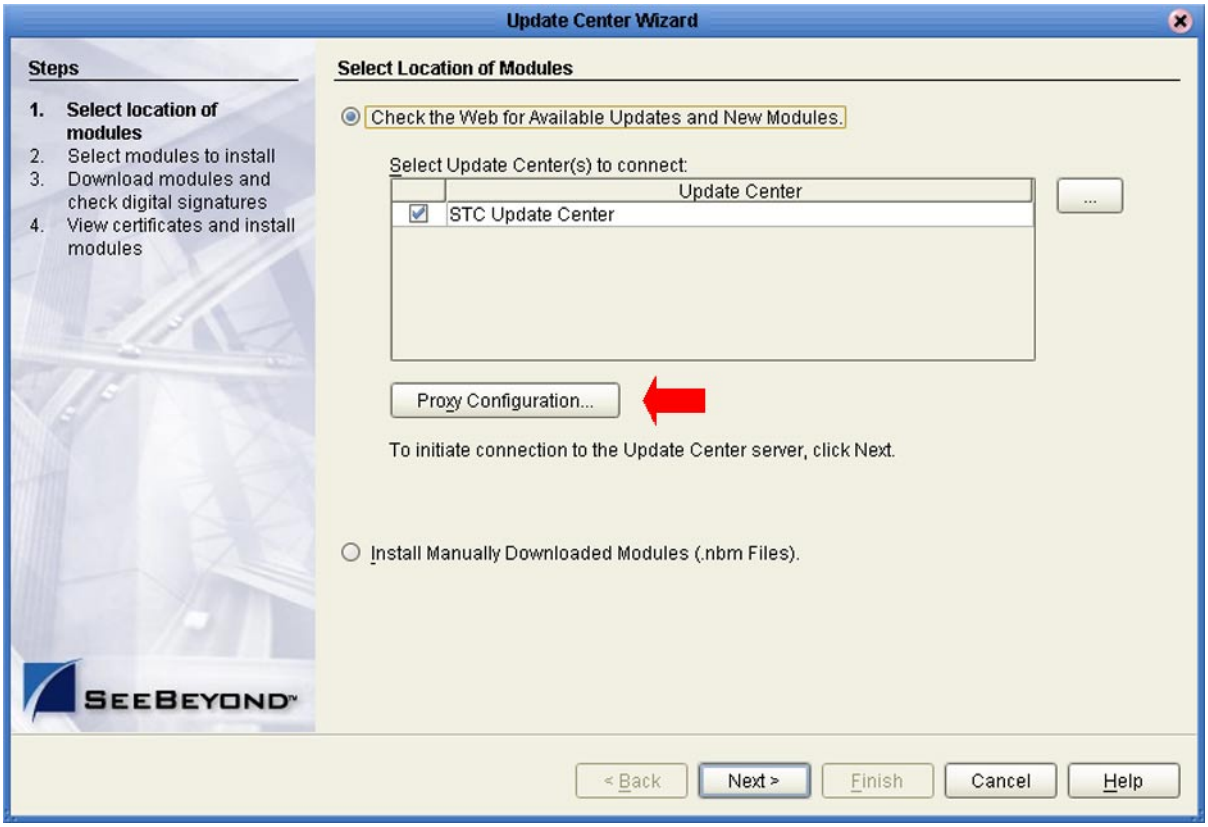
4.2 Using a Proxy Server

The following procedure allows Enterprise Designer to reference a WSDL file using a URL that points outside your firewall. See the *SeeBeyond ICAN Suite Installation Guide* for additional information.

To configure a proxy server

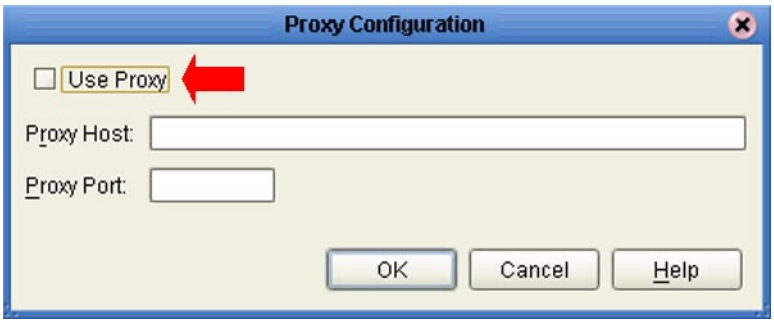
1 From Enterprise Designer's **Tools** menu, select **Update Center** to display the Update Center Wizard (see Figure 27).

Figure 27 Update Center Wizard



2 Click the **Proxy Configuration** button (see Figure 27) to display the Proxy Configuration dialog box shown in Figure 28.

Figure 28 Proxy Configuration Dialog Box



3 Check the **Use Proxy** box, and fill in the fully-qualified name of the proxy host and port for the proxy server.

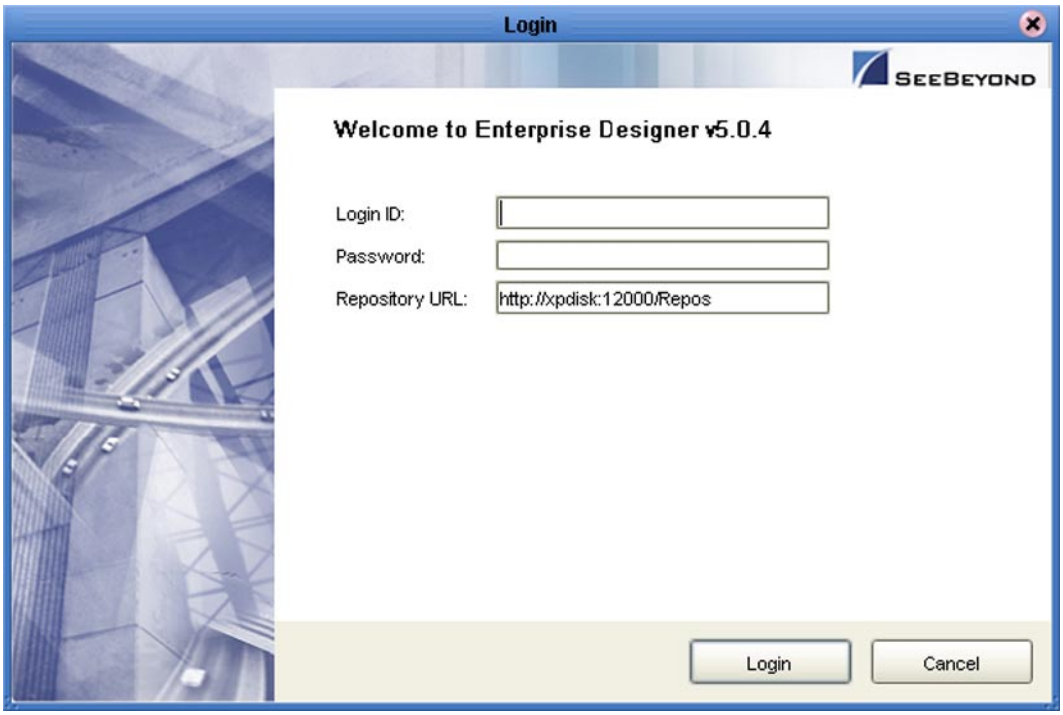
4 Click **OK** to finish.

4.3 Starting Enterprise Designer

To start the Enterprise Designer on a Windows Platform

1 Run the batch file ***ICAN-root*\edesigner\bin\runed.bat** to display the *Login* dialog box shown in Figure 29 (placing a shortcut on your desktop streamlines this procedure).

Figure 29 Login Dialog Box



2 Click in the *Login ID* text box, and enter your login ID.

3 Tab to the *Password* text box, and enter your password.

4 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 *SeeBeyond ICAN Suite Installation Guide* for details.

5 Click **Login** to complete the login process and display the Enterprise Designer GUI shown in Figure 26. A progress monitor will appear while the process is running.

To start the Enterprise Designer on a UNIX Platform

1 Run the script ***ICAN-root*/edesigner/bin/runed.sh** to display the *Login* dialog box shown in Figure 29.

2 Click in the *Login ID* text box, and enter your login ID.

3 Tab to the *Password* text box, and enter your password.

4 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 *SeeBeyond ICAN Suite Installation Guide* for details.

5 Click **Login** to complete the login process and display the Enterprise Designer GUI shown in Figure 26.

To start the Enterprise Designer on an HP NonStop (HNS) Platform

1 Open a command prompt and change directories to ***ICAN-root*/edesigner/bin**, and type the following command:

```
runed hostname port rep_name
```

where:

- ***hostname*** is the TCP/IP host name of the server where you installed the Repository—not the name of the Repository itself.
- ***port*** is the port number of the Repository.
- ***rep_name*** is the name of the Repository.

This displays the dialog box shown in Figure 29.

2 Click in the *Login ID* text box, and enter your login ID.

3 Tab to the *Password* text box, and enter your password.

4 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 *SeeBeyond ICAN Suite Installation Guide* for details.

5 Click **Login** to complete the login process and display the Enterprise Designer GUI shown in Figure 26.

4.4 Interface Features

4.4.1 Menus

The menu bar provides access to a variety of options for managing your Project. The individual menus are described in the following tables.

File Menu

Table 10 File Menu Options

Option	Function
Save	Saves changes to the selected objects (to the local workspace only).
Save All	Saves changes to all objects currently open in the editor (to the local workspace only).
Exit	Closes the Enterprise Designer.

Tools Menu

Table 11 Tools Menu Options

Option	Function
Impact Analyzer	Displays a dialog box in which you can view how one component of a Project impacts other components. See **Impact Analyzer** on page 81.
Options	Displays the Options Setup dialog box, in which you can specify selected options such as heap sizes and language extensions. See **Options Setup** on page 62.
Update Center	Displays a series of dialog boxes in which you can check for program updates. See the *eGate Integrator Installation Guide*.

View Menu

Table 12 View Menu Options

Option	Function
Environment Explorer	Activates the **Environment Explorer** tab on the Enterprise Explorer. See **Environment Explorer on page 65**.
Project Explorer	Activates the **Project Explorer** tab on the Enterprise Explorer. See **Project Explorer** on page 64.

Window Menu

Table 13 Window Menu Options

Option	Function
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 displays 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 14 Help Menu Options

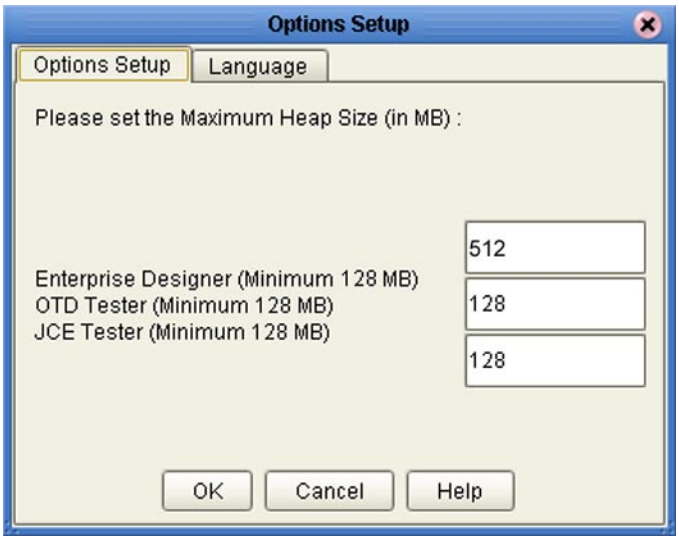
Option	Function
About Enterprise Designer	Displays an information box giving the version number, copyright information, and ICAN Repository connection information.
Contents	Displays the online help for all installed components of the ICAN Suite that operate within Enterprise Designer.
Help Sets	For future use (currently duplicates the Contents option).

Options Setup

Heap Size

The Heap Size tab allows you to increase the heap size of Enterprise Designer itself and the OTD and JCE Testers (see Figure 30). Although the default heap size settings should be adequate for most applications, occasions may arise when you will need to allocate additional memory in one or more of these modules to accommodate large file sizes.

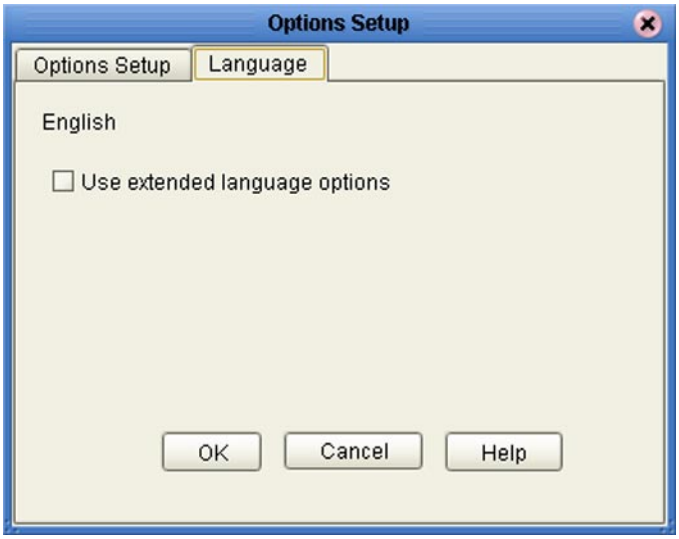
Figure 30 Options Setup - Heap Size Dialog



Language

The Language tab allows you to control the use of extended language options in Japanese and Korean localized versions of eGate Integrator. Checking the box adds data encoding features to selected OTDs and Java Collaboration Definitions.

Figure 31 Options Setup - Language Dialog



4.4.2 Toolbar

Table 15 Enterprise Designer Toolbar Icons

Icon	Function
	**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.
	Displays the **Impact Analyzer** dialog box, which allows you to view how one component of a Project impacts other components.

4.4.3 Browser Buttons

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

Table 16 Browser Buttons

Button	Function
	**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.).

4.5 Enterprise Explorer

The Enterprise Explorer organizes the components of a Project into tabs that display different views of an eGate system.

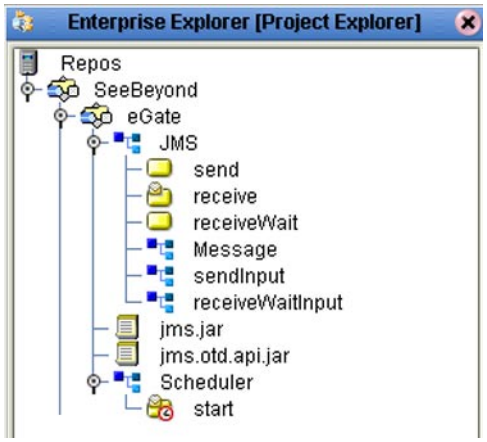
- **Project Explorer** on page 64 deals with logical components.
- **Environment Explorer on page 65** deals with physical resources, including the Logical Host and Integration Server.

Note: *The Project and Environment trees are initially loaded only to the Project or Environment level. The contents of a Project or Environment are loaded when you expand the particular node. This causes a slight delay when you expand the node, but eliminates a potentially-significant delay when you open Enterprise Designer, due to the large size of some OTD libraries.*

4.5.1 Project Explorer

The **Project Explorer** tab includes folders and icons that represent the names and contents of Projects. Some example components of a Project are shown in Figure 32.

Figure 32 Enterprise Explorer: Project Explorer View



Details of the features and usage of the Project Explorer are found in **eGate Projects** on page 92.

4.5.2 Environment Explorer

An Environment consists of Logical Hosts capable of hosting eGate components and information about external systems which may be involved with an eGate configuration.

Figure 33 Enterprise Explorer: Environment Explorer View

Details of the features and usage of the Environment Explorer are found in **Environments** on page 290.

4.6 Enterprise Designer Editors

The editor panel—which is initially blank—displays a variety of editors, depending upon what component is selected in the Enterprise Explorer. Additional facilities are also displayed here, such as the Java Debugger (see **Debugging Java-based Collaboration Definitions** on page 227).

Note: *See the eGate Integrator Tutorial for an end-to-end demonstration of the steps involved in setting up a Project.*

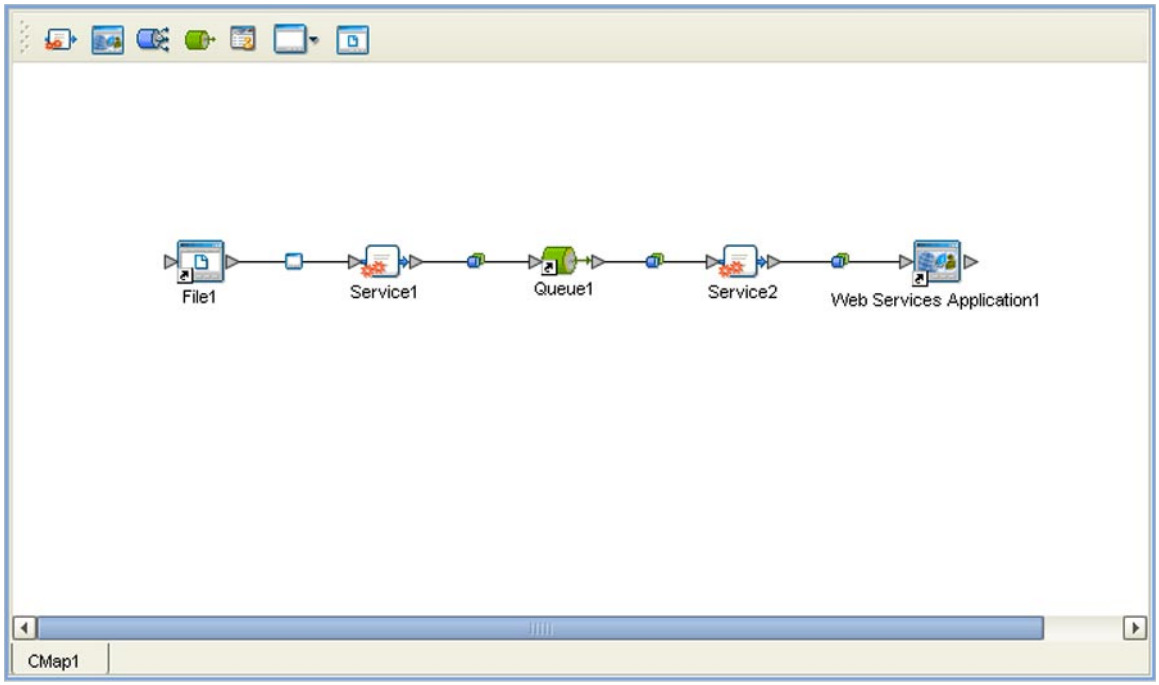
4.6.1 Connectivity Map Editor

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, shown in Figure 34, 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 dialog boxes that are displayed by clicking on the component icons.

Note: *You should create your Collaboration Definitions before using the Connectivity Map to connect components.*

See **Using the Connectivity Map Editor** on page 104 for detailed information.

Figure 34 Connectivity Map Editor

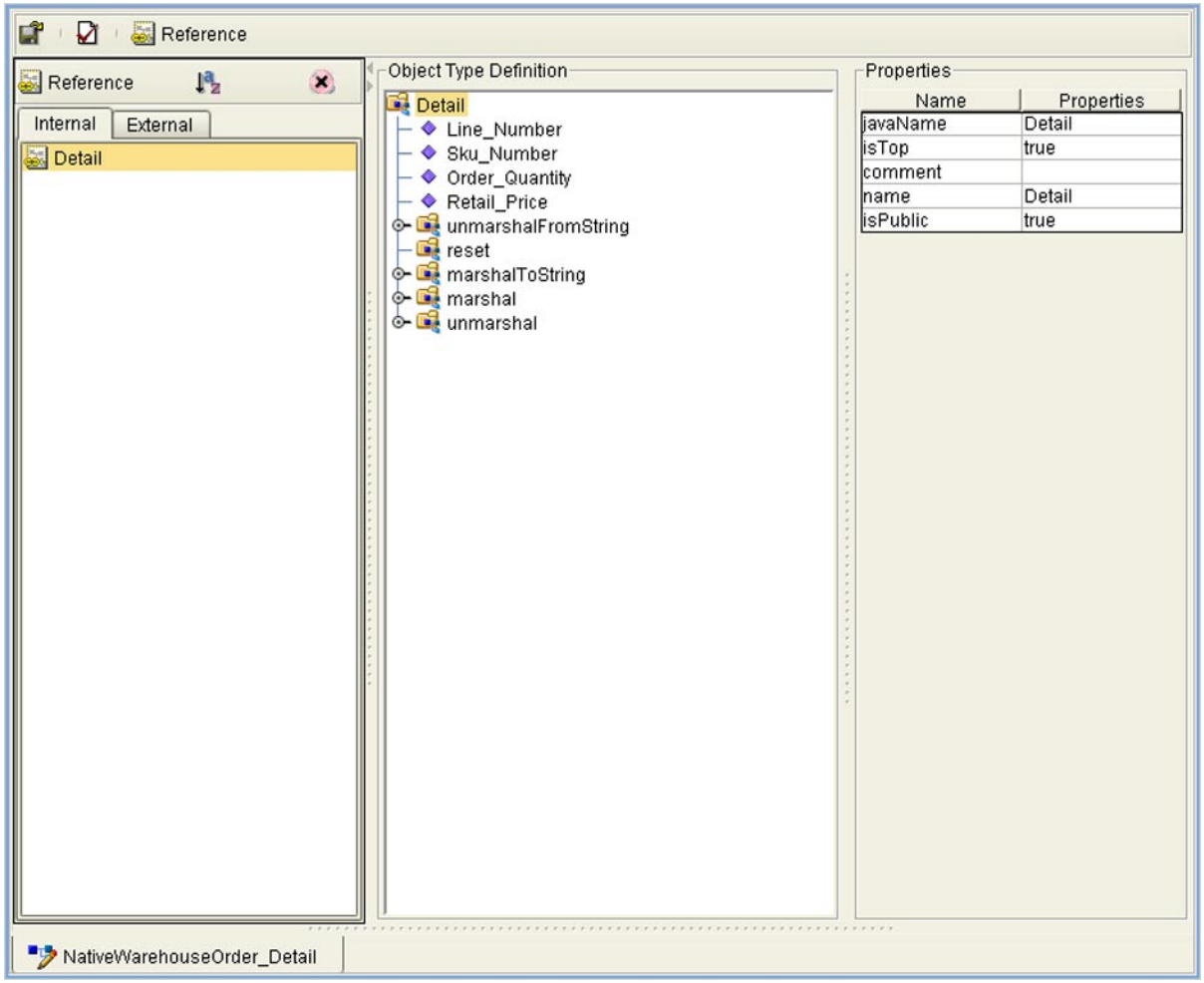


4.6.2 OTD Editor

The OTD Editor window, as shown in Figure 35, displays the source files used to create the Object Type Definitions (OTDs) to use with a Project. You use an OTD wizard tool to create OTD files and add them to the **Project Explorer** tab.

See **OTD Editor** on page 118 for detailed information.

Figure 35 OTD Editor

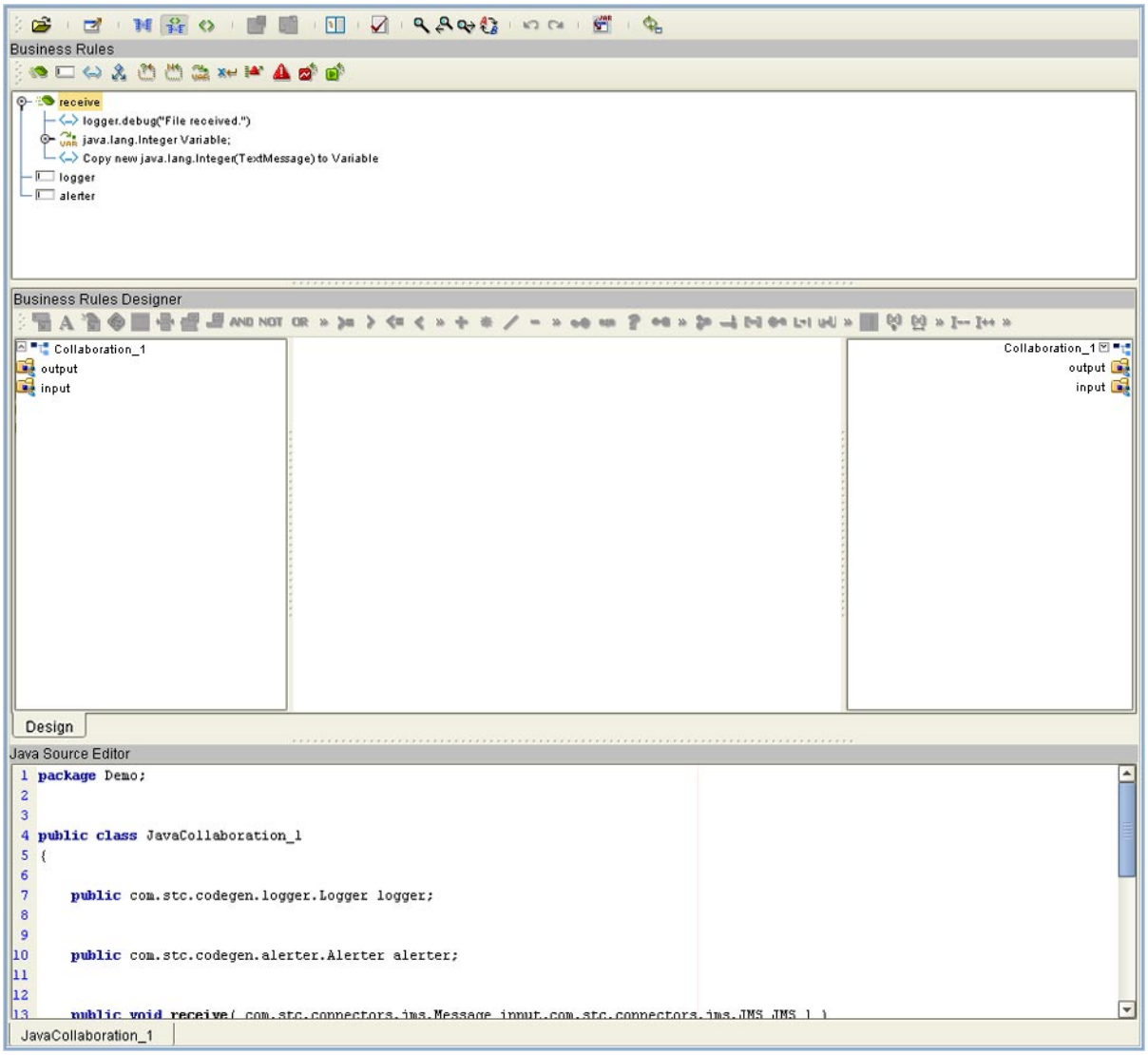


4.6.3 Collaboration Editor (Java)

The Collaboration Editor (Java) window, as shown in Figure 36, displays a Java-based Collaboration Definition that you want to include in a Project. You use a Java wizard tool to create Collaboration Definition files and add them to the **Project Explorer** tab.

See **Using the Collaboration Editor (Java)** on page 179 for detailed information

Figure 36 Collaboration Editor (Java)

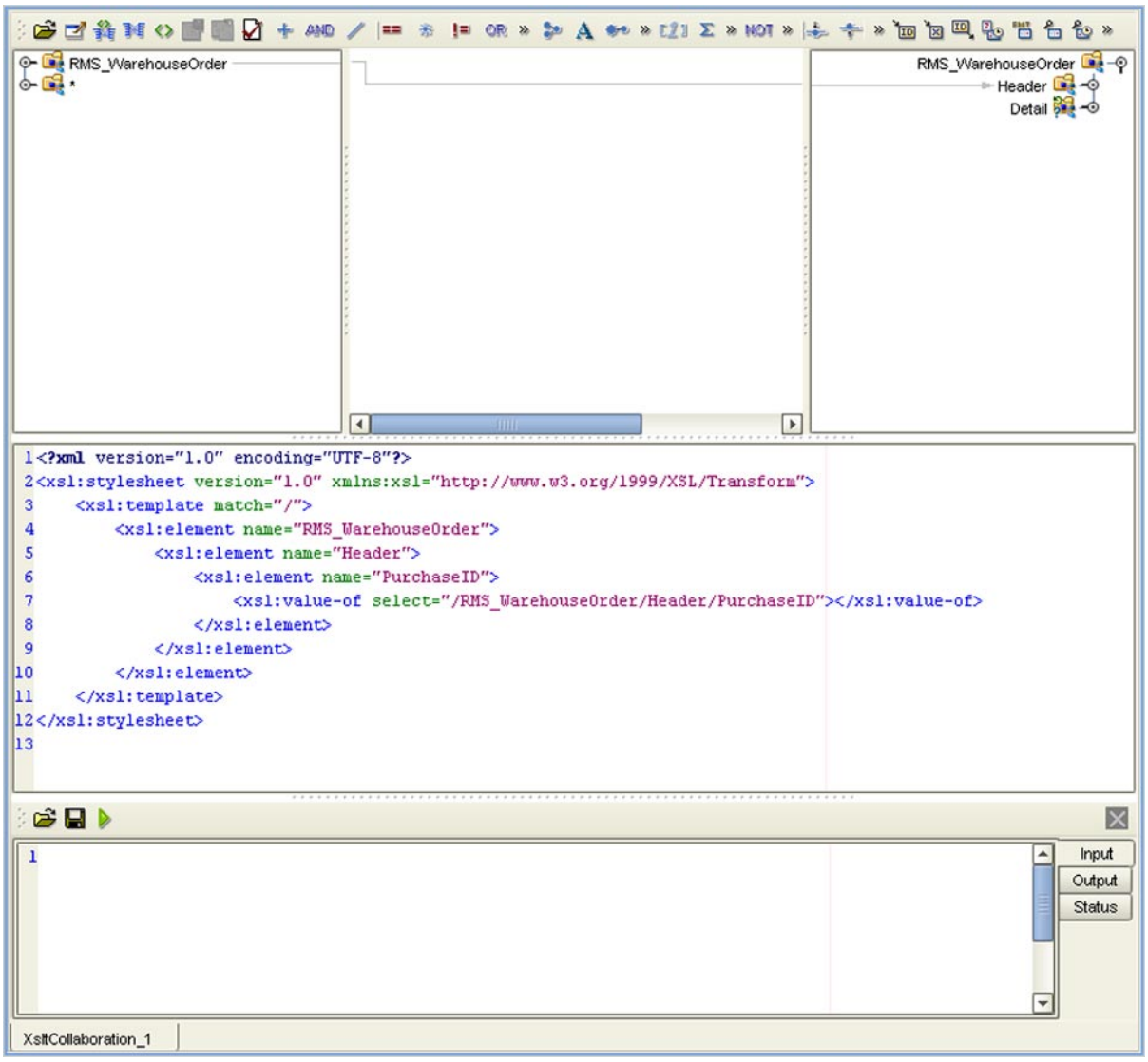


4.6.4 Collaboration Editor (XSLT)

The Collaboration Editor (XSLT) window, as shown in Figure 37, displays the XSLT-based Collaboration Definitions that you need to map together and include in the Project. You use a XSLT wizard tool to create Collaboration Definition files and add them to the **Project Explorer** tab.

See **Using the Collaboration Editor (XSLT)** on page 266 for detailed information

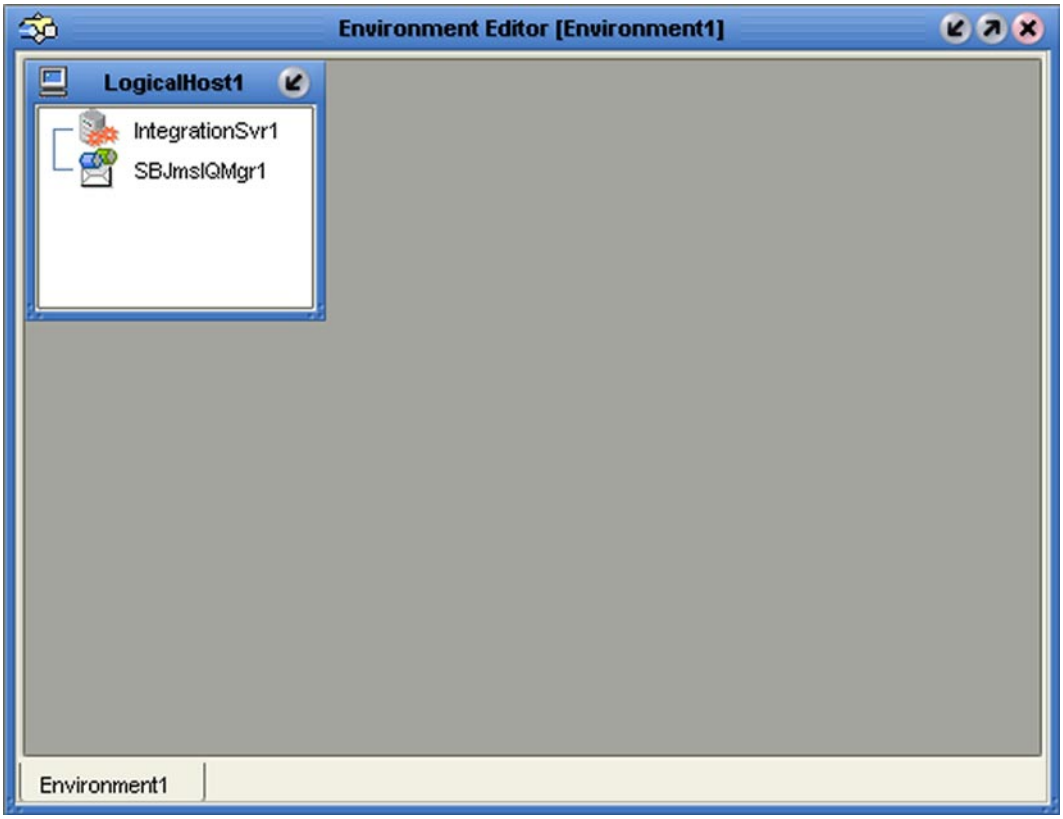
Figure 37 Collaboration Editor (XSLT)



4.6.5 Environment Editor

The Environment Editor provides a canvas in which you can create and customize an Environment. Here you can see the various components (Logical Hosts, servers, and external systems) included in the selected Environment. An environment containing example Logical Hosts is shown in Figure 38.

Figure 38 Environment Editor

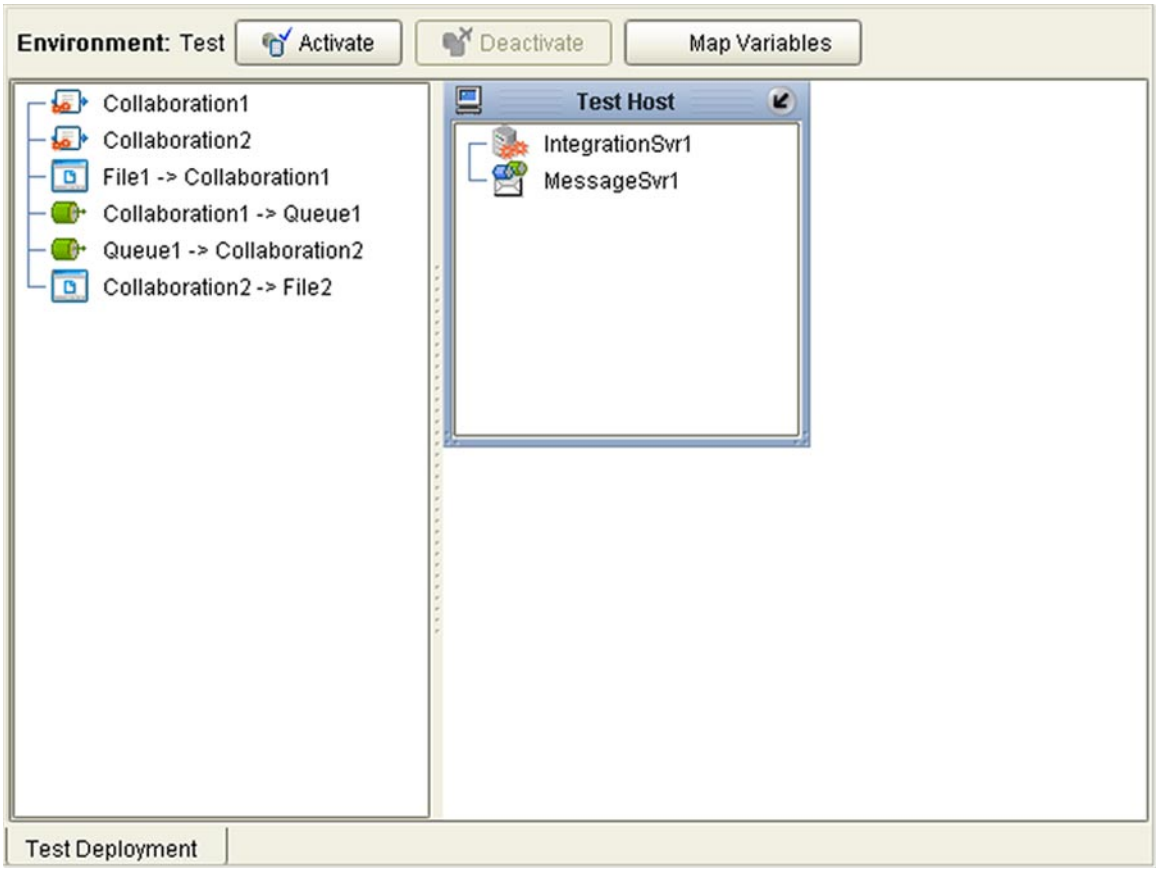


Note: *Unlike changes to Project-related configuration properties, changes to Environment-related properties do not require redeployment, only application.*

4.6.6 Deployment Editor

The Deployment Editor, as shown in Figure 39, contains information about how Project components will be deployed in an Environment. See **The Deployment Editor** on page 333 for detailed information

Figure 39 Deployment Editor



4.7 Additional Tools and Features

4.7.1 Project/Environment Import

The import function allows you to import an eGate Project or Environment file using the Enterprise Designer. Both follow essentially the same procedure.

Important: *Products installed in the source Repository must be installed in the Repository into which the Project is imported.*

When importing a Project, note that:

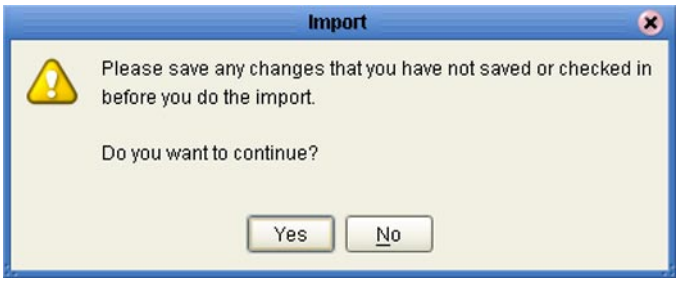
- Existing Projects are not affected by the imported Project.
- During import, if another Project having the same name exists in the target Repository, you will receive an error message and the existing file will not be overwritten.
- If you have not installed all of the necessary products (such as eWays) that a Project requires, you will not be able to import that Project and will get an error message.
- References are validated during import.
- Project deployment objects are not imported, because they have references to both Project and Environment elements that are not required at the Project level.

Importing a Project Using Enterprise Designer

To import a Project using Enterprise Designer

1. From the Repository context menu (for Projects) or the Project context menu (for Sub-Projects), select **Import.**
2. The message box shown in Figure 40 appears, prompting you to save your changes.

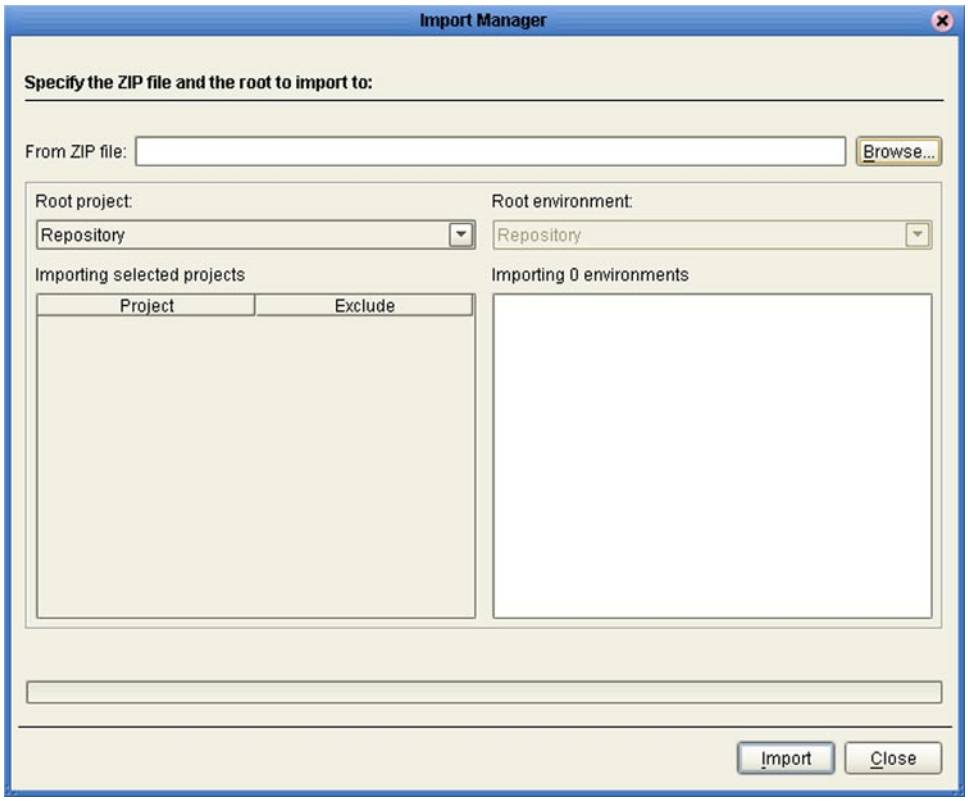
Figure 40 Import Message Box



A. If you want to save your changes, but have not already done so, click **No**. Save your changes, and then re-select **Import**, as in step 1.

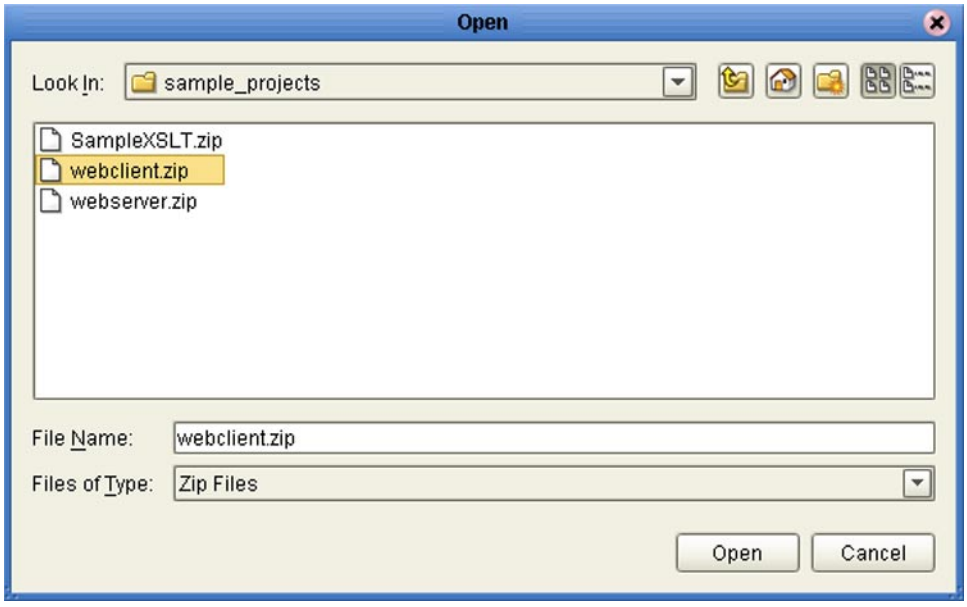
B. If you have saved any desired changes, click **Yes** to display the dialog box shown in Figure 41.

Figure 41 Import Manager Dialog Box (1)



3 Click the **Browse** button to display the *Open File* dialog box, as shown in Figure 42. If you browse to an Environment file, the *Root environment* field will be enabled.

Figure 42 Open File Dialog Box

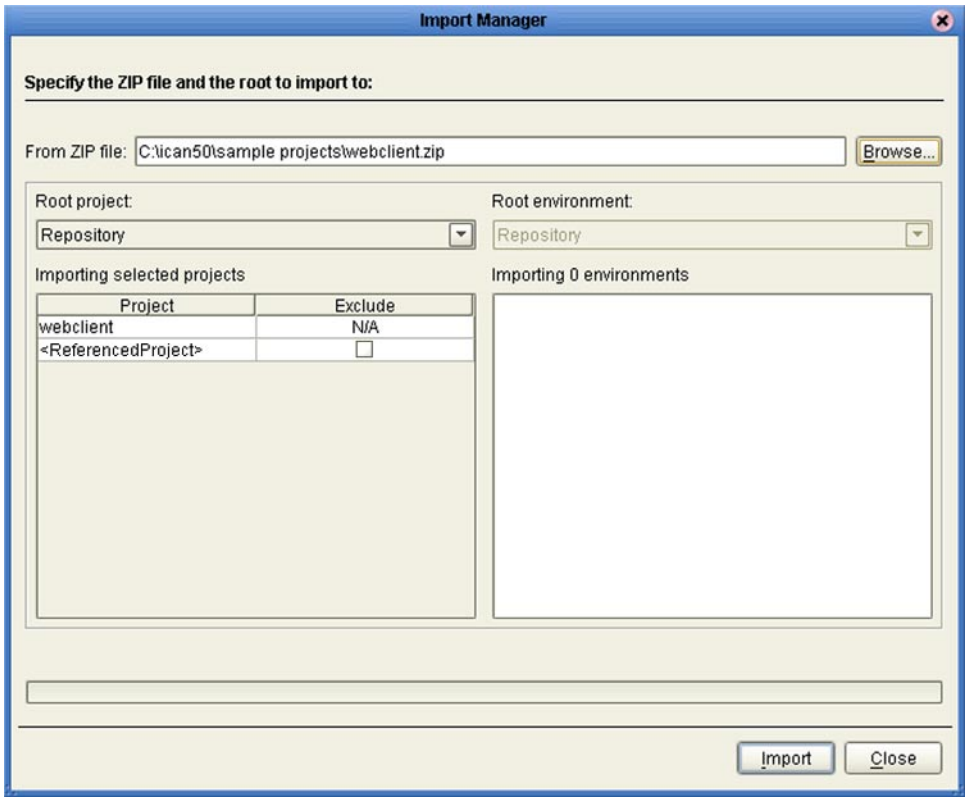


4 Locate and select the Project or Environment file that you want to import.

5 Click **Open** to import the file.

6 The Import Manager dialog box appears as shown in Figure 43.

Figure 43 Import Manager Dialog Box



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.

8 The Import Status message box shown in Figure 44 appears after the file has been imported successfully.

Figure 44 Import Status Message Box

9 Click **OK** to close the message box.

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

Importing a Project Using the Command Line

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

Location of script file:

ICAN-root\repository\util\importProject.bat (or importProject.sh)

Command Syntax:

importProject ***username password importfile rootproject***

where:

- ***importfile*** is the name and path of the archive file containing the Project or Environment you are importing.
- ***rootproject*** is the name of an existing Project, under which the imported Project will become a sub-Project. If the imported Project is not to become a sub-Project, then leave this parameter as an empty string (“”).

To import a Project using the import script

1 Open a command prompt and change directory to ***ICAN-root***\repository\util.

2 To extract a Project contained in the file **c:\project4import.zip** and import it into the Repository, type:

```
importProject username password c:\project4import.zip ""
```

To import a Project as a sub-Project using the import script

1 Open a command prompt and change directory to ***ICAN-root***\repository\util.

2 To extract a Project contained in the file **c:\project4import.zip** and import it into the Repository as a sub-Project of **mainProject**, type:

```
importProject username password c:\project4import.zip mainProject
```

To import an Environment using the import script

1 Open a command prompt and change directory to ***ICAN-root***\repository\util.

2 To extract an Environment contained in the file **c:\environment4import.zip** and import it into the Repository, type:

```
importProject username password c:\environment4import.zip ""
```

3 The Environment, including all Projects deployed to that Environment, will be imported.

4.7.2 Project/Environment Export

The export function allows you to export an eGate Project and/or Environment to an external file using either the Enterprise Designer or a command-line script.

When exporting a Project, 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, because they have references to both Project and Environment elements that are not required at the Project level.

Exporting a Project Using Enterprise Designer

To export a Project or Environment using Enterprise Designer

1 From the Project context menu, select **Export** to display the Export Manager dialog box. If you do not have any existing Environments in your Repository, you will see the dialog box shown in Figure 45. If you do, you will see the dialog box shown in Figure 46.

Figure 45 Export Manager Dialog Box (1a)

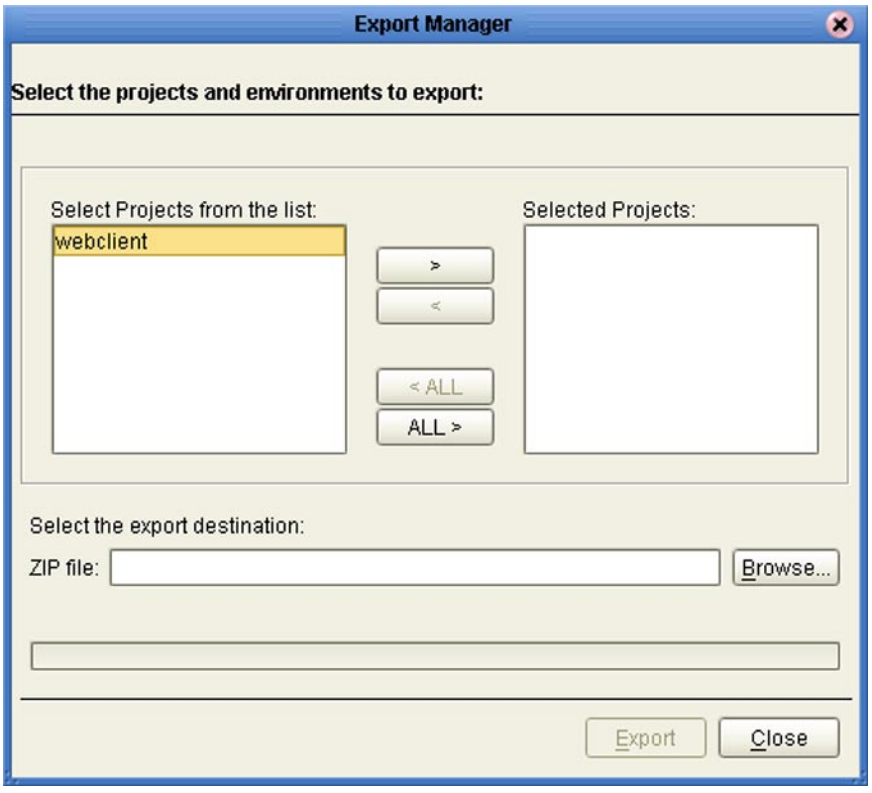
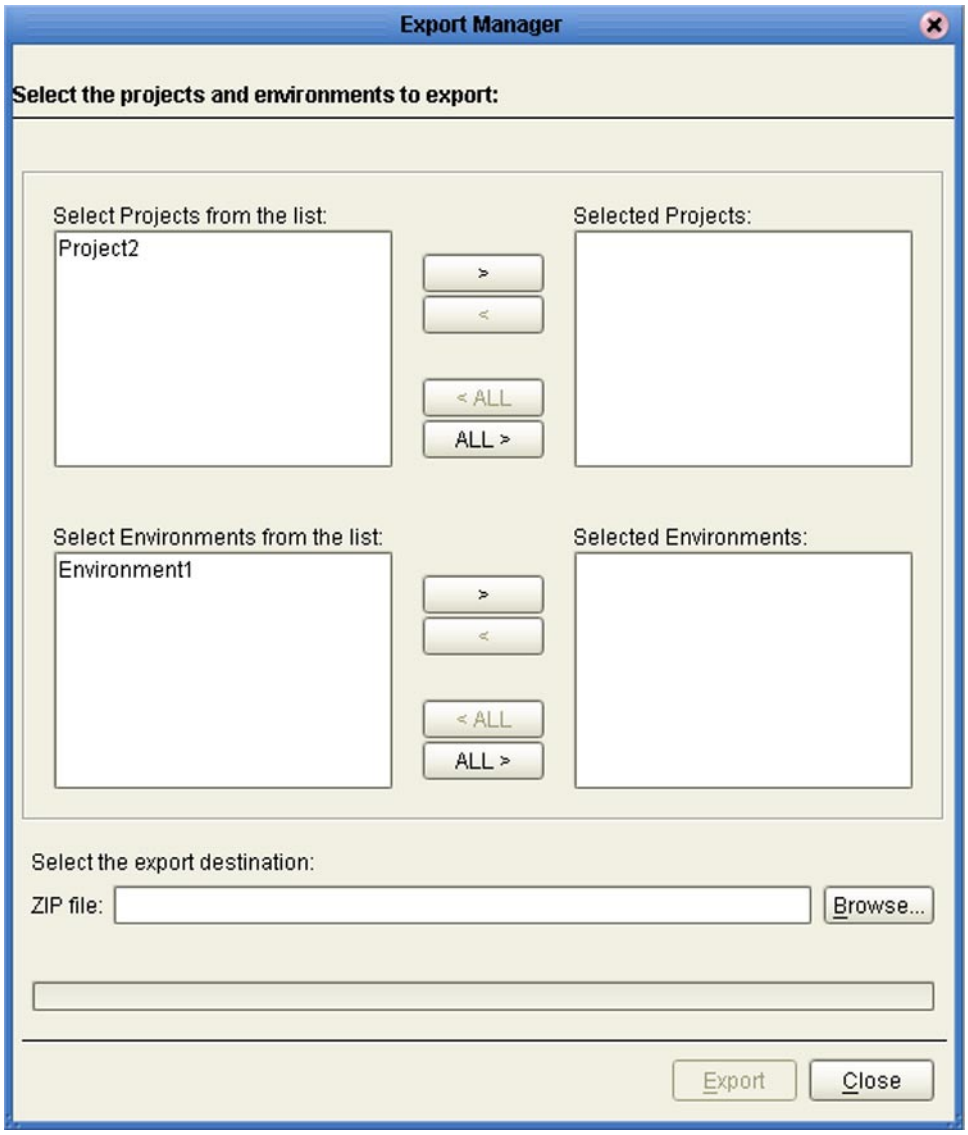
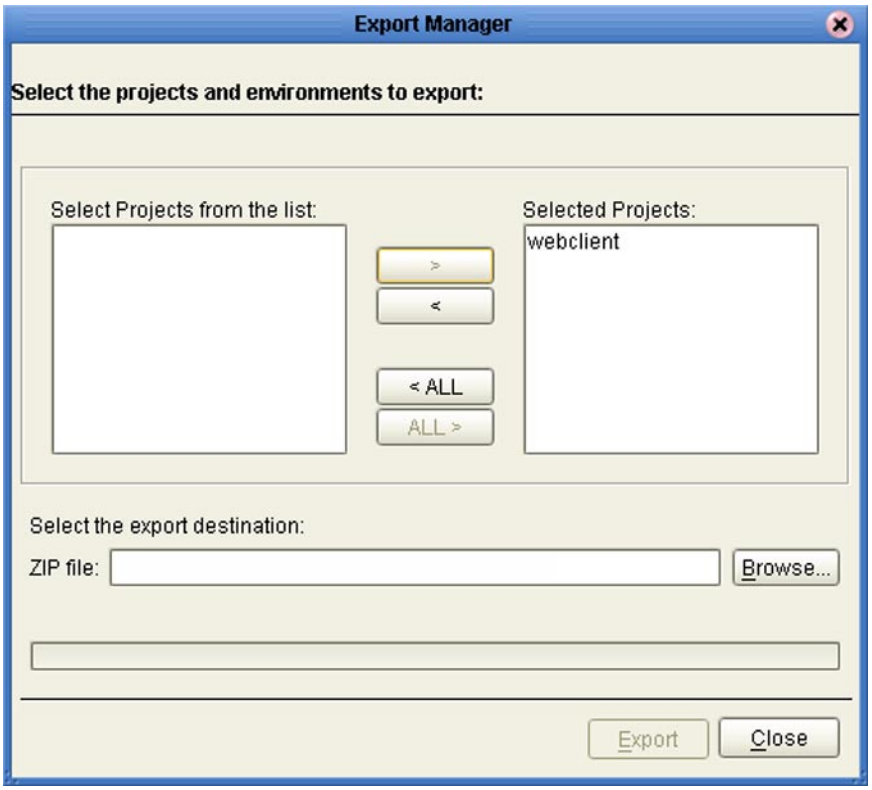


Figure 46 Export Manager Dialog Box (1b)



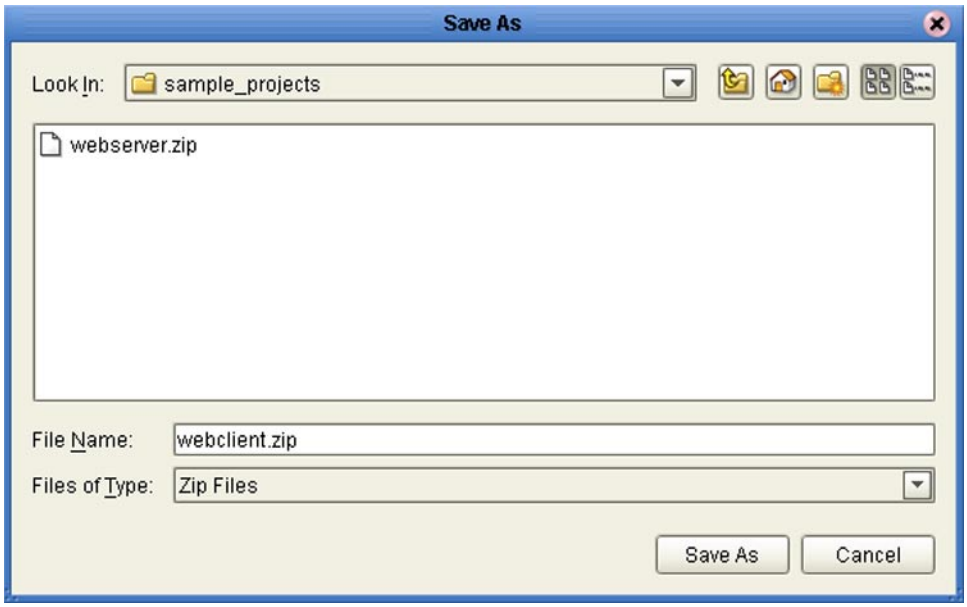
2 Highlight the desired Project(s) or Environment(s) in the displayed list, and transfer them to the *Selected Projects* or *Selected Environments* panel using the arrow buttons (see Figure 47).

Figure 47 Export Manager Dialog Box (2)



3 Click the **Browse** button to display the *Save As* dialog box, as shown in Figure 48.

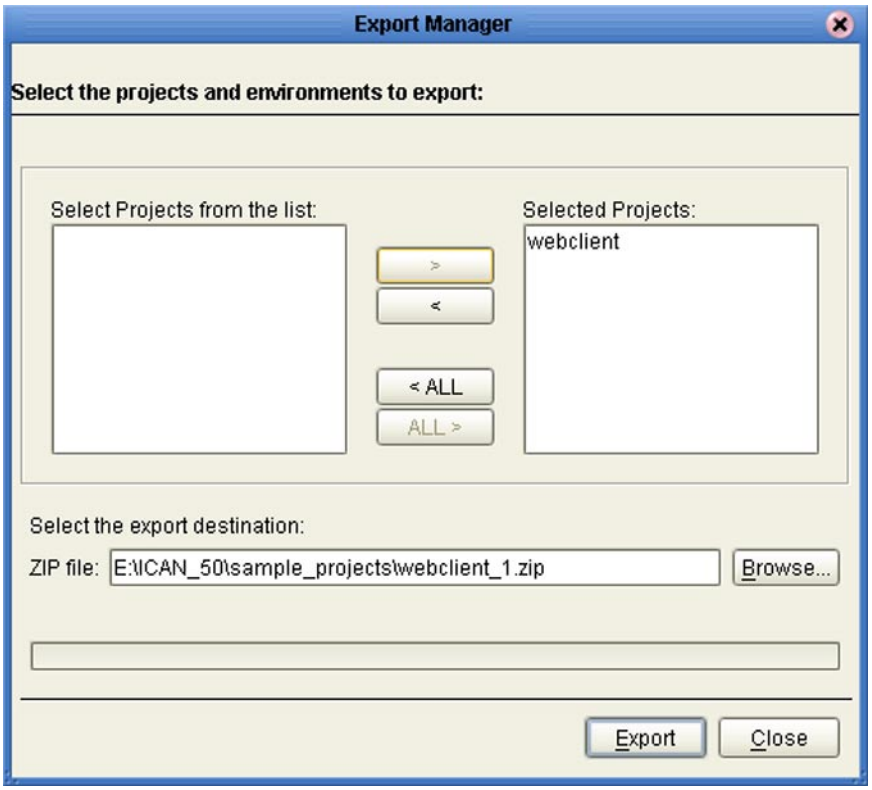
Figure 48 Save As Dialog Box



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

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

Figure 49 Enter File Name Dialog Box (2)



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

7 The Export Status message box shown in Figure 50 appears after the file has been exported successfully.

Figure 50 Export Status Message Box

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

Exporting a Project Using the Command Line

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

Location of script file:

ICAN-root\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 ("")

To export a Project using the export script

1. Open a command prompt and change directory to ***ICAN-root***\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 ***ICAN-root***\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 ***ICAN-root***\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.
```

4.7.3 Impact Analyzer

The Impact Analyzer helps you determine how a change to one component of a Project or Environment will affect other components in that Project or Environment.

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 box shown in Figure 51.
3. In the *Please show me* drop-down list, select items you would like to view. You have the option of viewing either:
   - Objects that have references to the selected object.
   - Objects that are referenced by the selected object.
4. You can filter the number of listed objects using the *Please show me impacted objects in* drop-down list; the default is the entire **Repository**.
5. You can print the object list by clicking **Print** to display the Windows *Print* dialog box.

Figure 51 Impact Analyzer Dialog Box

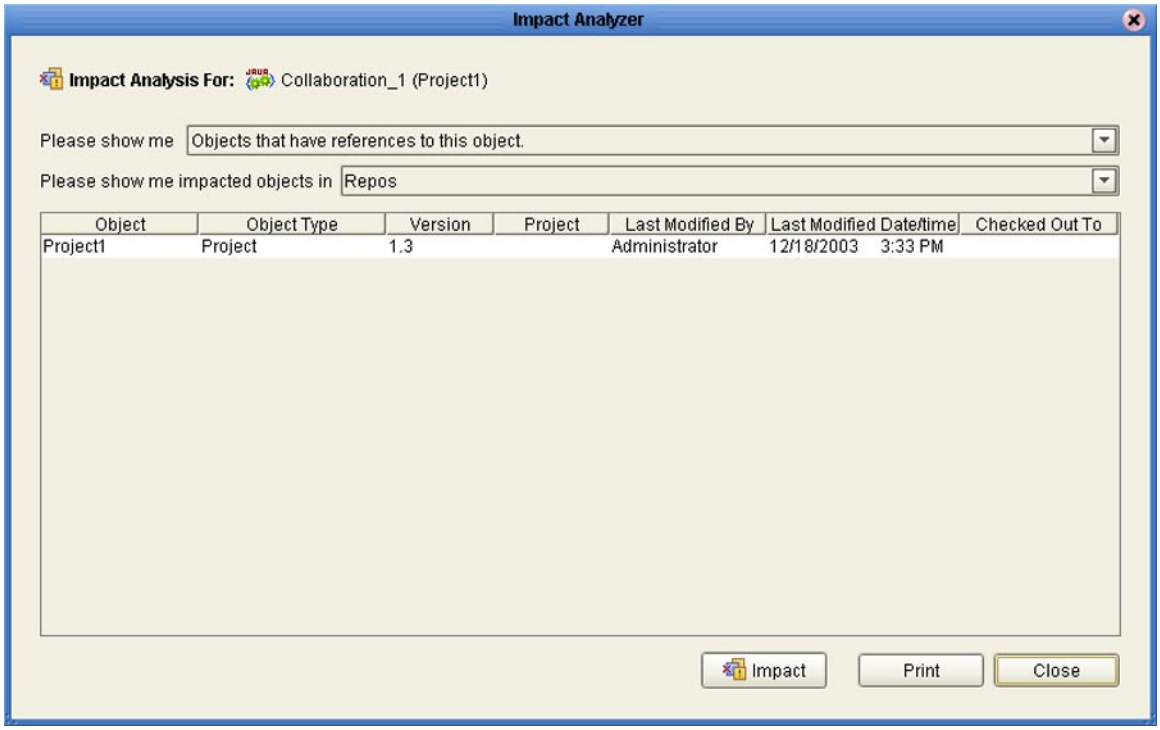
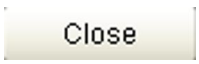


Table 17 Impact Analyzer Command Buttons

Button	Function
	Performs an impact analysis for the object selected from the object list (not fully implemented).
	Displays the Windows **Print** dialog box, which you can use to print the object list.
	Closes the **Impact Analyzer** dialog box.

4.7.4 Version Control

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

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

Checked-In State

When a component is checked in to the version control system, it is locked against modification until being checked out, and a lock is displayed in the component's icon in the Enterprise Explorer. Figure 52 shows the lock, using the OTD icon as an example. See **Checking a Component In** on page 85, **Checking a Component In Without Revisions** on page 86, and **Checking In a Previous Version as the Latest Version** on page 90.

Figure 52 *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 53) 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 87.

Figure 53 *Checked Out* Icon

Retrieved State

When any version of a component is retrieved from the version history dialog box, it is *not* locked against another user checking it out or retrieving it. A combined writing pad/warning icon (see Figure 54) 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 88 and **Checking In a Previous Version as the Latest Version** on page 90.

Figure 54 *Retrieved* Icon

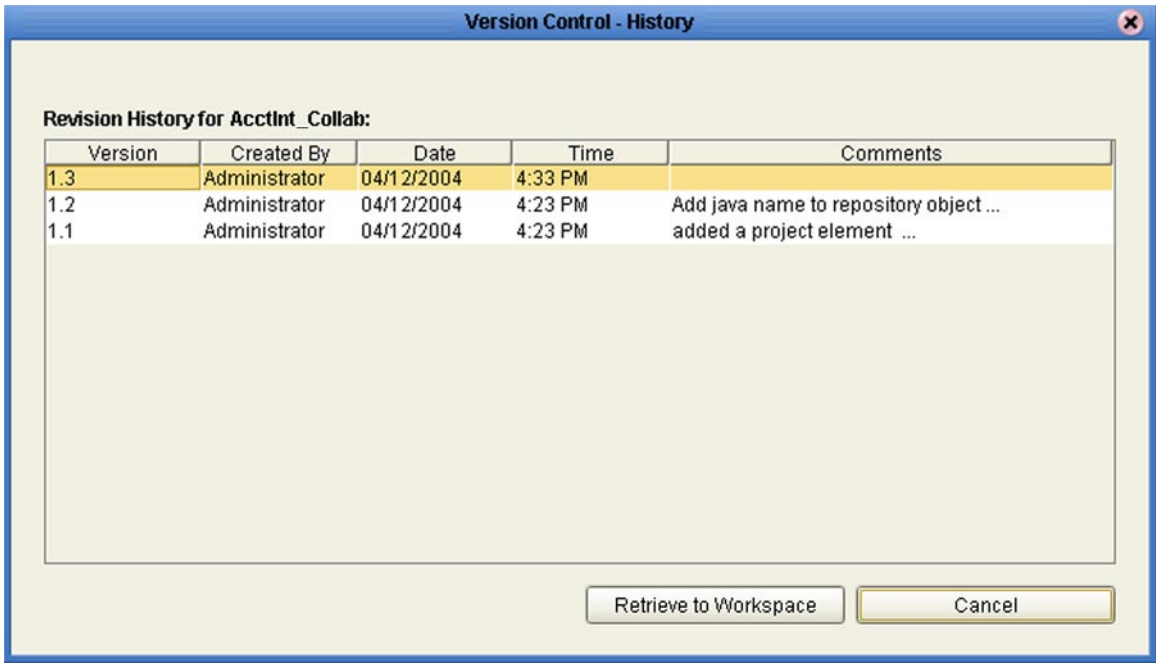
Viewing a Component's Version History

To view the version history for a component

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

2 Select **Version History** to display the *Version Control - History* information box shown in Figure 55.

Figure 55 Version Control - History Dialog Box



3 Double-click in 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.*

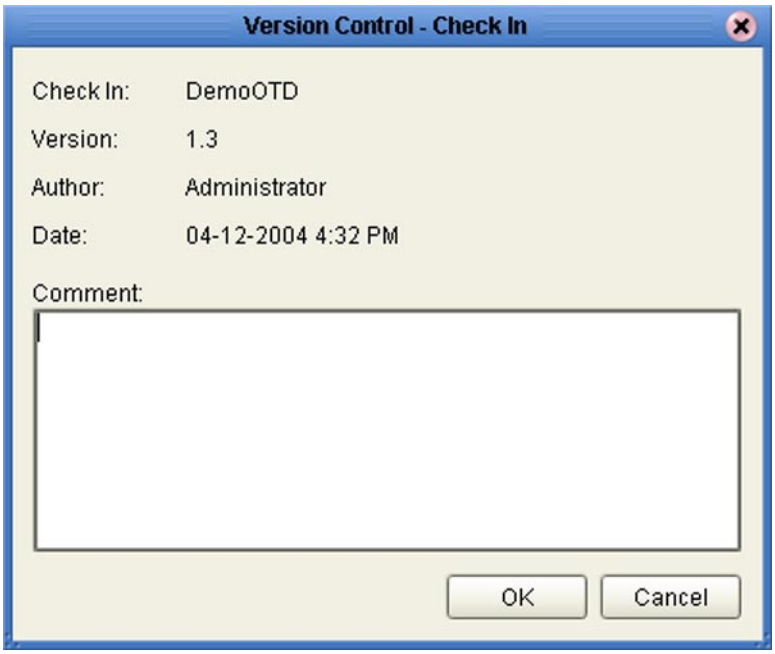
Checking a Component In

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.

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

1. In the Enterprise Explorer, select the component and right-click to display its context menu.
2. Select **Check In** to display the *Version Control - Check In* dialog box shown in Figure 56.

Figure 56 Version Control - Check In Dialog Box



3. Type in a description of the changes in the new version.
4. Click **OK** to check the new version in.

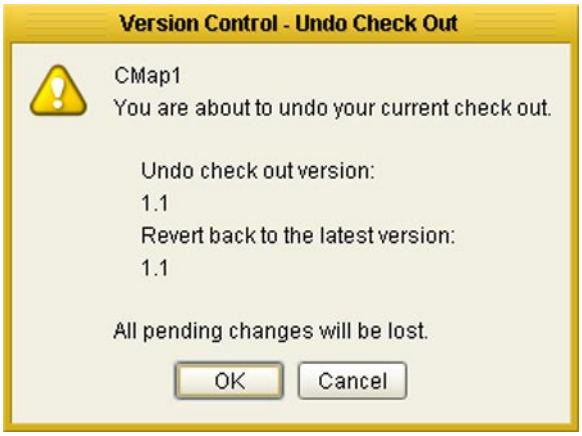
Checking a Component In Without Revisions

When you have checked the latest version of a component out and want to check it back in without any revisions, you can simply cancel the check-out by using the following procedure. The version number is not incremented.

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.
2. Select **Undo Check Out** to display the *Version Control - Undo Check Out* dialog box shown in Figure 57. (This option is not available for all components.)

Figure 57 Version Control - Undo Check Out Dialog Box



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

Note: *This procedure is also valid for retrieved versions.*

Checking a Component Out

You can check out the latest version of a component for editing by using the following procedure.

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.*

To check out the latest version of a Project or Environment component

1. In the Enterprise Explorer, select the component and right-click to display its context menu.
2. Select **Check Out** to display the *Version Control - Check Out* dialog box shown in Figure 58.

Figure 58 Version Control - Check Out Dialog Box

3. Click **OK** to check the component out.
4. Click the **Save** or **Save All** icon to place the version in your Repository workspace.

Note: *Checking out a Java Collaboration disables the cut-and-paste feature for other users.*

Retrieving a Component to Your Workspace

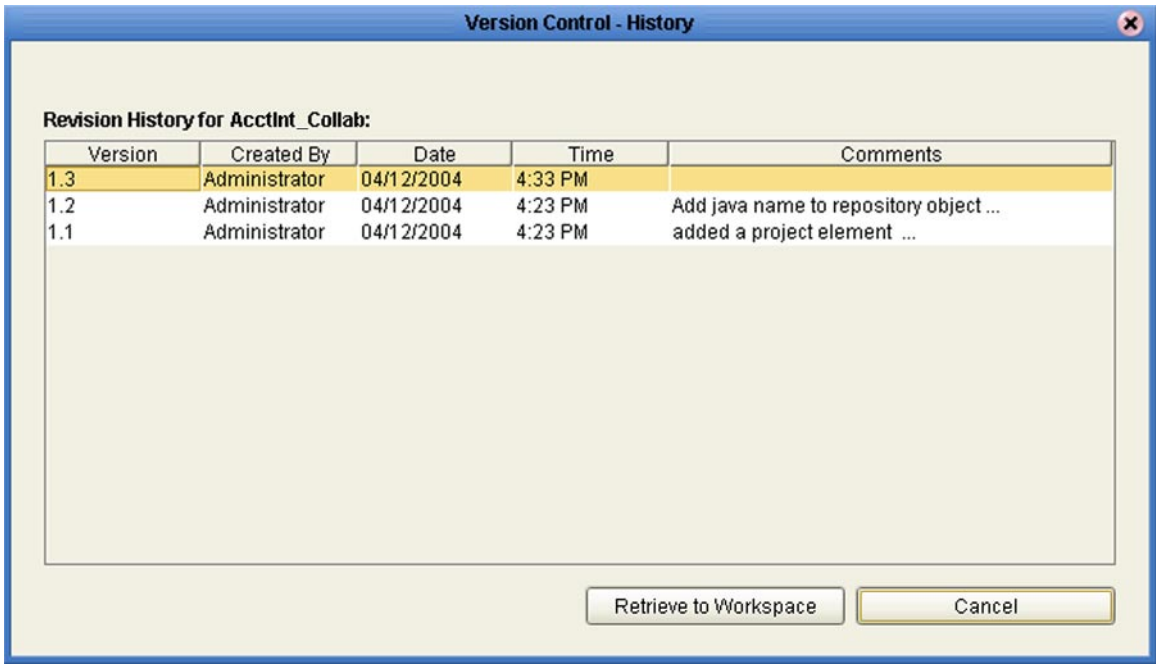
You can retrieve either the current or a previous version of a component by retrieving it from the Version History information box. Retrieving does *not* lock the file from being checked out or retrieved by other users for editing. To check a retrieved version back in as the latest version, you must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

Note: *Currently applies only to Java Collaborations and eInsight Business Processes.*

To retrieve an older version of a Project or Environment component

1. In the Enterprise Explorer, select the component and right-click to display its context menu.
2. Select **Version History** to display the *Version Control - History* dialog box shown in Figure 55.

Figure 59 Version Control - History Dialog Box



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 box shown in Figure 60.

Figure 60 Access File Dialog Box

- **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 87. To check the latest version out from the dialog box, you must select this option and click **OK**. You will then be presented with the dialog box shown previously (Figure 58).
- **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 box; simply click **OK**.

B If you are attempting to retrieve a *previous* version of the component, you will be presented with the dialog box shown in Figure 61. 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 61 Confirm Version Replace Dialog Box

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 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.

Note: *Currently applies only to Java Collaborations and eInsight Business Processes.*

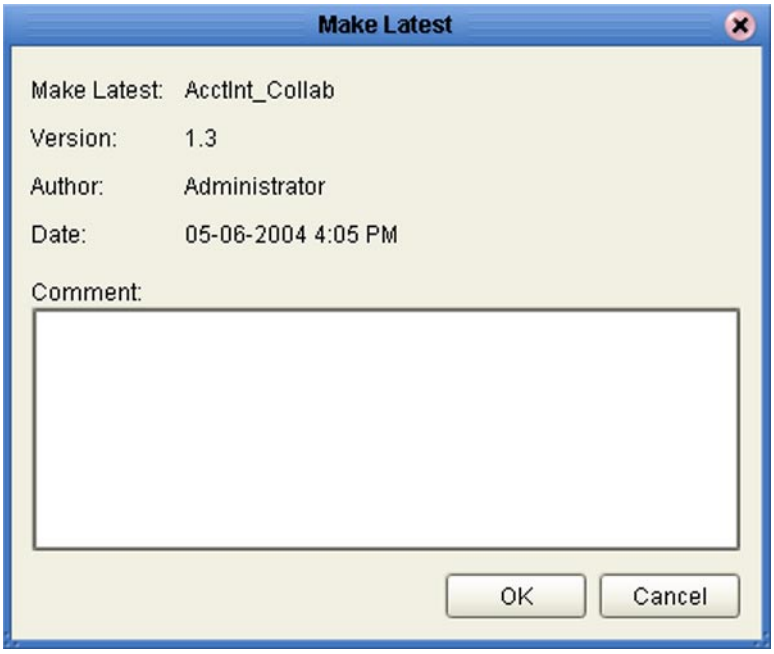
To make a previous version of a component become the latest version

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

2 Click **Make Latest** to display a confirmation dialog box.

A If the latest checked-in version of the component has not changed since you retrieved the previous version, you will see the dialog box shown in Figure 61.

Figure 62 Make Latest Dialog Box



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 box shown in Figure 63.

Figure 63 Confirm Latest Version Override Dialog Box

If you are sure you want to replace the current latest version, click **OK** to display the dialog box shown previously (Figure 62). 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.*

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 *eGate Integrator System Administration Guide* for information.

Chapter 5

eGate Projects

This chapter describes components of an eGate Project, and the use of the Enterprise Designer in defining your Project.

5.1 Overview

An eGate Project represents the logical system designed to solve either all or part of a business problem. Projects are created using tools contained within the Enterprise Designer, and are deployed to specific Logical Hosts in specific Environments by means of Deployment Profiles (see **Environments** on page 290). Components developed for use in one Project can be used in another, and a Project can internally reference another Project.

Note: *See the eGate Integrator Tutorial for an end-to-end demonstration of the steps involved in setting up a Project.*

5.1.1 Project Components

The components found in a typical Project are described in the following sections of this chapter:

- **Services** on page 106
- **External Applications** on page 107
- **Component Connections** on page 109
- **Message Destinations** on page 107

Behind the scenes, and not explicitly shown in a Connectivity Map, are other Project components such as:

- **Collaboration Definitions**

  A Collaboration Definition defines the logical operation taking place in the related Collaboration. It is created in either the Java Collaboration Editor or the XSLT Collaboration Editor, and is based on an Object Type Definition. See **Collaboration Definitions (Java)** on page 168 and **Collaboration Definitions (XSLT)** on page 259.

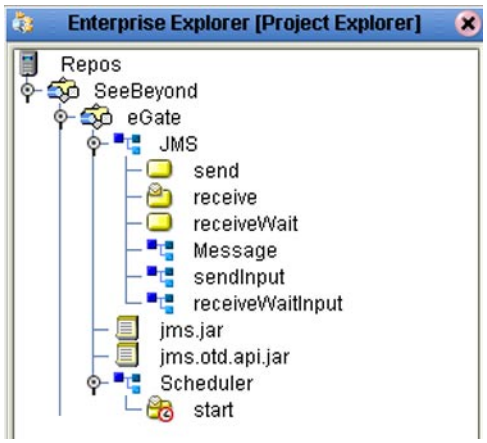
- **Object Type Definitions**

  Object Type Definitions (OTDs) are sets of rules that define the encoding of an object. They describe messages that are propagated through eGate, and the methods available for operating on them, and also interactions with external APIs. See **Object Type Definitions** on page 114.

5.2 The Project Explorer

A Project consists of logical constructs and configurations designed to solve some or all of a business problem. The **Project Explorer** displays the contents of the Repository that belong to the selected Project (see Figure 64).

Figure 64 Project Explorer



The Project Explorer is used in conjunction with the various editors to create and configure the components of a Project. Each component in the Project Explorer has an icon to identify the component type (see **Project Explorer Icons** on page 94). Right-clicking on a component displays a context menu for that component (see **Context Menus** on page 95), from which you can select appropriate actions.

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.*

5.2.1 Project Explorer Icons

The icons described in Table 18 appear in the Project Explorer.

Table 18 Project Icons

Icon	Description
	Represents the **Repository**, which is the central ICAN database where all Project information is saved. Binary files required at run time are also stored here.
	Represents the **Project** or subproject.
	Represents a **Connectivity Map**, which contains the business logic and information about the data transmission. A lock displayed in the lower-left corner indicates that the Connectivity Map is currently checked into the version control system (see OTD example below).
	Represents a **Project variable** or **constant**.
 	Represents an **Object Type Definition** (OTD) file. A lock displayed in the lower-left corner indicates that the OTD is currently checked into the version control system.
	Represents a **Collaboration Definition (Java)** file. A lock displayed in the lower-left corner indicates that the Collaboration Definition is currently checked into the version control system (see OTD example above).
	Represents an **Collaboration Definition (XSLT)** file. A lock displayed in the lower-left corner indicates that the Collaboration Definition is currently checked into the version control system (see OTD example above).
	Represents a **Deployment Profile**, which specifies how Project components are deployed to a run-time Environment.
	Displayed along side one of the above icons, indicates that the current latest version of the component has been checked out for editing.
	Displayed along side one of the above icons, indicates that some version of the component has been retrieved to the local workspace.

5.2.2 Context Menus

Right-clicking a component in the Project Explorer displays a context menu for that component. Only those menu options that are allowed for the component in its current state are activated.

Repository Menu

Figure 65 Repository Menu

Table 19 Repository Menu Options

Option	Function
Project	Adds a new Project to the Repository.
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.
Import	Displays a dialog box with which you can import a Project or Environment into the Repository.
Export	Displays a dialog box with which you can export a Project or Environment from the Repository to an archive file.
Refresh All from Repository	Refreshes the Project Explorer to display the current contents of the Repository. (Open editors are not refreshed.)
User Management	Displays the User Management dialog box, where an Administrator can manage user access to the Repository with options for adding, modifying, and deleting users. See the *eGate Integrator System Administration Guide*.
Properties	Displays a dialog box showing the configuration properties of your Repository. See the *eGate Integrator System Administration Guide*.

Project Menu

Figure 66 Project Menu

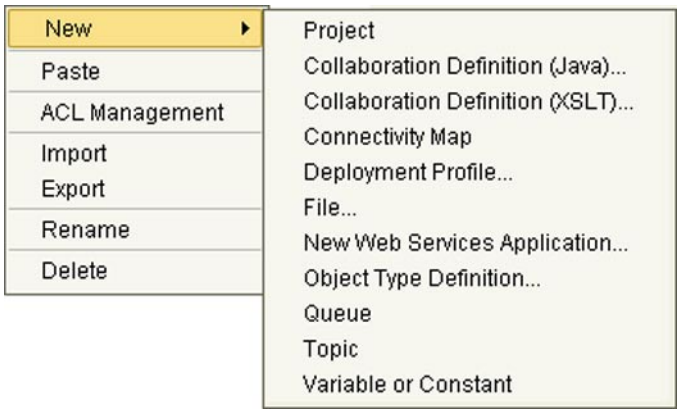


Table 20 Project Menu Options

Option	Option	Function
New	Project	Adds a Subproject folder to the selected Project.
	Collaboration Definition (Java)	Displays the Collaboration Definition Wizard (Java), with which you can create a Java-based Collaboration Definition. See**Using the Collaboration Definition Wizard (Java)** on page 169.
	Collaboration Definition (XSLT)	Displays the Collaboration Definition Wizard (XSLT), with which you can create an XSLT-based Collaboration Definition. See**Using the Collaboration Definition Wizard (XSLT)** on page 260.
	Connectivity Map	Adds a Connectivity Map to the Project. See **Using the Connectivity Map Editor** on page 104.
	Deployment Profile	Displays a dialog box with which you can create a Deployment Profile for the selected Project. See **The Deployment Editor** on page 333.
	File ...	If the File eWay is installed, displays a dialog box with which you can create an external file to use with the Project. (This is an example of an External Application—the applications that are displayed in the menu depend upon which eWays are installed on your system.)
	New Web Services Appl.	Adds a Web services application to the selected Project. See **Web Services Application** on page 352.
	Object Type Definition	Displays the OTD Wizard, with which you can create an Object Type Definition (OTD) file. See **OTD Wizards** on page 116 for more information.
	Queue	Adds a queue to your Project.
	Topic	Adds a topic to your Project.

Table 20 Project Menu Options

Option	Option	Function
New (continued)	Variable or Constant	Displays a dialog box with which you can add a constant or variable icon to your Project.
Paste		Pastes a Collaboration that has been cut or copied from another Project into the selected Project. Only one paste operation can be performed—if you want to paste multiple times, you must recopy the Collaboration each time.
ACL Management		Displays the ACL Properties dialog box, with which an Administrator can assign read/write/delete privileges to users for the selected Project. See the *eGate Integrator System Administration Guide*.
Import		Displays a dialog box with which you can import a Project as a Subproject under the selected Project. See **Project/Environment Import** on page 72.
Export		Displays a dialog box with which you can export the selected Project. See **Project/Environment Export** on page 76.
Rename		Activates the field, allowing you to rename the selected Project.
Delete		Deletes the selected Project, subject to the following conditions: ▪ You have *delete* 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 box is displayed in which you confirm that you want to delete the selected Project. Clicking **Yes** then deletes the Project.

Connectivity Map Menu

Figure 67 Connectivity Map Menu

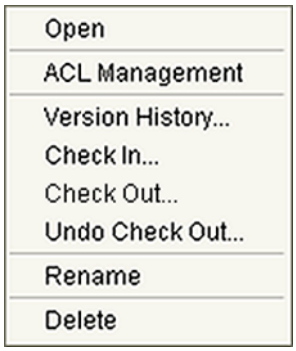


Table 21 Connectivity Map Menu Options

Command	Function
Open	Opens the Connectivity Map Editor, showing the selected Connectivity Map. See **Using the Connectivity Map Editor** on page 104.
ACL Management	Displays the ACL Properties dialog box, with which an Administrator can assign read/write/delete privileges to users for the selected Connectivity Map. See the *eGate Integrator System Administration Guide*.
Version History	Displays a dialog box with which you can track the version history for the selected Connectivity Map. See **Viewing a Component's Version History** on page 84 for more information.
Check In	Displays a dialog box, with which you can check in a new version of the selected Connectivity Map. See **Checking a Component In** on page 85 for more details.
Check Out	Displays a dialog box with which you can check out the current version of the selected Connectivity Map. See **Checking a Component Out** on page 87 for more information.
Undo Check Out	Displays a dialog box with which you can undo the check-out of the selected Connectivity Map. See **Checking a Component In Without Revisions** on page 86 for more information.
Rename	Activates the field, allowing you to rename the selected Connectivity Map.
Delete	Deletes the selected Connectivity Map, subject to the following conditions: ▪ You have *delete* privileges for the Connectivity Map (see *ACL Management*, above). ▪ The Connectivity Map is not checked out by anyone other than yourself. If these conditions are true, a dialog box is displayed in which you confirm that you want to delete the selected Connectivity Map. Clicking **Yes** then deletes the Connectivity Map.

Object Type Definition Menu

Figure 68 OTD Menu

Table 22 OTD Menu Options

Command	Function
Open	Opens the OTD Editor, showing the selected OTD. See **OTD Editor** on page 118.
ACL Management	Displays the ACL Properties dialog box, with which an Administrator can assign read/write/delete privileges to users for the selected OTD. See the *eGate Integrator System Administration Guide*.
Version History	Displays a dialog box with which you can track the version history for the selected OTD. See **Viewing a Component's Version History** on page 84 for more information.
Check In	Displays a dialog box, with which you can check in a new version of the selected OTD. See **Checking a Component In** on page 85 for more details.
Check Out	Displays a dialog box with which you can check out the current version of the selected OTD. See **Checking a Component Out** on page 87 for more information.
Relaunch (XSD OTDs only)	Relaunches the XSD OTD Wizard, so that you can re-define the selected OTD while retaining the original OID. The following conditions apply: ▪ You must have *write* privileges for the OTD (see *ACL Management*, above). ▪ The OTD must not be checked out by anyone other than yourself. ▪ The OTD must not be imported from another Project. ▪ The OTD must have been created in eGate Integrator 5.0.4 (or later release).
Rename	Activates the field, allowing you to rename the selected OTD.
Delete	Deletes the selected OTD, subject to the following conditions: ▪ You have *delete* privileges for the OTD (see *ACL Management*, above). ▪ The OTD is not checked out by anyone other than yourself. If these conditions are true, a dialog box is displayed in which you confirm that you want to delete the selected OTD. Clicking **Yes** then deletes the OTD.

Collaboration Definition Menu

Figure 69 Collaboration Definition Menu

Table 23 Collaboration Definition Menu Options

Command	Function
Open	Opens the appropriate Collaboration Editor, showing the selected Collaboration Definition. See **Using the Collaboration Editor (Java)** on page 179 and **Using the Collaboration Editor (XSLT)** on page 266.
Cut	Copies the selected Collaboration Definition (Java only) and removes it from the current Project, after which you can paste it to another Project (once only). All changes must be committed before you can cut the Collaboration. Cut and paste is disabled for other users when you have the Collaboration checked out.
Copy	Copies the selected Collaboration Definition (Java only), after which you can paste it to other Projects (multiple times). All changes must be committed before you can copy the Collaboration. You can copy and paste a Collaboration even when another user has the Collaboration checked out.
Version History	Displays a dialog box with which you can track the version history for the selected Collaboration Definition. See **Viewing a Component's Version History** on page 84 for more information.
Check In	Displays a dialog box, with which you can check in a new version of the selected Collaboration Definition. Refer to **Checking a Component In** on page 85 for more details.
Check Out	Displays a dialog box with which you can check out the current version of the selected Collaboration Definition. See **Checking a Component Out** on page 87 for more information.
Undo Check Out	Displays a dialog box with which you can undo the check-out of the selected Collaboration Definition. See **Checking a Component In Without Revisions** on page 86 for more information.

Table 23 Collaboration Definition Menu Options

Command	Function
Make Latest	Allows you to check in the version of the selected Collaboration Definition that was retrieved to your workspace, making it the latest version. See **Checking In a Previous Version as the Latest Version** on page 90.
Create Diff	Displays a dialog box with which you can create a different version of the selected Collaboration Definition. See **Creating a Modified Collaboration Definition (Java)** on page 209 and **Creating a Modified Collaboration Definition (XSLT)** on page 288.
Merge Diff	Displays a dialog box with which you can merge two different versions of the selected Collaboration Definition. See **Merging Two Versions of a Collaboration Definition (Java)** on page 210 and **Merging Two Versions of a Collaboration Definition (XSLT)** on page 288.
Delete	Deletes the selected Collaboration Definition, subject to the following conditions: ▪ You have *delete* privileges for the Collaboration Definition (see *ACL Management*, below). ▪ The Collaboration Definition is not checked out by anyone other than yourself. If these conditions are true, a dialog box is displayed in which you confirm that you want to delete the selected Collaboration Definition. Clicking **Yes** then deletes the Collaboration Definition.
ACL Management	Displays the ACL Properties dialog box, with which an Administrator can assign read/write/delete privileges to users for the selected Collaboration Definition. See the *eGate Integrator System Administration Guide*.
Properties	Displays the appropriate Collaboration Definition Properties dialog box for the selected Collaboration Definition.

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.*

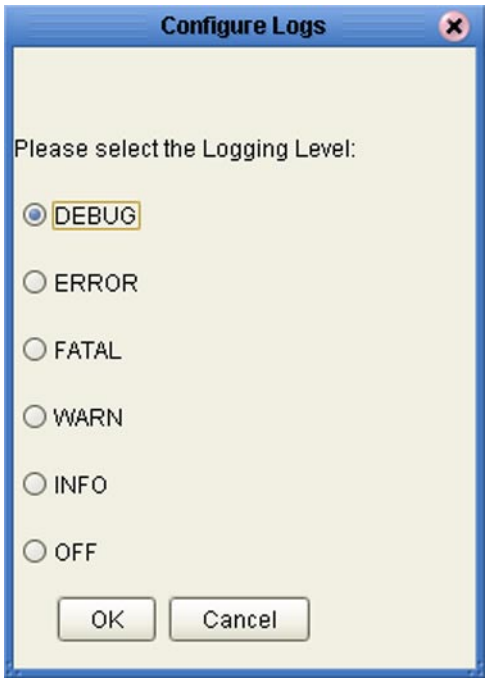
Deployment Profile

Figure 70 Deployment Profile Menu

Table 24 Deployment Profile Menu Options

Command	Function
Version History	Displays a dialog box with which you can track the version history for the selected Deployment Profile. See **Viewing a Component's Version History** on page 84 for more information.
Check In	Displays a dialog box, with which you can check in a new version of the selected Deployment Profile. See **Checking a Component In** on page 85 for more details.
Check Out	Displays a dialog box with which you can check out the current version of the selected Deployment Profile. See **Checking a Component Out** on page 87 for more information.
Undo Check Out	Displays a dialog box with which you can undo the check-out of the selected Deployment Profile. See **Checking a Component In Without Revisions** on page 86 for more information.
Open	Opens the Deployment Editor, showing the selected Deployment Profile. See **The Deployment Editor** on page 333.
Rename	Activates the field, allowing you to rename the selected Deployment Profile.
Delete	Deletes the selected Deployment Profile, subject to the following conditions: ▪ You have *delete* privileges for the Deployment Profile (see *ACL Management*, above). ▪ The Deployment Profile is not checked out by anyone other than yourself. If these conditions are true, a dialog box is displayed in which you confirm that you want to delete the selected Deployment Profile. Clicking **Yes** then deletes the Deployment Profile.

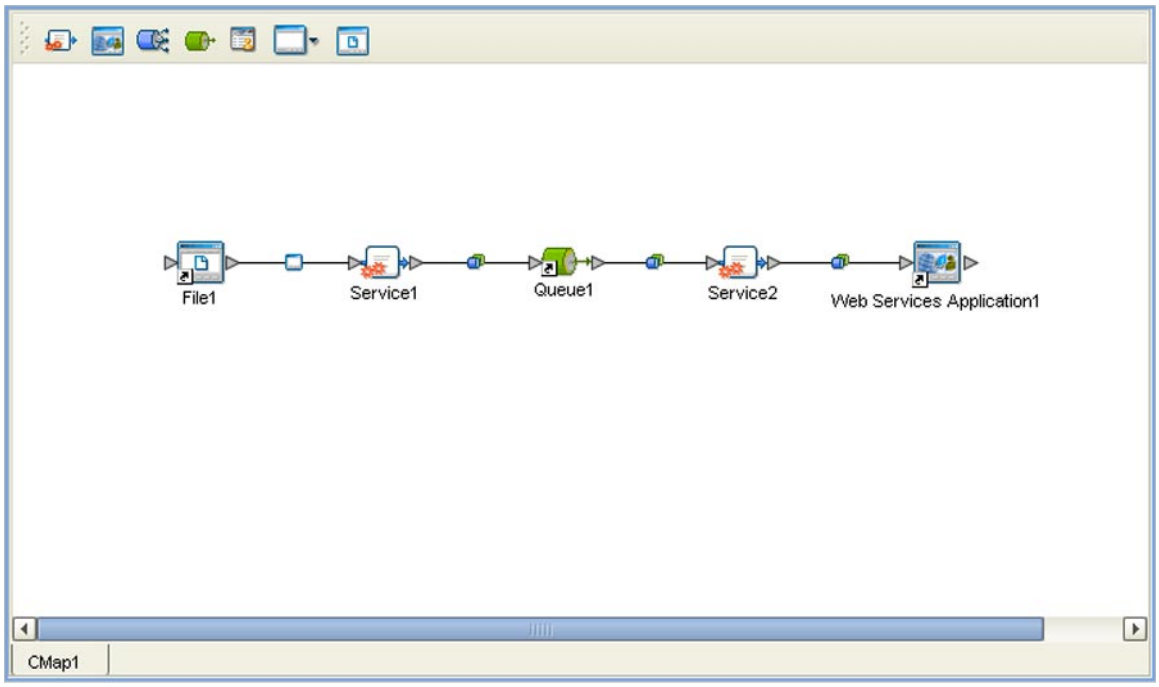
Table 24 Deployment Profile Menu Options

Command	Function
Configure Log	Displays a dialog box in which you can set the logging level for the selected Deployment Profile.  The default value is DEBUG. See the *eGate Integrator System Administration Guide* for information about logging.
ACL Management	Displays the ACL Properties dialog box, with which an Administrator can assign read/write/delete privileges to users for the selected Deployment Profile. See the *eGate Integrator System Administration Guide*.

5.3 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 71). 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 subsequently link the components by dragging the cursor from one to the other.

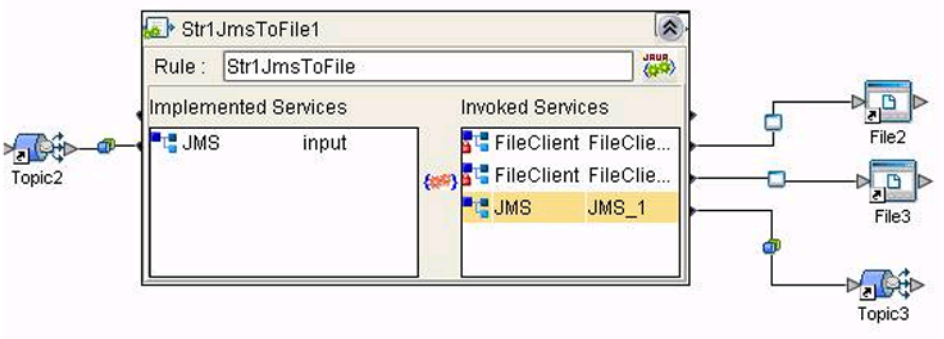
Figure 71 Connectivity Map Editor



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.

When there are multiple destinations, as with a JMS topic, the Connectivity Map Editor cannot resolve which output port connects to which destination. Because of this, the Collaboration definition must be created first, and the connections must be drawn by opening the Collaboration Binding box in Connectivity Map (see Figure 72).

Figure 72 Linking JMS Topics



The Connectivity Map Editor toolbar contains the icons listed in Table 25, plus additional icons representing eGate add-ons and other ICAN components that you may have installed.

Table 25 Connectivity Map Toolbar Icons

Icon	Component	Function
	Service	A logical component that provides the framework for a process or Collaboration. See **Service Component** on page 106.
	Queue	A Message Destination that conforms to the point-to-point messaging paradigm, having one sender and one receiver. See the *eGate Integrator JMS Reference Guide* for information.
	Topic	A Message Destination that conforms to the publish/subscribe messaging paradigm, having one sender (publisher) and multiple receivers (subscribers). See the *eGate Integrator JMS Reference Guide for* information.
	Web Services Application	Represents a Web services application (see **Web Services Application** on page 352).
	External Applications	Represents an application external to eGate. Click the arrow beside the icon to view a list of specific applications to which you can connect. See **External Application Drop-Down Menu** on page 107.
	Scheduler	Represents a scheduling component of the Connectivity Map. Use this component to set data transfer to occur at set intervals. See **Schedulers** on page 108.

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.

5.4 Services

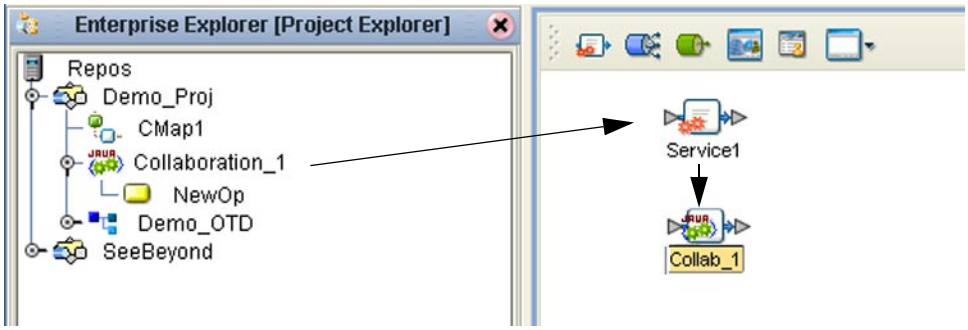
A service provides a framework for a process or a Collaboration, which contains the information required to execute a set of business rules.

5.4.1 Collaborations

A Collaboration is a logical operation performed between some combination of message destinations and external applications. The operation is defined by a Collaboration Definition, which can be encoded in 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 **Component Connections** on page 109. Dragging a Collaboration from the Project Explorer to the Service icon in the Connectivity Map defines the service as a Collaboration (see Figure 73).

Figure 73 Service Component



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. For safe measure, this should also be done before creating the Connectivity Map and Deployment Profile.*

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

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

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

5.5 Message Destinations

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

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

See the *eGate Integrator JMS Reference Guide* for information on message destinations.

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 system. These are linked to a Service by means of an eWay. Clicking the drop-down arrow beside the external application icon displays a menu showing those applications corresponding to eWays that have been purchased and installed, plus the Scheduler. An example is shown in Figure 74.

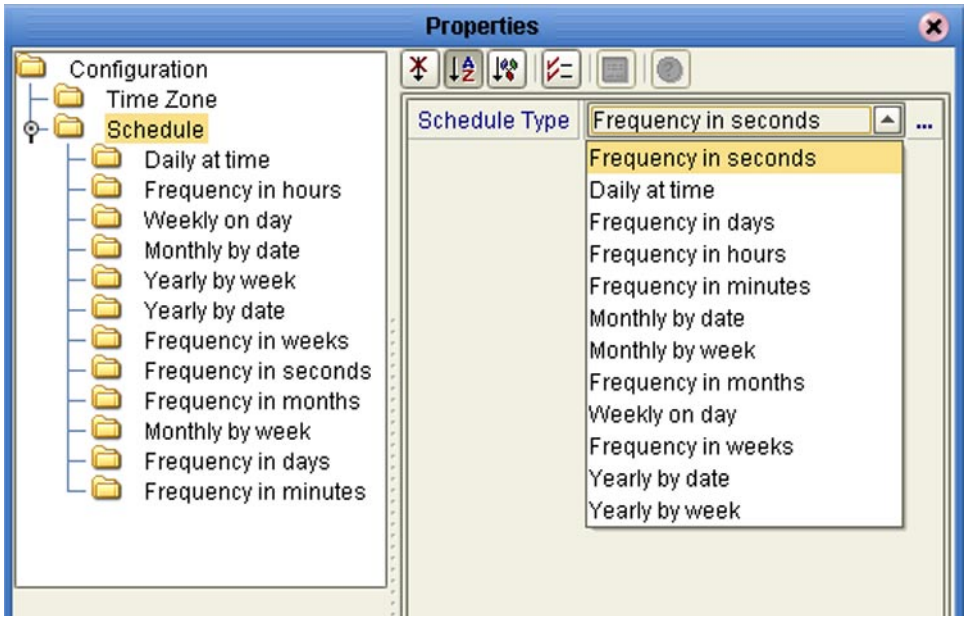
Figure 74 External Application Drop-Down Menu

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

5.6.1 Schedulers

A Scheduler allows a service to be performed at a prescribed interval. The interval can be static, or can be made dynamic by using a Project variable for the interval value. Once the scheduler is connected to a service in the Connectivity Map, double-clicking the JMS Client displays the Properties dialog box for that scheduler (seeFigure 75).

Figure 75 Scheduler Properties Dialog Box



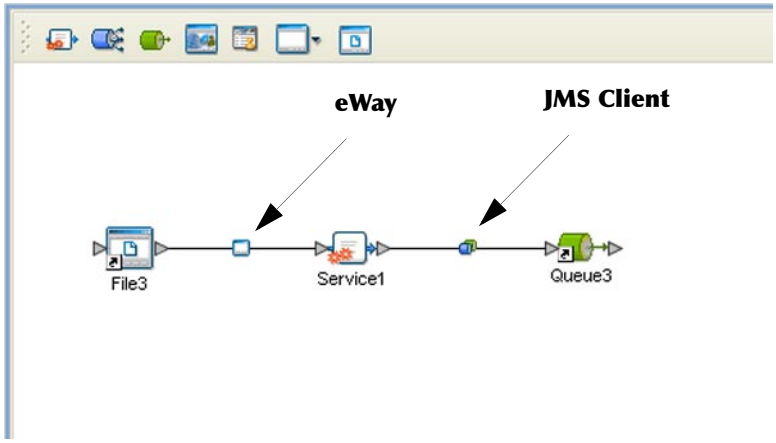
Selecting **Time Zone** displays the **Time Zone** property field in which you specify your local time zone, so that your schedule will be synchronized to the local time, if appropriate.

Selecting **Schedule** displays the **Schedule Type** property field which you set to the type of schedule you want to use. Selecting the corresponding node in the explorer tree displays the property field for that schedule type, in which you specify the desired value. The text in the *Description* box will include the appropriate units.

5.7 Component Connections

When you link two components on a Connectivity Map, the Enterprise Designer places either an eWay or JMS Client connection icon on the link, depending upon the type of components you are linking (see Figure 76).

Figure 76 Connection Icons in a Connectivity Map

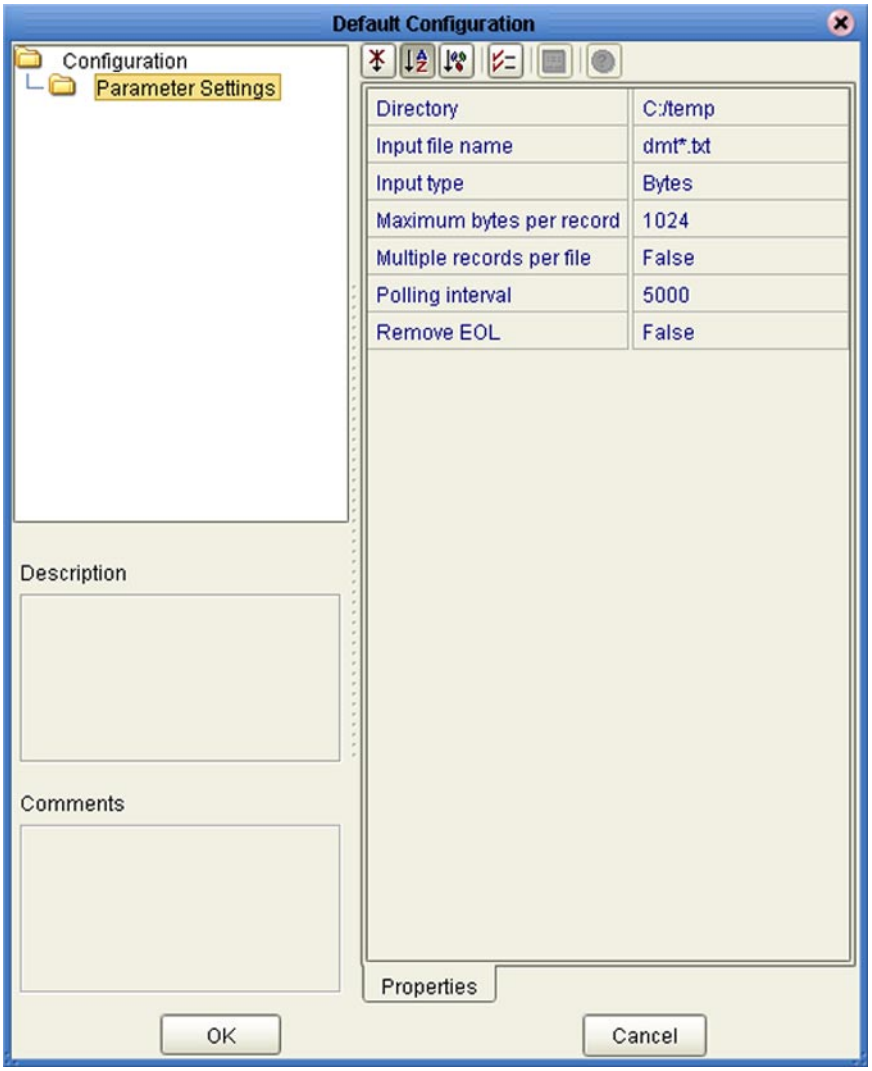


- When you link an external application with a Collaboration, the Enterprise Designer automatically adds an eWay Connection icon to the link. The eWay enables communication and movement of data between the external application and the eGate system. The eWay configuration specifies the logical connection properties for the link. See the individual eWay Intelligent Adapter User's Guides for specific information.
- When you link a Service with a Message Destination (queue or topic), the Enterprise Designer adds a JMS Client Connection icon. The JMS Client configuration specifies the logical connection properties for the linked Service. See the *eGate Integrator JMS Reference Guide* for information.

5.7.1 Configuring a Connection

Double-clicking an eWay or JMS Client connection icon in the Connectivity Map displays the Default Configuration dialog box. As an example, Figure 77 shows a dialog box that lists the configuration properties for a File eWay.

Figure 77 Default Configuration Dialog Box



Note: *The first time you double-click an eWay or JMS Client icon, you will see a Templates dialog box. Here, you must designate an eWay to be inbound or outbound. Clicking* ***OK*** *will then display the Default Configuration dialog box.*

The constituent parts of the Default Configuration dialog box are:

- The **Configuration Tree** includes folders that contain configuration and connection properties for the selected eWay or message destination.
- The **Toolbar** contains a series of buttons used to sort and modify the information listed in the Properties folder, as described in Table 26.
- The **Description** box contains a brief description of the contents of the item currently selected in the Configuration Tree.
- The **Comments** box is for user comments about the item selected in the Configuration Tree.

Table 26 Configuration Dialog Box Toolbar Buttons

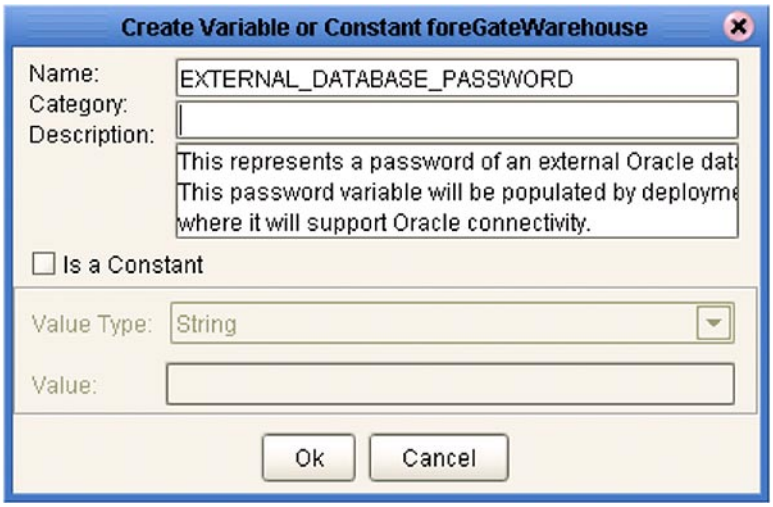
Button	Command	Function
	Unsorted	Displays configuration properties in their default order.
	Sort by Name	Sorts configuration properties alphabetically by name.
	Sort by Type	Displays configuration properties by property type.
	Show Editable Properties Only	Displays only the properties of an eWay or message destination that can be modified.
	Customizer	Displays the **Customizer** dialog box, which you can use to customize the selected eWay or message destination.
	Help	Displays the online help documentation for the Configuration Editor.

5.8 Defining Constants and Variables

You can define variables and constants for a specific Project. Variables function as placeholders, having values that are determined when you create a specific Deployment Profile (see **Mapping Variables** on page 340). Project variable values can be literals or Environmental constants.

For example, Figure 78 shows a project variable defined to represent a password of a database user in a target environment. System managers will assign an actual value to this variable in the deployment profile editor. The value of the assigned project variable—an Environment constant— is then used to connect the database in the target environment.

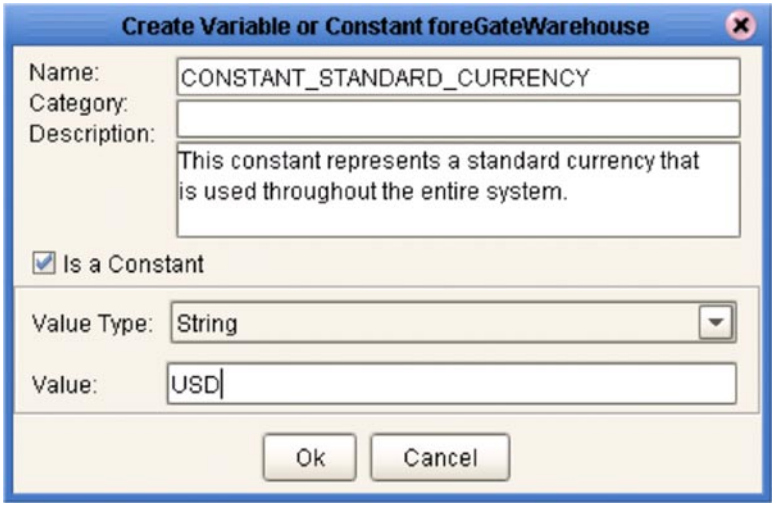
Figure 78 Project Variable Creation



Project constants are name/value pairs that are visible across the Project. For example, Figure 79 shows a standard currency defined to be used globally throughout the system.

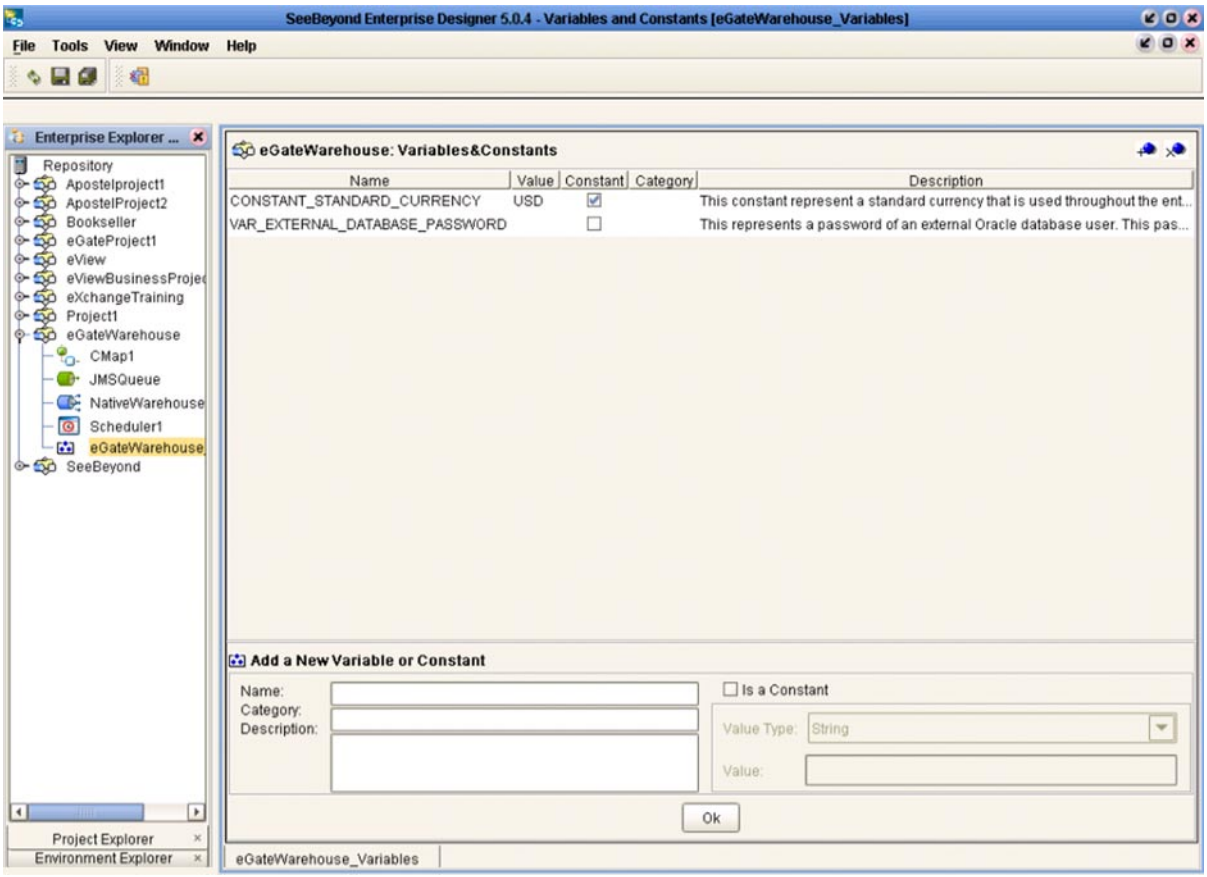
Note: *When you create an Project constant, you assign a permanent value to it—which cannot be overridden.*

Figure 79 Project Constant Creation



Constants and variables are automatically added to a Variables and Constants object group within the Project (see Figure 80).

Figure 80 Variables and Constants Object Group



Chapter 6

Object Type Definitions

This chapter contains a brief introduction to OTDs, and describes the procedures for creating both externally-defined and user-defined OTDs.

6.1 Overview

Object Type Definitions (OTDs) describe external data formats that characterize the input and output data structures in a Collaboration Definition. OTDs typically have a specific external representation format that is used to store and transport the OTD contents through an eGate Project. The OTD defines both this external representation and the run-time structure.

Typically, a Collaboration will 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 either directly from Java in a Java-based Collaboration, using accessors resembling Java beans, 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.

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* **Impact Analyzer** *on page 81).*

6.1.1 OTD Types

Externally-Defined OTDs

Externally-defined OTDs are based on formats or standards external to eGate Integrator, such as Document Type Definition (DTD), Web Services Definition Language (WSDL), XML Schema Definition (XSD), and various proprietary formats such as SAP BAPI. Some of these OTDs are *messagable*, others are API-based. Externally-defined OTDs are read-only. See **Creating Externally-Defined OTDs** on page 126.

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. User-defined OTDs are read/write—you can add or delete nodes and edit their properties. See **Creating User-Defined OTDs** on page 142.

JMS OTDs

For Collaborations that read from and write to topics or queues, you must add the JMS OTD to the Collaboration Definition. This 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. JMS OTDs are described in the *eGate Integrator JMS Reference Guide*.

OTD Libraries

Several libraries containing large numbers of OTDs for specific purposes are available to use with eGate Integrator. These OTDs are pre-defined templates corresponding to message types used by SWIFT, HL7, and other industry-specific data exchange systems. See **OTD Libraries** on page 167.

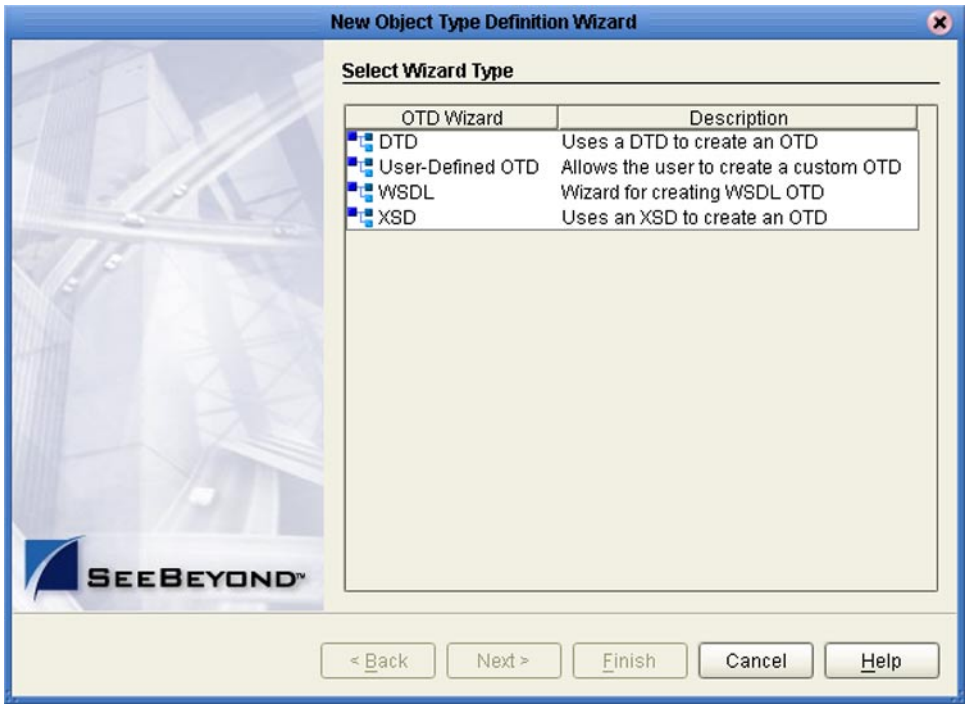
6.2 OTD Wizards

Wizards are provided in the Enterprise Designer GUI to guide you through the OTD building process. These Wizards call back-end builders that actually implement the building of the code, based on the provided information.

Right-click on a Project in the Enterprise Explorer to display the Project context menu, then select **New Object Type Definition** to display the OTD Wizard selection dialog, shown in Figure 81. This initial dialog allows you to select the specific type of OTD Wizard needed for your application. The basic Wizards supplied with eGate Integrator are described in:

- **Using the DTD Wizard** on page 126
- **Using the WSDL Wizard** on page 132
- **Using the XSD Wizard** on page 136
- **Using the User-Defined OTD Wizard** on page 142

Figure 81 OTD Wizard Selection Dialog



Additional OTD Wizards are supplied with eGate 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 81.

The OTD Wizards guide you through the initial phases of creating an Object Type Definition, and then invoke the OTD Editor. The user interface is highly self-explanatory, but details of the navigation buttons are listed in Table 27 for your reference.

Table 27 OTD Wizard Navigation Buttons

Button	Function
< Back	Returns to the previous step in the wizard. This button is disabled on the first step.
Next >	Goes to the next step in the wizard. This button is disabled on the last step.
Finish	Saves all OTD settings and closes the wizard. This button is only enabled on the last step.
Cancel	Closes the wizard without saving the OTD.
Help	Displays the online help documentation for the **OTD Wizard** dialog box.

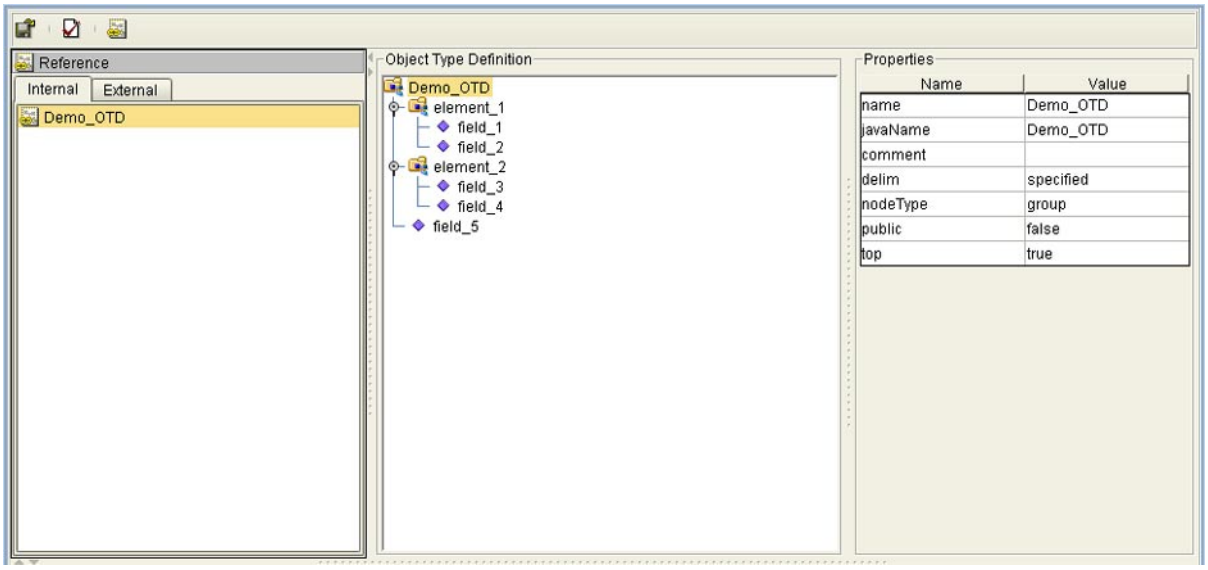
6.3 OTD Editor

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 82. 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 **Creating User-Defined OTDs** on page 142.

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 82 OTD Editor



Major features of the OTD Editor interface are:

- **Reference**
  This area contains internal and external templates for the OTD file.
- **Object Type Definition**
  This area displays each field and element included in the OTD file.
- **Properties**
  This area displays details about the OTD file or field selected in the *Object Type Definition* list.

Table 28 OTD Editor Toolbar Icons

Icon	Command	Function
	Save as New Name in Repository	Saves current OTD under a new name in the Repository.
	Tester	Displays/refreshes the Tester area.
	Toggle Reference Tab Panel	Displays/hides the Reference area.

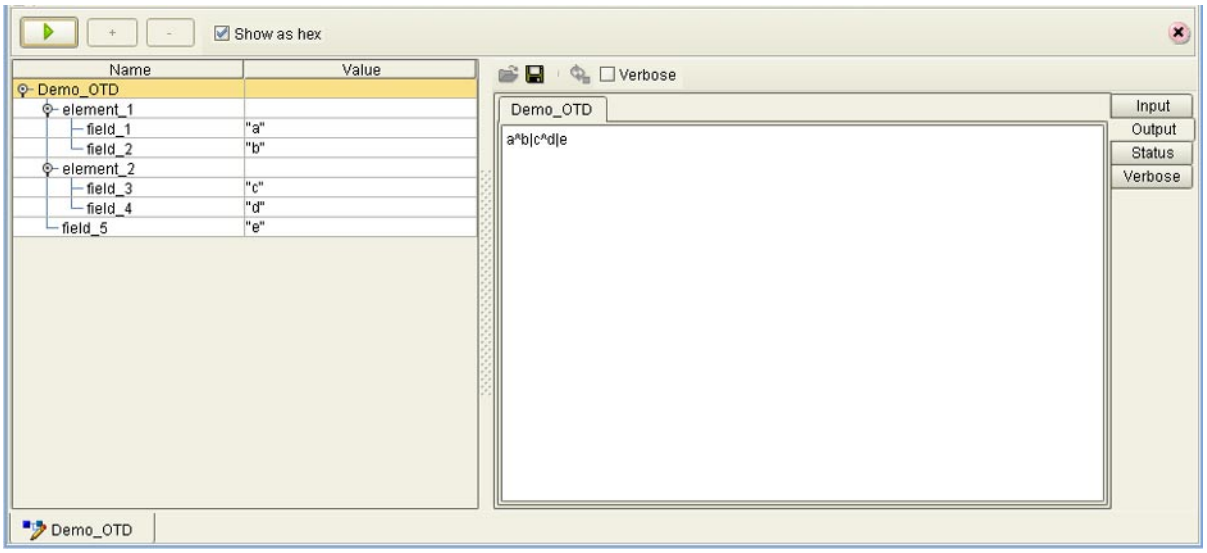
6.3.1 OTD Tester

The OTD tester provides a facility to verify the correctness of OTDs, for example to:

- Prevent data errors at runtime.
- Verify that all required data elements are available.
- Verify that all used data formats are correct.

Clicking the **Tester** icon (see Table 28) in the OTD Editor toolbar saves the OTD to the Repository and displays the OTD Tester as the lower part of the editor, as shown in Figure 83.

Figure 83 OTD Tester



The data display panel on the right has four data display modes, selectable by tabs. The *Input* tab is selected by default. Use of the OTD Tester is described in **Using the OTD Tester** on page 123.

Table 29 OTD Tester Buttons

Button	Command	Function
	Run Tester	Runs the tester with the data values entered in the table; the data is marshaled and displayed in the *Output* panel.
	Add Instance	Adds an instance of an optional node (button is deactivated when an optional instance already exists) or an 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.

Next to the OTD Tester buttons is a check box labeled **Show as hex**. When checked, the values for **byte[]** nodes are shown in hexadecimal format (default setting). When not checked, the values are shown as regular alphanumeric text.

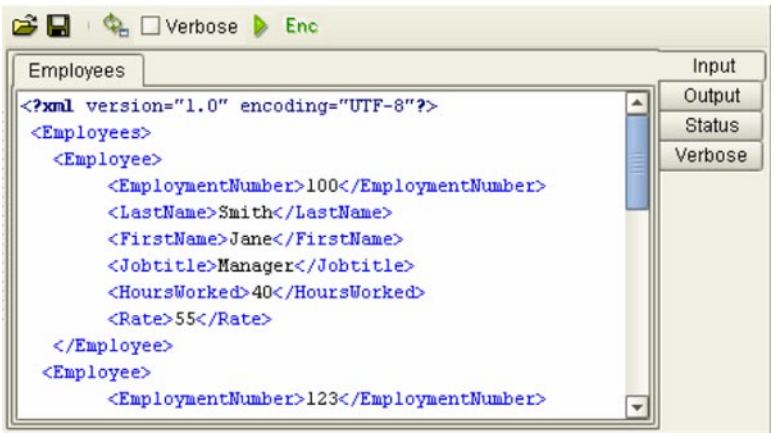
Table 30 OTD Tester Icons

Icon	Command	Function
	Open	Displays file browser.
	Save	Saves displayed file.
	Refresh	Repopulates the OTD object elements with the values from the data display *Input* panel.
	Encoding	Invokes the *Specify Encoding* dialog box, where you can specify the data encoding for the input file (applies to XML-based data only). **Note:** Icon is displayed only if you are using the extended language option (see **Options Setup** on page 62).

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

If you are using the extended language options (see **Options Setup** on page 62), an *Encoding* icon appears in the OTD Tester as shown in Figure 84. This icon allows you to specify the data encoding for the input file.

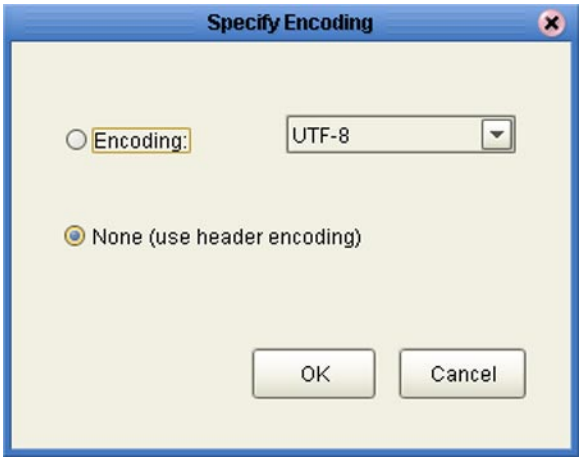
Figure 84 OTD Tester - Data Encoding Enabled



To specify the data encoding (Japanese and Korean locales only)

1 Click the *Encoding* (**Enc**) icon above the data display panel to display the Specify Encoding dialog box shown in Figure 85.

Figure 85 Specify Encoding Dialog Box



- Select **None** (the default setting) to use the header encoding; if the header encoding does not exist, the default locale encoding is used instead.
- Select **Encoding** to explicitly specify the encoding from a drop-down list, the contents of which depends upon the locale (see Table 31).

Table 31 Encoding Options

Japan	Korea
Shift JIS	
MS932	CP949
EUC-JP	EUC-KR
UTF-8	UTF-8
ASCII	ASCII

2 Click **OK** to save your selection.

Note: *The locale is defined at the operating-system level. For example, the locale for Windows is defined in the* ***Regional and Language Options****.*

6.3.2 Using the OTD Tester

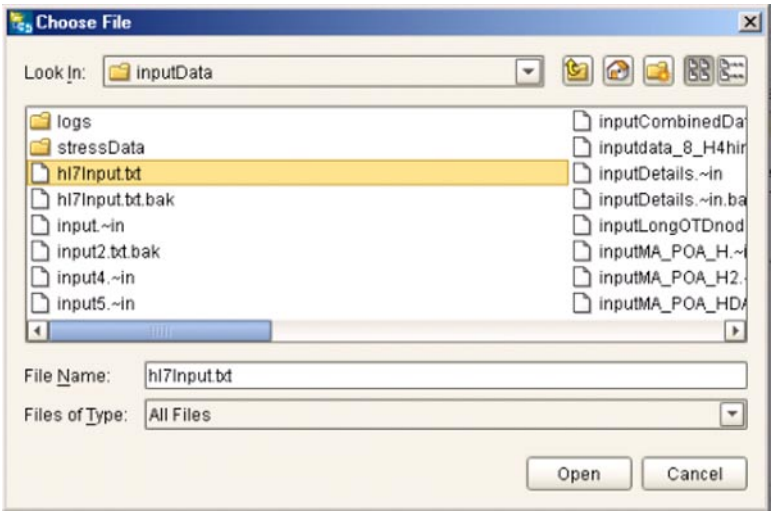
To test data unmarshaling functionality

1 Open or create an OTD.

2 Click the **Tester** icon to display the OTD Tester.

3 Provide the input test data by selecting a data file (see Figure 86).

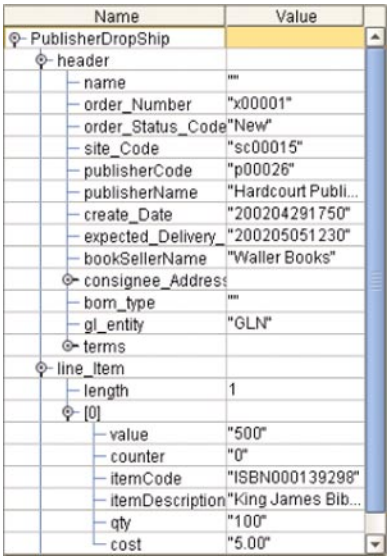
Figure 86 Select Data File



4 Click the **Refresh** button to unmarshal the data.

5 Verify the output by checking the values for each element for correctness (see Figure 87).

Figure 87 Object Elements and Values



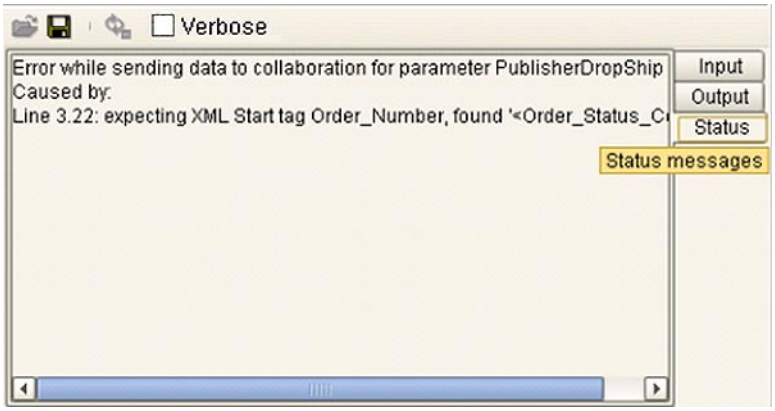
6 Save your input test data to a file for re-use by selecting the *Input* panel and clicking the **Save** icon.

7 You can also change your test data in the *Input* panel, then re-test the OTD by clicking the **Refresh** icon (see **Table 30 on page 120**) to repopulate your OTD object elements with the new values.

Note: *Data, which you can edit for testing, is shown in black type.*

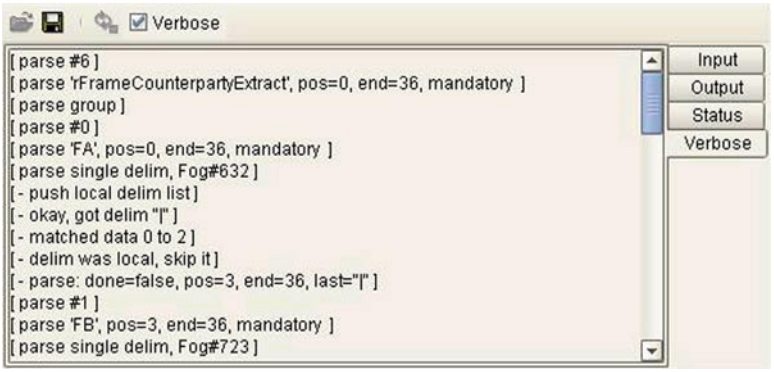
8 If there are errors in your input data, the **Status** panel is automatically invoked, showing the appropriate error messages (see Figure 88).

Figure 88 Data Display - Status Panel



9 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 (see Figure 89). The format and content of the data display are OTD-specific.

Figure 89 Data Display - Verbose Panel



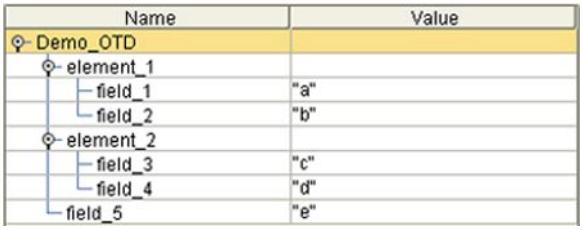
To test data marshaling functionality

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 90). Use the +/- buttons to add or remove instances where appropriate.

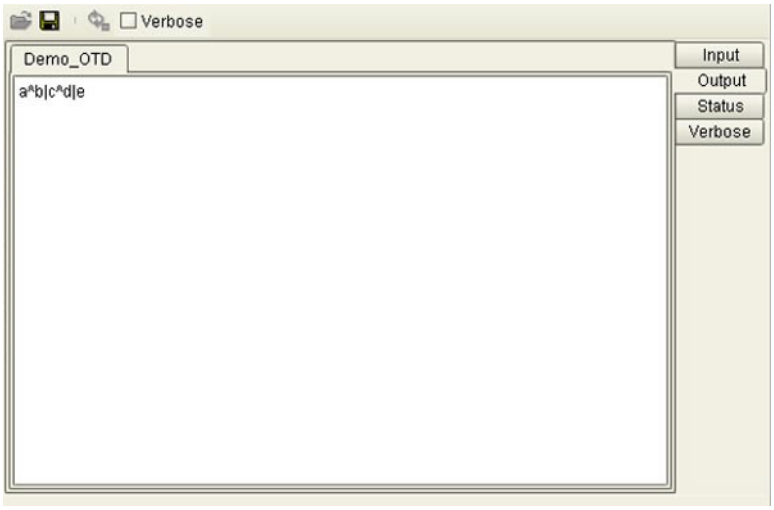
Figure 90 OTD Tester Node Table



4 Click the Run Tester button to marshal (serialize) the data.

5 The output is displayed in the *Output* panel (see Figure 91).

Figure 91 Serialized Data in Output Panel



6.4 Creating Externally-Defined OTDs

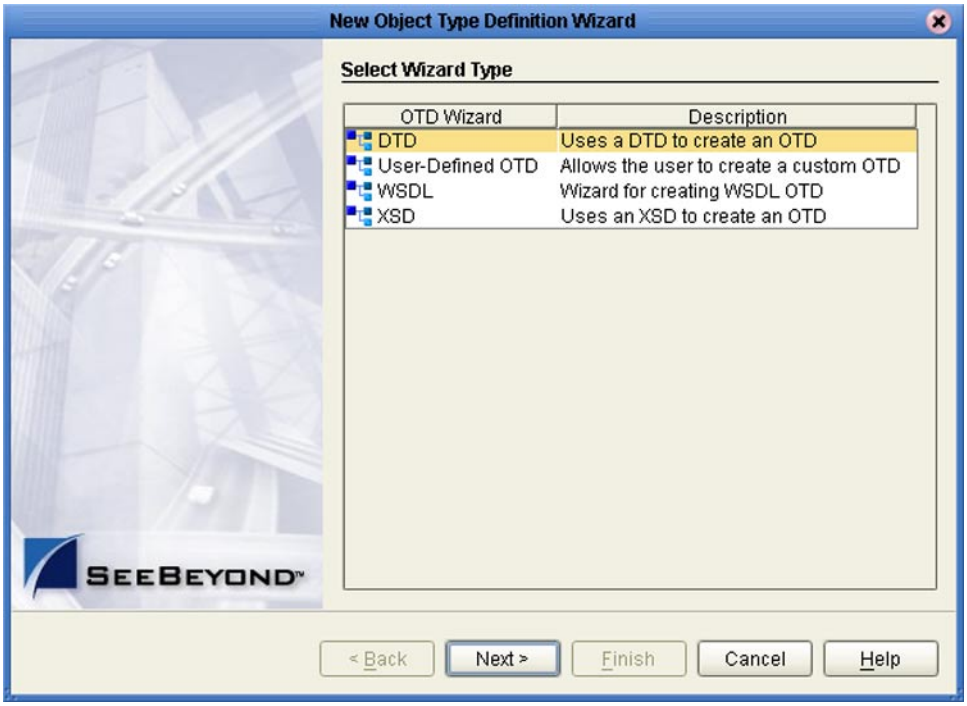
6.4.1 Using the DTD Wizard

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

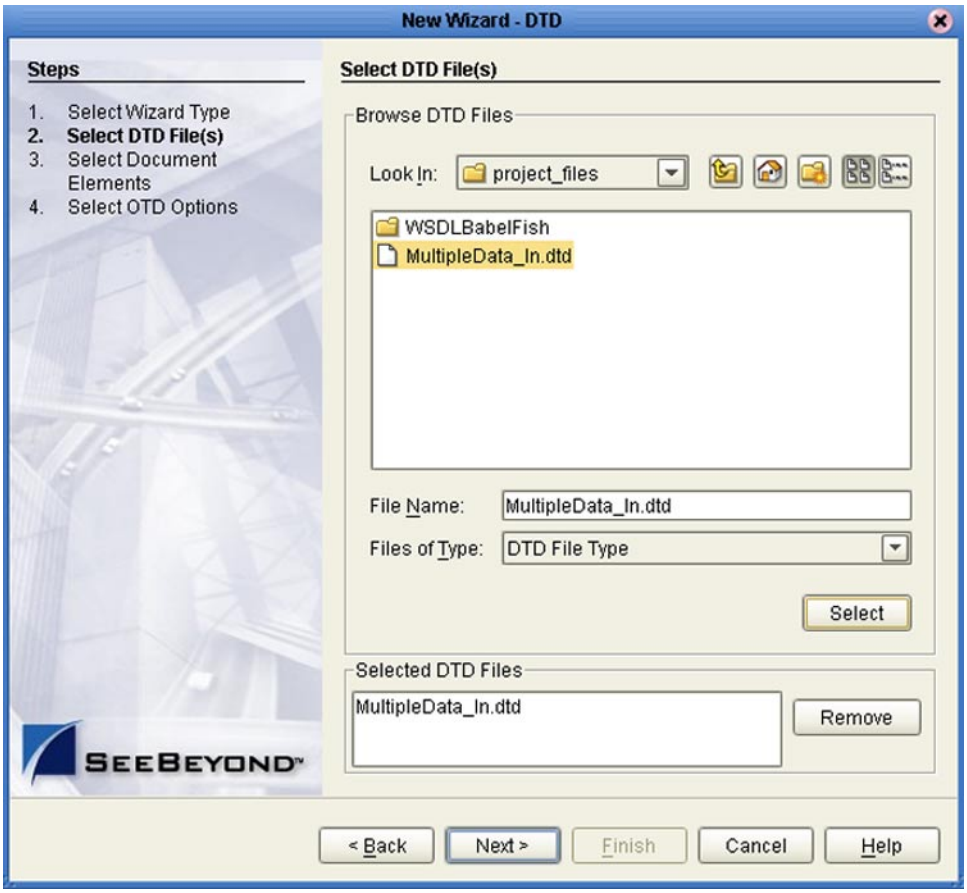
1 In the *Select Wizard Type* dialog, select **DTD** from the *OTD Wizard* list (see Figure 92) to create an OTD from a Data Type Definition (DTD) file.

Figure 92 OTD Wizard Selection: DTD Wizard



2 Click **Next** to display the *Select DTD File(s)* dialog box, shown in Figure 93.

Figure 93 Select DTD File(s) Dialog Box



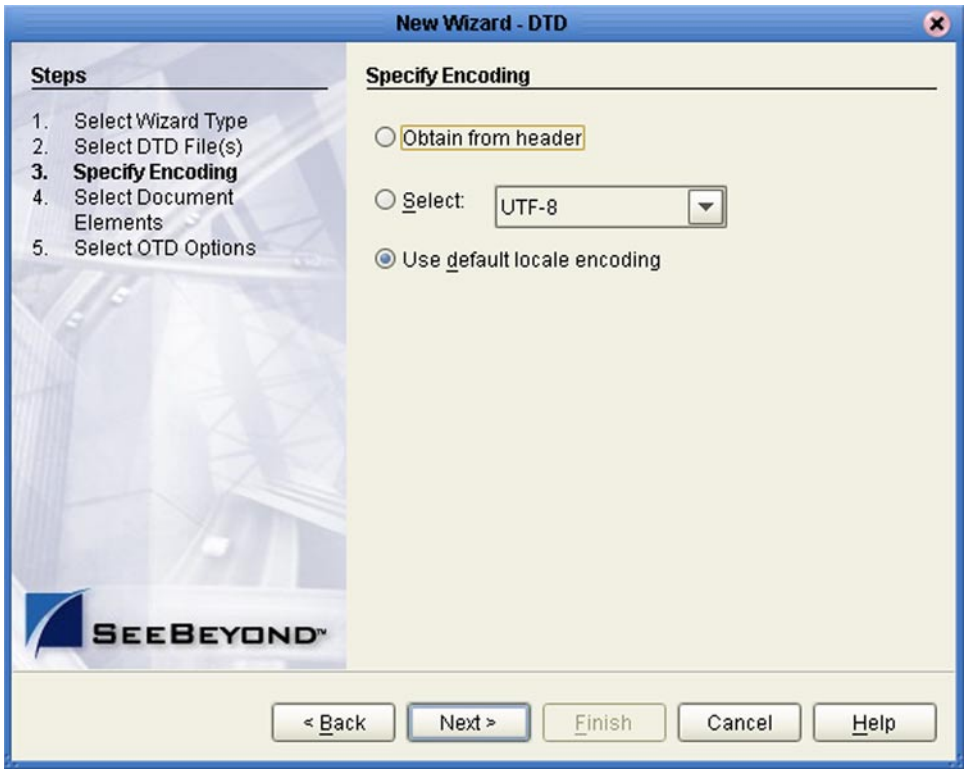
3 In the *Look In* drop-down list, navigate to the DTD file or files that you want to use to create the OTD. Click **Select** to add the files to the *List of Selected DTDs*.

- 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 94 will be displayed.

Figure 94 Cannot Create OTD Warning Box

4 If you are using the extended language options (see **Options Setup** on page 62) click **Next** to display the *Specify Encoding* dialog box, shown in Figure 95. If you are *not* using the extended language options, this Wizard dialog will not appear—proceed to step 6.

Figure 95 Select Encoding Dialog Box



5 Click the appropriate option button to specify one of the following methods for the OTD to encode the data:

- **Obtain from header** does not apply to DTD OTDs.
- **Select** explicitly specifies the encoding from a drop-down list, the contents of which depends upon the locale (see Table 32).

Table 32 Encoding Options

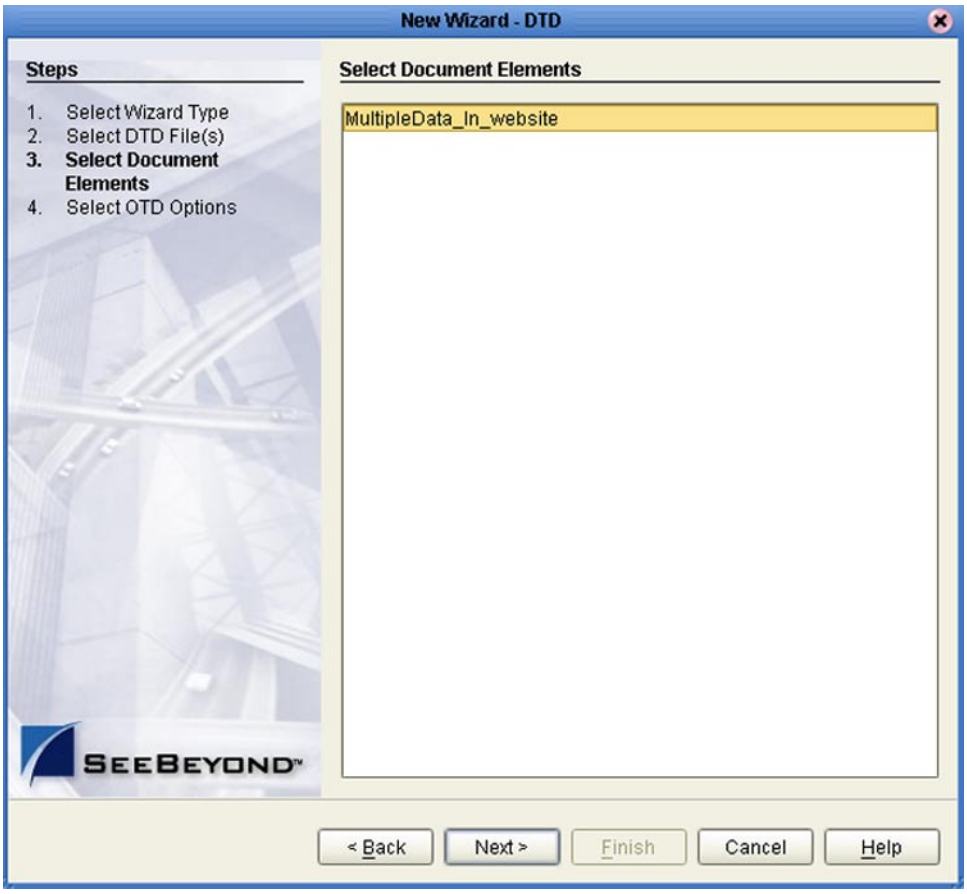
Japan	Korea
Shift JIS	
MS932	CP949
EUC-JP	EUC-KR
UTF-8	UTF-8
ASCII	ASCII

- **Use default locale encoding** uses the default encoding for the locale, based on the installed version of eGate Integrator.

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

6 Click **Next** to display the *Select Document Elements* dialog box, shown in Figure 96.

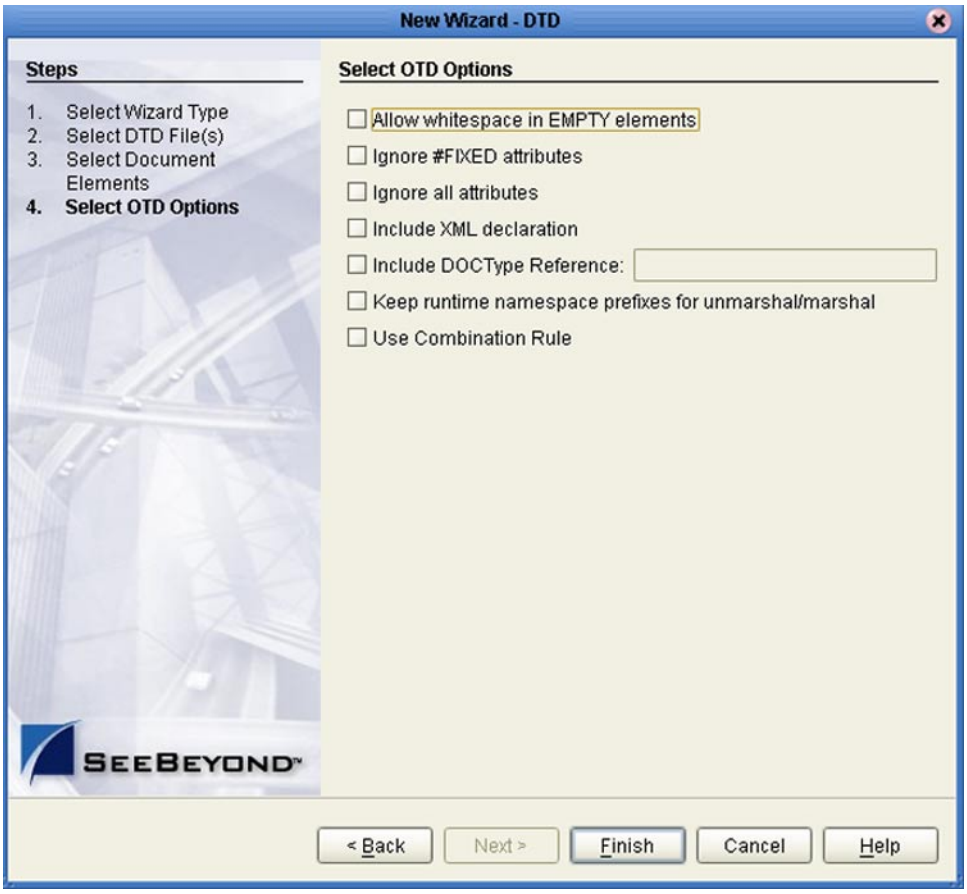
Figure 96 Select Document Elements Dialog Box



7 Select the elements of the document that you want to include in the OTD.

8 Click **Next** to display the *Select OTD Options* dialog box, shown in Figure 97.

Figure 97 Select OTD Options Dialog Box



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

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	This option 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	This option 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	This option controls whether or not the XML declaration **<?xml version="1.0" encoding="....."?>** appears in the output during the marshal process.
Include DOC Type Reference	This option controls whether or not the **"<!DOCTYPE ..."** string appears in the output during the marshal process.
Keep runtime namespace prefixes for unmarshal/ marshal	This option 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.

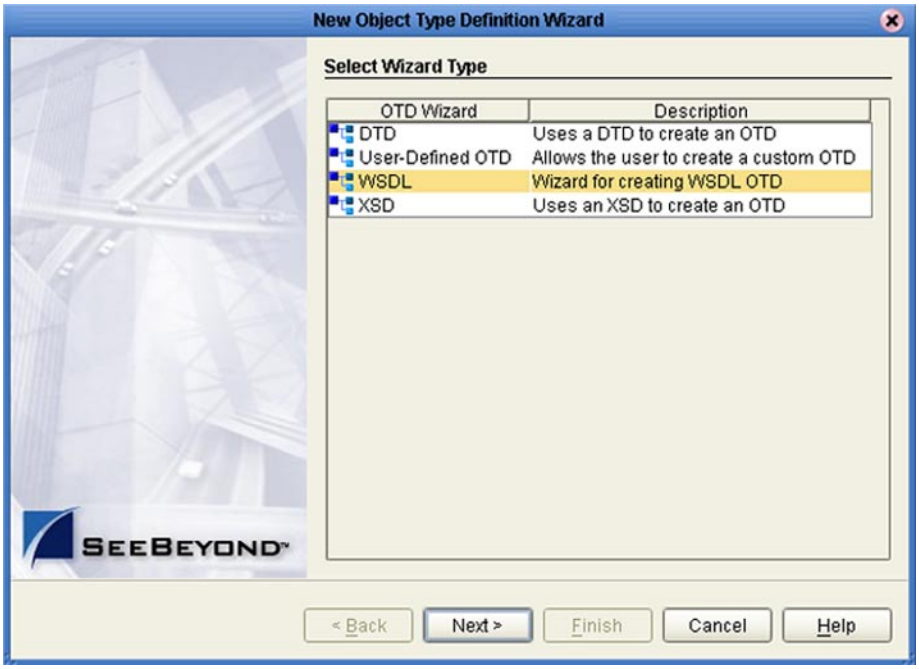
10 Click **Finish** to add the OTD to your Project and invoke the OTD Editor.

6.4.2 Using the WSDL Wizard

To create an OTD file from a WSDL file

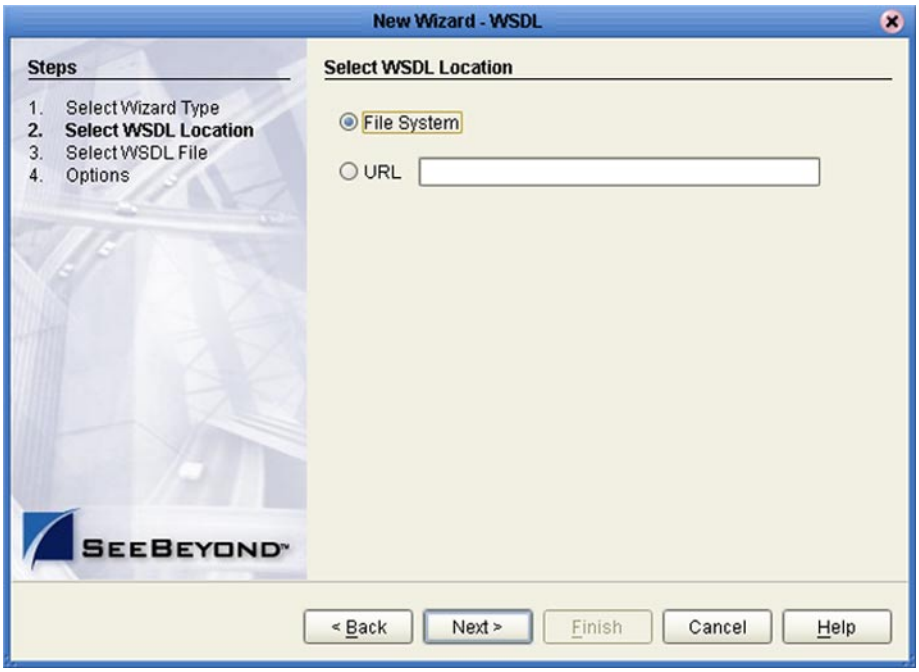
1 In the *Select Wizard Type* dialog, select **WSDL** from the *OTD Wizard* list (see Figure 98) to create an OTD from an WSDL file.

Figure 98 OTD Wizard Selection: WSDL Wizard



2 Click **Next** to display the *Select WSDL File Location* dialog, shown in Figure 99

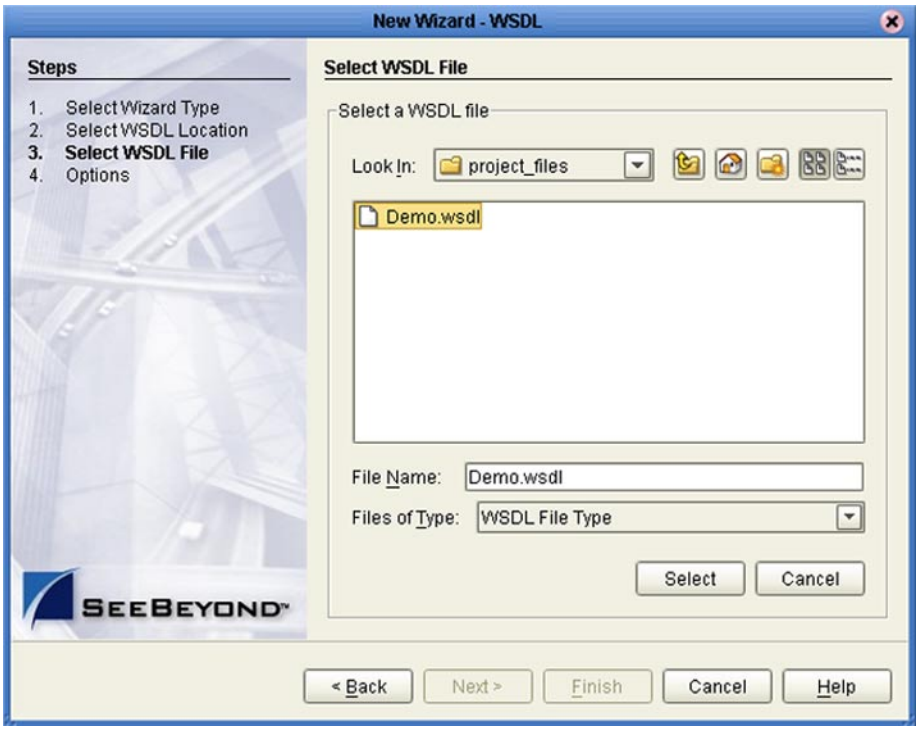
Figure 99 WSDL Wizard: Select WSDL Location



3 In the *Select WSDL Location* dialog, select **File System** or enter a **URL**, depending upon where your WSDL file is located.

4 Click **Next** to display the *Select WSDL File* dialog, shown in Figure 100.

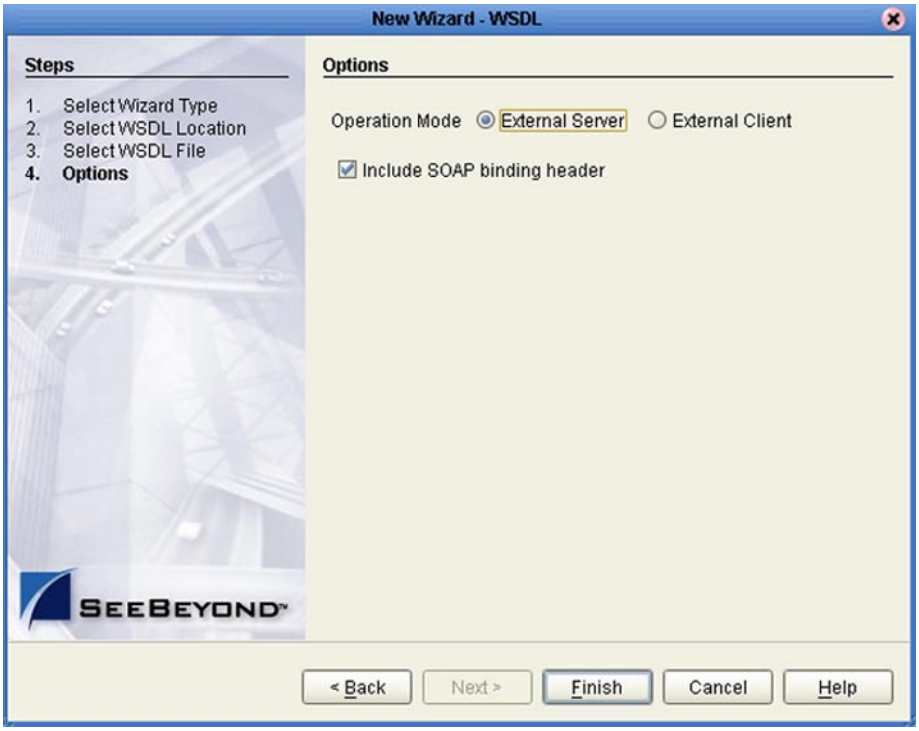
Figure 100 WSDL Wizard: Select WSDL File



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

6 Click **Next** to display the *Options* dialog, shown in Figure 101.

Figure 101 WSDL Wizard: Select OTD Options



7 Select the check boxes next to the OTD options you want to enable:

- If you are using a Web client, select **External Server**.
- If you are using a Web server, select **External Client**.
- To include the SOAP binding header in the WSDL file, select the check box.

8 Click **Finish** to add the OTD to your Project and invoke the OTD Editor.

WSDL OTD Structure

The WSDL OTD has the following basic structure:

Root Node
PortType_XXX
Operation_XXX
Input_XXX
Output_XXX
PortType_XXX
Operation_XXX
Input_XXX
Output_XXX (and so on)

Where **XXX** is the name for each element given in the original WSDL file.

WSDL Operation Elements

To tie your messages together as a request-response pair corresponding to a method call, you must define operations using the WSDL **<operation>** element. A WSDL operation identifies messages as being *input* or *output*.

Inside the WSDL file's **<operation>** element**,** you specify your **<input>** and **<output>** elements. Each element refers to the corresponding message by its fully-qualified name. The collection of all WSDL operations (that is, methods) exposed by your service is called a **portType** and is defined using the WSDL **<portType>** element.

The **<operation>** element is a child of **<portType>**. You can name the **<portType>** whatever you want. The port type **name** attribute provides a unique name among all the PortTypes defined within the enclosing WSDL file. Each WSDL operation is named via the **name** attribute.

Each operation within a WSDL OTD (like its WSDL file counterpart) uses one of the following operation modes for communication:

- **One-way:** The server receives a message from the client; also referred to as "fire and forget."
- **Request-response:** The server receives a message from the client and sends a correlated message back

6.4.3 Using the XSD Wizard

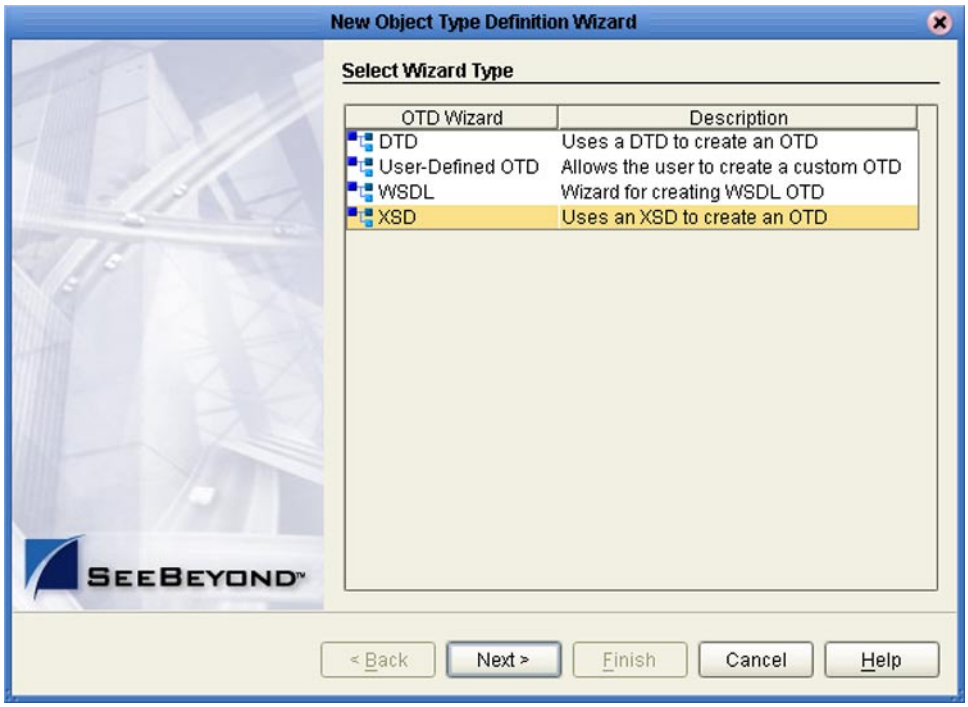
Note: *The **reset()** method resets the XSD OTD to its initial conditions, including any default values. It is highly recommended that this method be used in your Collaborations whenever marshaling in a loop, to conserve system resources.*

Note: *A newly-created XSD OTD can be re-defined by selecting the **Relaunch** option from its context menu in Enterprise Explorer (see* **Object Type Definition Menu** *on page 99**). This relaunches the XSD Wizard, and allows you to reselect files and options while preserving the original OID. Imported XSD OTDs cannot be modified.*

To create an OTD file from an XSD file

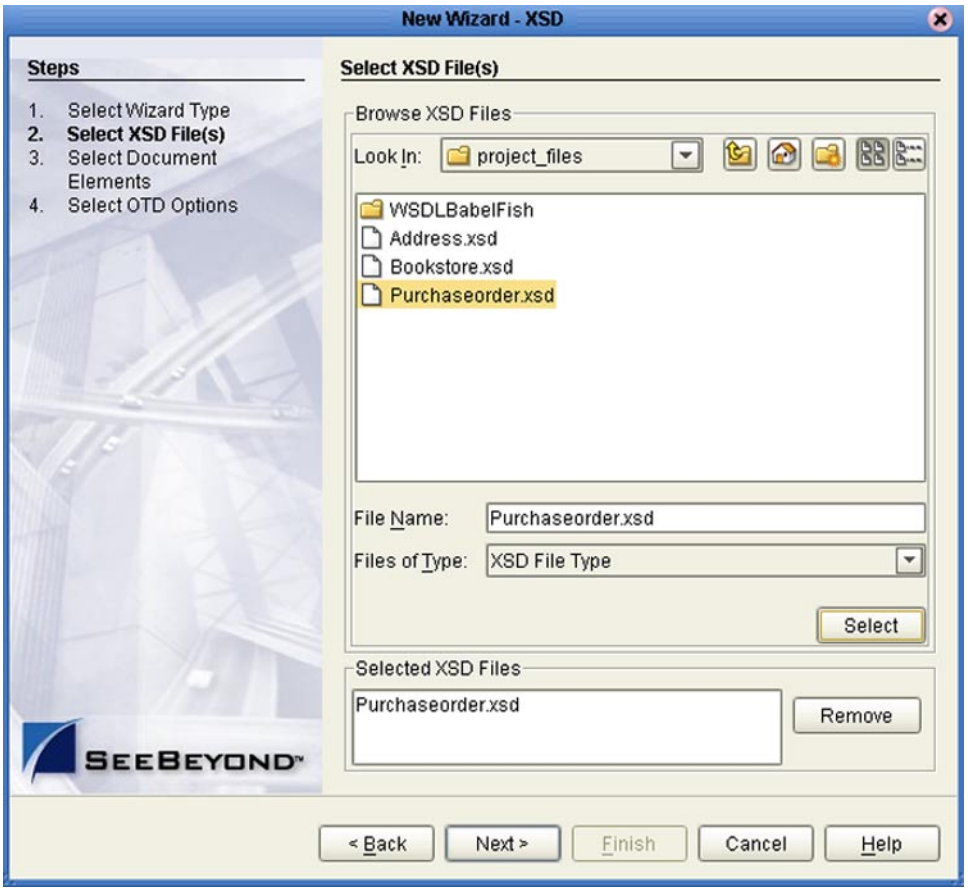
1 In the *Select Wizard Type* dialog, select **XSD** from the *OTD Wizard* list (see Figure 102) to create an OTD from an XSD file.

Figure 102 OTD Wizard Selection: XSD Wizard



2 Click **Next** to display the Select XSD File(s) dialog box, shown in Figure 103.

Figure 103 XSD Wizard: Select XSD File(s)



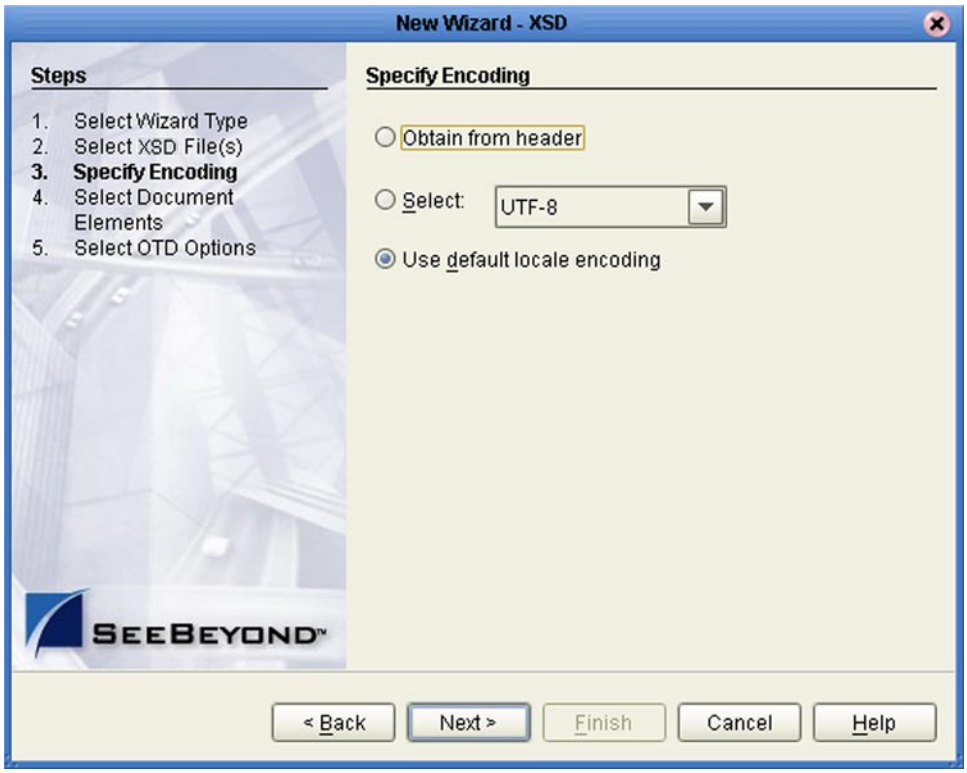
3 In the *Look In* drop-down list, navigate to the XSD file or files that you want to use to create the OTD. Click **Select** to add the files to the *List of Selected XSDs*.

- 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 104 will be displayed.

Figure 104 Cannot Create OTD Warning Box

4 If you are using the extended language options (see **Options Setup** on page 62) click **Next** to display the *Specify Encoding* dialog box, shown in Figure 105. If you are *not* using the extended language options, this Wizard dialog will not appear—proceed to step 6.

Figure 105 Select Encoding Dialog Box



5 Click the appropriate option button to specify one of the following methods for the OTD to encode the data:

- **Obtain from header** specifies the encoding from the DTD header; if the header encoding does not exist, uses the default locale encoding.
- **Select** explicitly specifies the encoding from a drop-down list, the contents of which depends upon the locale (see Table 34).

Table 34 Encoding Options

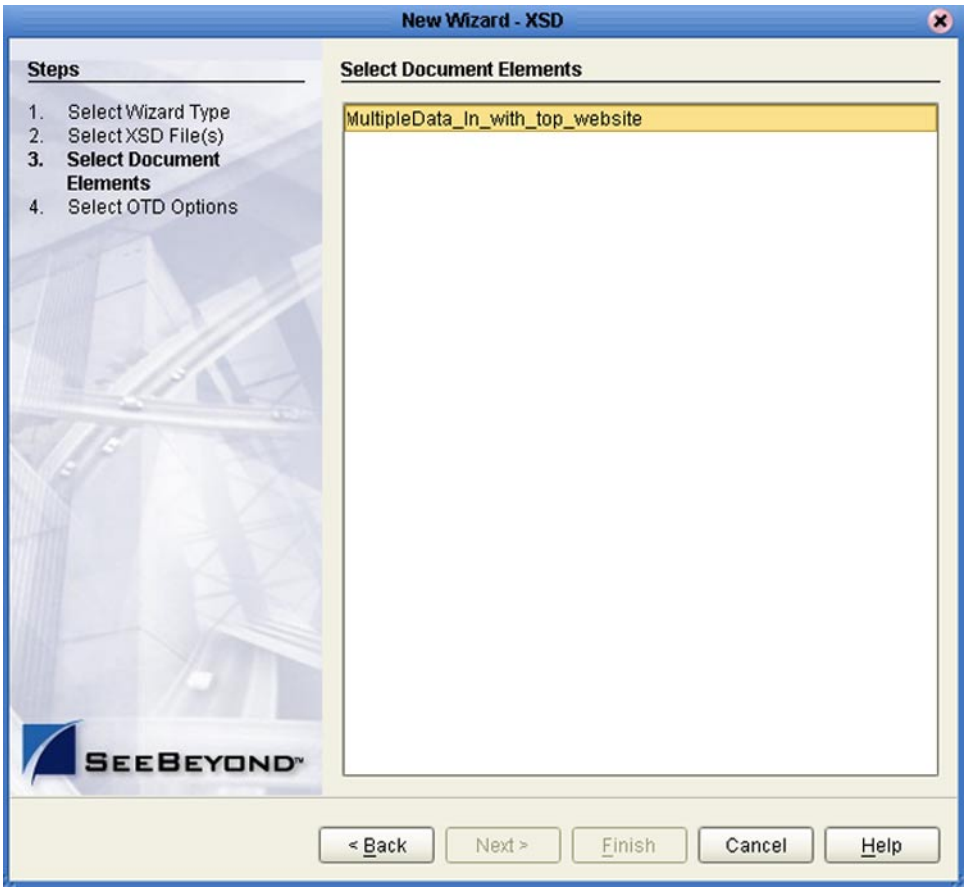
Japan	Korea
Shift JIS	
MS932	CP949
EUC-JP	EUC-KR
UTF-8	UTF-8
ASCII	ASCII

- **Use default locale encoding** uses the default encoding for the locale, based on the installed version of eGate Integrator.

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

6 Click **Next** to display the *Select Document Elements* dialog box, shown in Figure 106.

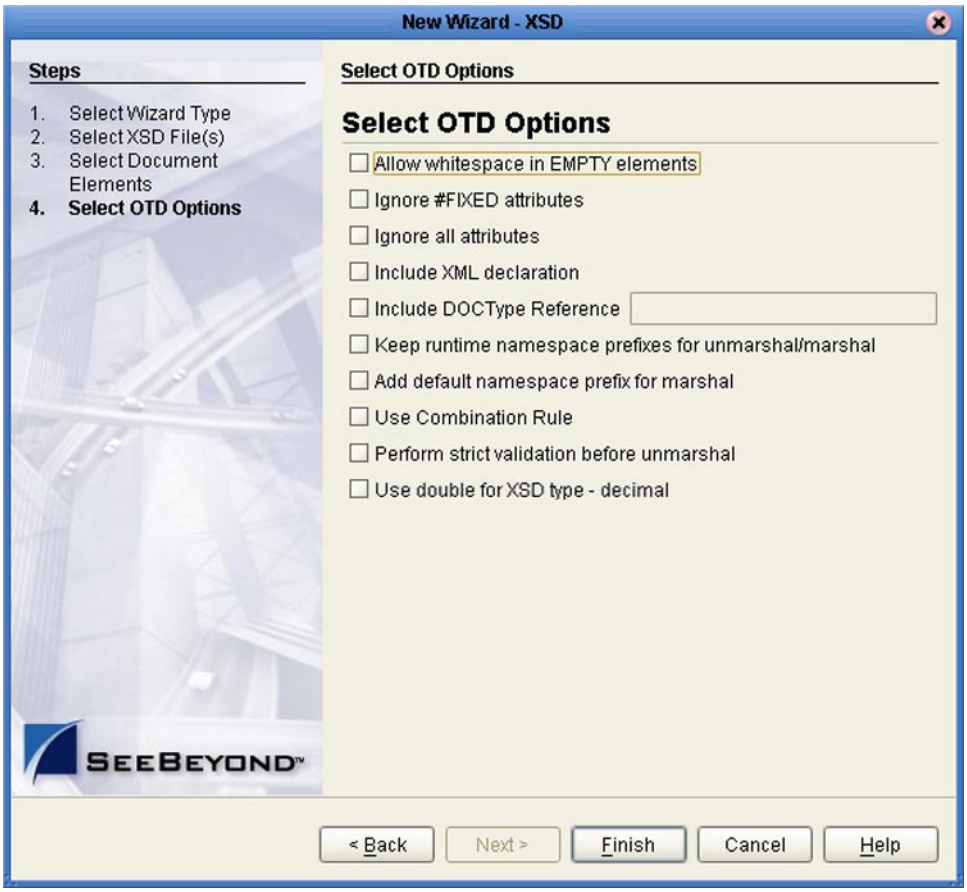
Figure 106 Select Document Elements Dialog Box



7 Select the elements of the document that you want to include in the OTD.

8 Click **Next** to display the *Select OTD Options* dialog box, shown in Figure 107.

Figure 107 Select OTD Options Dialog Box



9 Select the check boxes next to the OTD options you want to enable (see Table 35).

Table 35 XSD OTD Options

Option	Description
Allow whitespace in EMPTY elements	Not currently used for XSD OTDs.
Ignore #FIXED attributes	This option 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	This option 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	This option controls whether or not the XML declaration **<?xml version="1.0" encoding="....."?>** appears in the output during the marshal process.
Include DOC Type Reference	Not currently used for XSD OTDs.
Keep runtime namespace prefixes for unmarshal/ marshal	This option 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	This option 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.
Use Combination Rule	Not currently used.
Perform strict validation before unmarshal	Not currently used.
Use double for XSD type - decimal	If this option is checked, Enterprise Designer will use 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.

10 Click **Finish** to add the OTD to your Project and invoke the OTD Editor.

6.5 Creating User-Defined OTDs

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

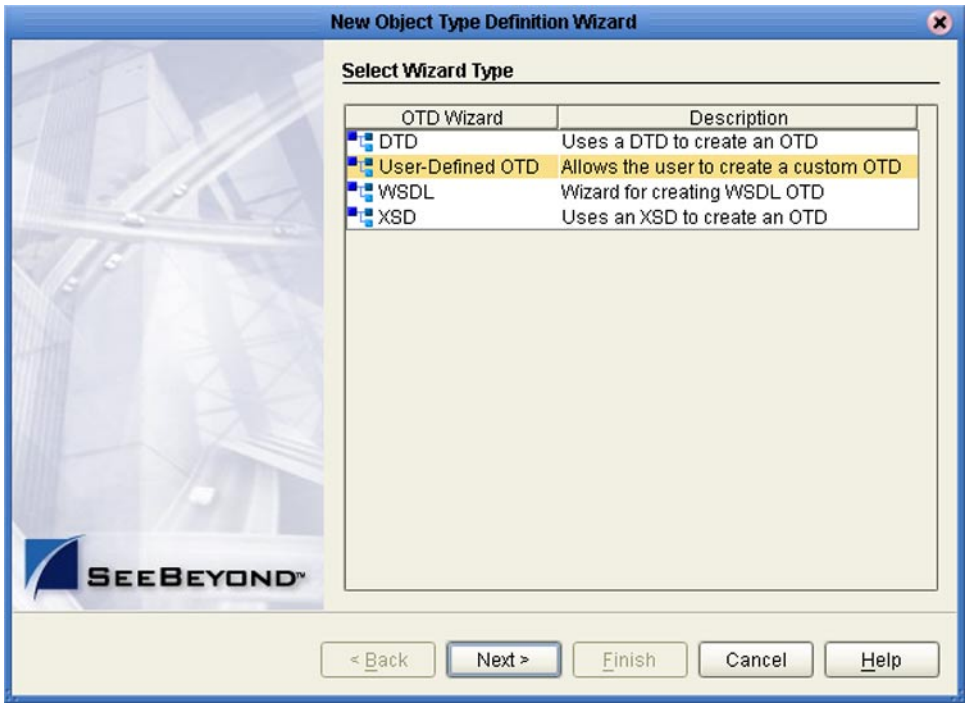
Note: *The **reset()** method resets the User-Defined OTD to its initial conditions, including any default values. It is highly recommended that this method be used in your Collaborations whenever marshaling in a loop, to conserve system resources. (Logical Hosts running on HP NonStop systems will show a dramatic improvement in performance when the **reset()** method is properly used.)*

6.5.1 Using the User-Defined OTD Wizard

To create a User-Defined OTD

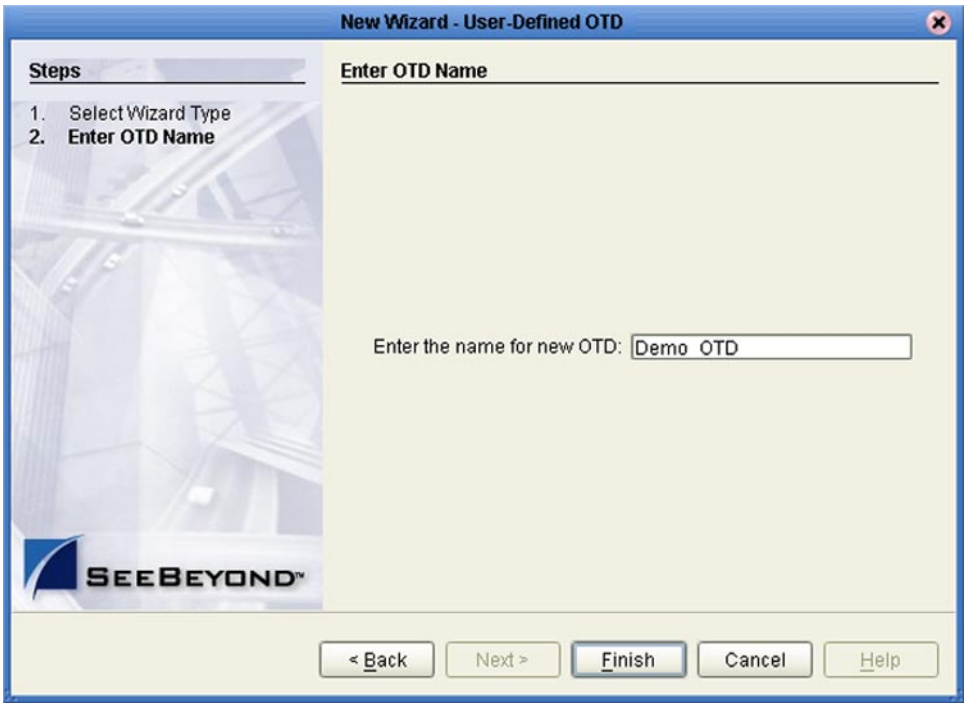
1 In the *Select Wizard Type* dialog, select **User-Defined OTD** from the *OTD Wizard* list (see Figure 108) to create an OTD without using a source file.

Figure 108 OTD Wizard Selection: User-Defined OTD



2 Click **Next** to display the *Enter OTD Name* dialog box, shown in Figure 109.

Figure 109 Enter OTD Name



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

4 The OTD Editor will open, displaying the new OTD (see **OTD Editor** on page 118). You now must specify the OTD in detail, as described in the following sections.

6.5.2 Creating and Managing Nodes

The OTD Editor allows you to:

- **Rename** a User-Defined OTD node.

  Renaming the root node renames the OTD.

- **Add** element and field nodes to a User-Defined OTD.
- **Delete** root, element, and field nodes from a User-Defined OTD.

  When a node is *deleted*, both the node and its associated 'children' are deleted.

- **Prune** root nodes in a User-Defined OTD.

  When a node is *pruned*, only its associated 'children' are deleted, while the node itself is preserved. Pruning can only be performed on root nodes.

- **Configure** the User-Defined OTD node properties.

Note: *If you move an OTD node, you must reset the* ***nodeType*** *for that node. See* **Editing OTD Properties** *on page 146.*

Creating and Modifying Elements

Once they are created, elements can be renamed or deleted, and their properties can be configured as described in **Element Node Properties** on page 148. Elements are created and placed as described below.

To add an Element to an OTD

1 Select the node of interest.

2 Right-click to invoke the node's context menu.

3 Click **Add Element**, which displays the options shown in Figure 110 and described in Table 36.

Figure 110 Add Element Options

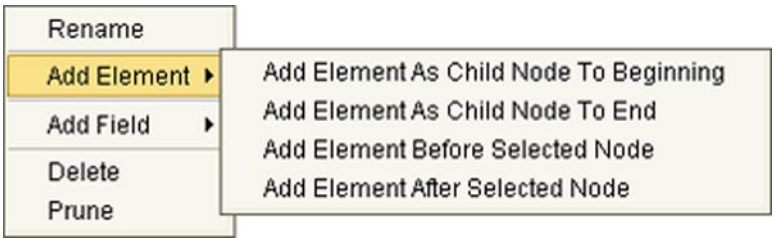
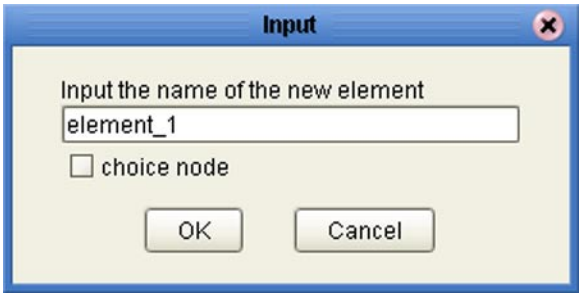


Table 36 Add Element Options

Option	Description
Add Element as a Child Node to Beginning	Adds an Element as a child of the selected node, before any other children.
Add Element as a Child Node to End	Adds an Element as a child of the selected node, after any other children.
Add Element Before Selected Node	Adds an Element at the same level as the selected node, above the selected node.
Add Element After Selected Node	Adds an Element at the same level as the selected node, below the selected node.

4 Select the desired option, which invokes the **Input** dialog box shown in Figure 111.

Figure 111 Input Dialog Box



5 Type in the name for the element.

6 If you want the element to be a *choice* node, check the box provided.

7 Click **OK** to complete the addition of the element to the OTD.

Note: *A **choice** node is either an alternate or transient Node Type. See* **Specifying the Node Type** *on page 151 for more information.*

Creating and Modifying Fields

Once they are created, fields can be renamed or deleted, and their properties can be configured as described in **Field Node Properties** on page 149. Fields are created and placed as described below.

To add a Field to an OTD

1 Select the node of interest.

2 Right-click to invoke the node's context menu.

3 Click **Add Field**, which displays the options shown in Figure 112 and described in Table 37.

Figure 112 Add Field Options

Table 37 Add Field Options

Option	Description
Add Field as a Child Node to Beginning	Adds a Field as a child of the selected node, before any other children.
Add Field as a Child Node to End	Adds a Field as a child of the selected node, after any other children.
Add Field Before Selected Node	Adds a Field at the same level as the selected node, above the selected node.
Add Field After Selected Node	Adds a Field at the same level as the selected node, below the selected node.

4 Select the desired option, which invokes an **Input** dialog box similar to that shown in Figure 111.

6.5.3 Editing OTD Properties

The hierarchical data structure of an OTD is represented at run time by a set of generated Java classes. These classes follow the Java bean rule. They have a set of zero or more properties, each of which has a specific type and a given name, and may be optional and/or repeating.

In contrast to regular Java beans, however, 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.

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

Note: *For reasons of compatibility with BPEL, OTDs cannot support a pure string-BLOB.*

Root Node Properties

The set of properties associated with the root node is shown in Figure 113.

Figure 113 User-Defined OTD Root Node Properties

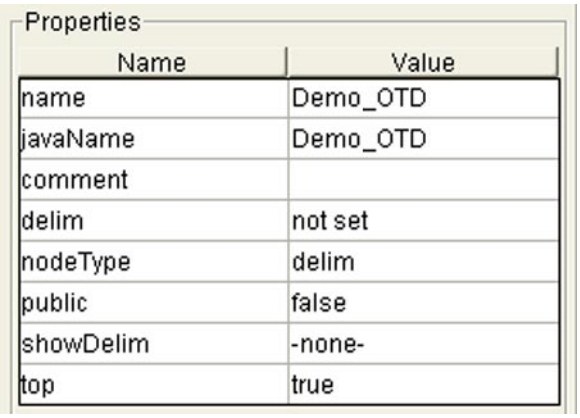


Table 38 Root Node Properties

Name	Description
name	Node display name (see **Overview** on page 114).
javaName	Property accessor basename, automatically generated—*do not modify*. See **Overview** on page 114.
comment	Free-form text (no run-time effect).
delim	Delimiter specification (see **Specifying Delimiters** on page 152). Default value is **not set**.
nodeType	Governs the marshal/unmarshal format (see **Specifying the Node Type** on page 151). The default value is **group**.
public	For future use, not currently active; **false** by default.
showDelim	Displayed only if **nodeType** is *delim*. Once delimiters are specified, value field shows the delimiters. (Root nodes are not typically delimited.)
top	Flag on root node: support marshal/unmarshal (**true** or **false**).

Element Node Properties

The set of properties associated with element nodes is shown in Figure 114.

Figure 114 User-Defined OTD Element Node Properties

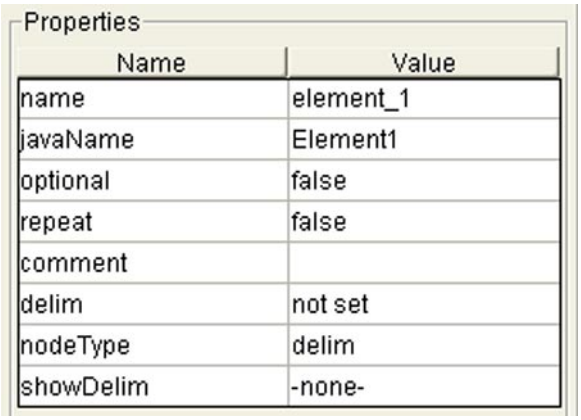


Table 39 Element Node Properties

Name	Description
name	Element display name.
javaName	Property accessor basename, automatically generated—*do not modify*.
optional	Flag: Can the element be absent? (T/F) Not applicable to root, or child of a choice Node.
repeat	Flag: Can the node appear multiple times? (T/F) Not applicable to root, or child of a choice Node.
comment	Free-form text (no run-time effect).
delim	Delimiter specification (see **Specifying Delimiters** on page 152).
nodeType	Governs the marshal/unmarshal format. The default value is **delim**, unless the element is a choice node—then the default is **trans**.
showDelim	Displayed only if **nodeType** is *delim*. Once delimiters are specified, value field shows the delimiters.

Field Node Properties

The set of properties associated with field nodes is shown in Figure 115.

Figure 115 User-Defined OTD Field Node Properties

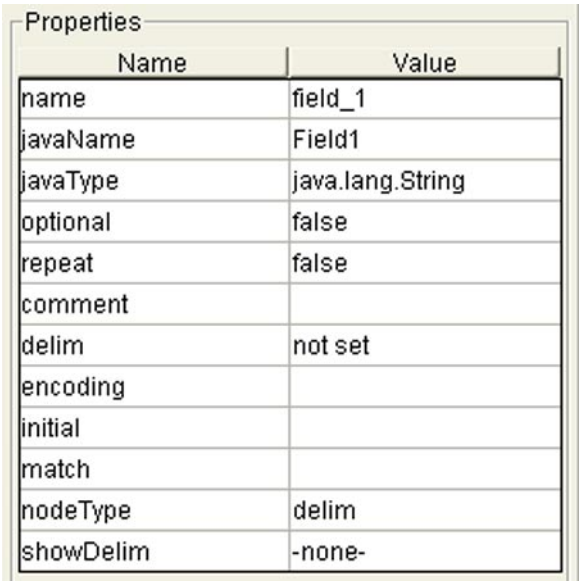


Table 40 Field Node Properties

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[]**).
optional	Flag: Can the field be absent? (T/F) Not applicable to root, or child of a choice Element.
repeat	Flag: Can the node appear multiple times? (T/F) Not applicable to root, or child of a choice Element.
comment	Free-form text (no run-time effect).
delim	Delimiter specification (see **Specifying Delimiters** on page 152).
encoding	Specifies the encoding for Unmarshaling (see Table 41).
initial	Initial field value, set when the parent node is created or reset. When provided, it is assigned to the node when node is not populated with any data.
match	If **nodeType** is *delimited* or *fixed*, performs exact match to the data.
nodeType	Governs the marshal/unmarshal format. The default value is **delim**.
showDelim	Displayed only if **nodeType** is *delim*. Once delimiters are specified, value field shows the delimiters.

Table 41 Encoding Options

Japan	Korea
Shift JIS	
MS932	CP949
EUC-JP	EUC-KR
UTF-8	UTF-8
ASCII	ASCII

6.5.4 Specifying the Node Type

Double-clicking in the **nodeType** properties field activates the field for editing. Click the arrow button to display the drop-down menu (see Figure 116 for a root node example). Descriptions of the property options are listed in Table 42.

Figure 116 Node Type Property Options (Root Node Example)

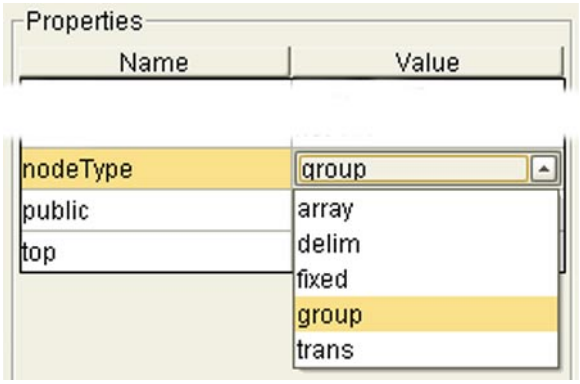


Table 42 Node Type Property Options

Option	Description	Applicable Nodes			
		Root	Element	Field	Internal
alter	Applies only to *element* nodes; option is displayed only if you have checked the *choice node* box when naming the element. Alter (*alternate*) selects one child or the other. One child is always present after the unmarshal operation.	No	Yes	No	choice
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.	Yes	Yes	Yes	simple or group
delim	Delim (*delimited*) structure. If repeated, occurrences are separated by a *normal* delimiter.	Yes	Yes	Yes	simple or group
fixed	Fixed indicates a fixed length, which is specified by non-negative integer (or zero to indicate end of parent node data).	Yes	Yes	Yes	simple or group
group	Group provides organizational grouping for purposes such as repetition. Applies only to *root* and *element* nodes, not *field* nodes.	Yes	Yes	No	group
trans	Trans (*transient*) appears only in an internal tree as a scratch pad field. It does not appear in external data representation, and can only have *trans* nodeTypes as children.	Yes	Yes	Yes	choice, simple, or group

6.5.5 Specifying Delimiters

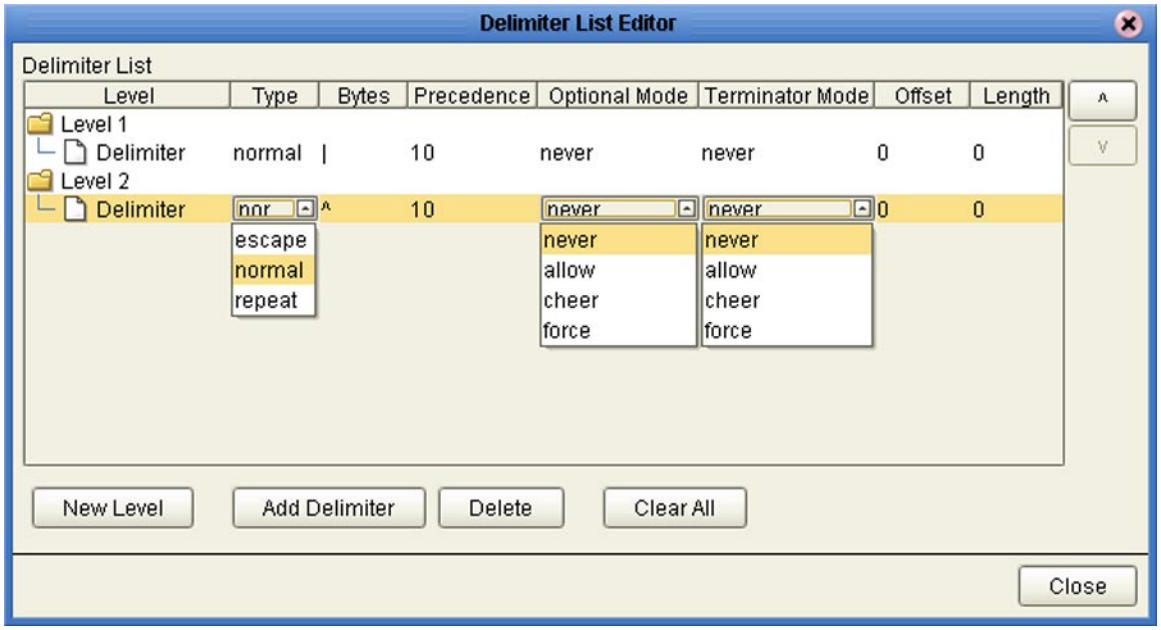
Delimiter List

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 117). 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 162.

Clicking within a field in the Delimiter List Editor enables the field for editing; double-clicking within a field displays its drop-down menu (only for the three fields shown in the figure).

Figure 117 Delimiter List Editor



The command buttons at the bottom of the Delimiter List Editor are enabled only when the action is appropriate to the selected line item.

Table 43 Delimiter List Editor Command Buttons

Command	Action
New Level	Adds a new level after the selected level.
Add Delimiter	Adds a new delimiter after the selected delimiter, or to the bottom of list under the selected level.
Delete	Deletes the selected line item (level or delimiter) from the list.
Clear All	Deletes all levels and delimiters from the list, following confirmation (displays a warning box first).
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 (v)	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 there is one).
Close	Closes the editor and saves your entries.

You can also right-click anywhere within the table area to invoke a popup menu (see Figure 118), which contains options corresponding to the buttons (except for *Close*).

Figure 118 Popup Menu

Delimiter Properties

Table 44 Delimiter Properties and Value Options

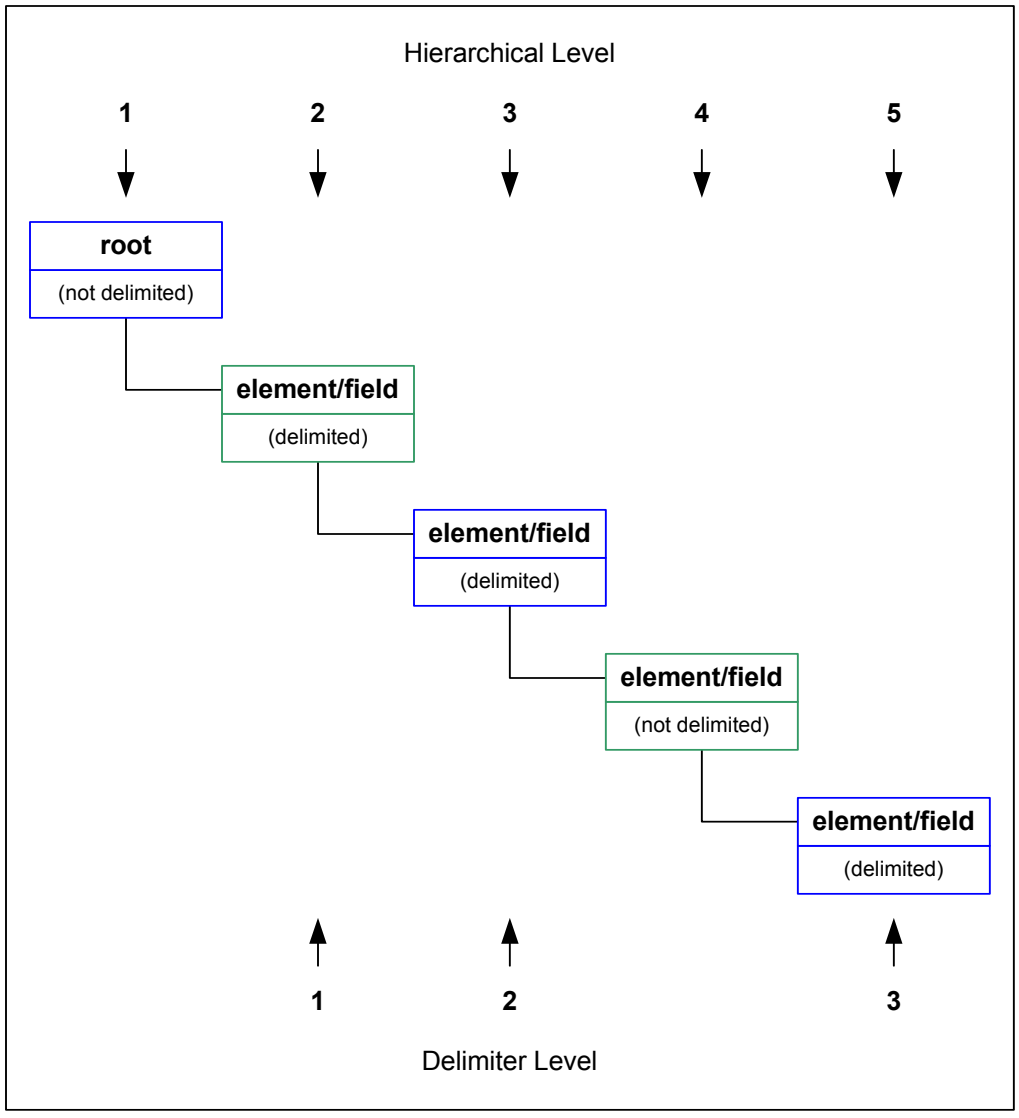
Property	Option	Description
Level		Delimiter levels, assigned consecutively to delimited nodes in the OTD node hierarchy.
Type	escape	Escape sequence.
	repeat	Array delimiter/separator.
	normal	Terminator.
Bytes		Delimiter character(s). **Note:** Entering a *Length* or *Offset* clears this field.
Precedence		See **Precedence** on page 159.
Optional Mode	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. Only the *force* option allows trailing delimiters for a sequence of absent optional nodes.
Terminator Mode	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).
Offset		Delimiter offset (non-negative integer). You can specify the delimiter by length and offset of the data field, rather than delimiter *Byte*s.
Length		Delimiter length (positive integer). You can specify the delimiter by length and offset of the data field, rather than delimiter *Byte*s.

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* applies to those nodes that are children of the root node. See Figure 119 and the following example.

Figure 119 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
  - element1
    - field1
    - field2
  - element2
    - field3
    - field4
  - field5

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 element1, element2, and field5. The *Level 2* delimiter (^) applies to field1 - field4.

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 element1, element2, and field5, and a new *Level 3* delimiter would apply to field1 - field4.

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 162 for a step-by-step description of the procedure for creating a Delimiter List.

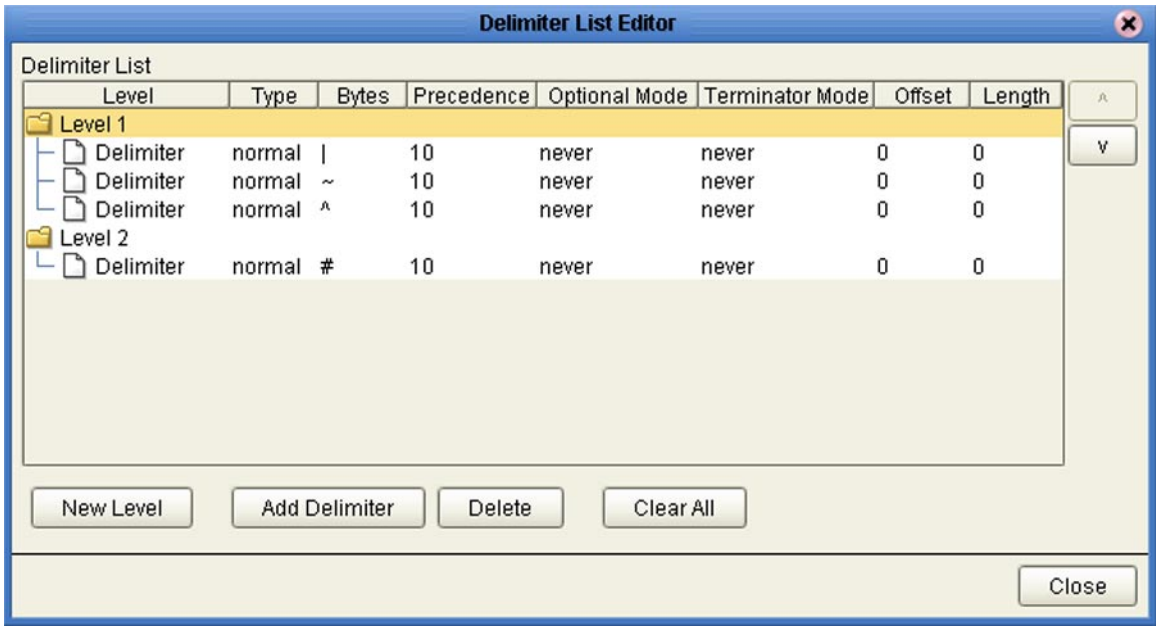
Multiple Delimiters

You can specify multiple delimiters at a given level; for example, if you specify |, ~, and ^ as delimiters for a specific level (see Figure 120), the parser will accept any of these delimiters:

- root
  - element (delimiters = "|", "~", "^")
    - field1 (delimiter = "#")
  - field2 (delimiters = "|", "~", "^")

This will successfully parse the data **abc|def**, **abc~def**, and **abc^def**.

Figure 120 Multiple Delimiter Example



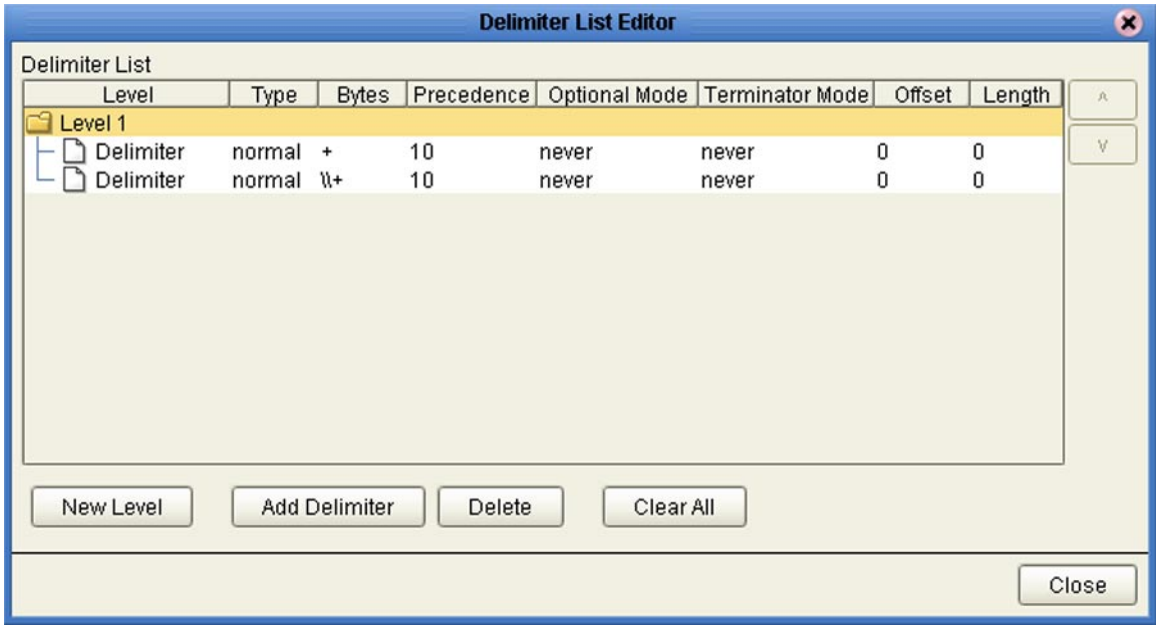
Delimiter Type

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 121), 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 121 Escape Type Delim



If there is *only* an escape delimiter on a given level, however, it presents a *no delimiter defined* situation for **delim** and **array** nodes.

Precedence

Precedence (see **Figure 131 on page 164**) indicates the priority of a certain delimiter, relative to the other delimiters. 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).

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)
    - field1 (type fixed, length = 5)
    - field2 (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**)
    - field1 (type fixed, length = 5)
    - field2 (type fixed, length = 8, optional)

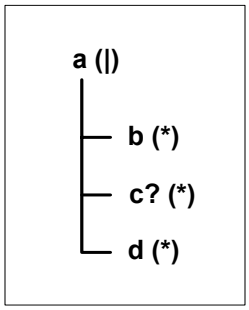
This will successfully parse the text **'abcde^zyxvuABCDEFGH'**.

Optional Mode

Consider the tree structures shown in Figure 122 and Figure 123, where the node **a** has a pipe (|) as its delimiter, and the sub-nodes **b**, **c**, and **d** all have asterisks (*) as their delimiters.

- **Example 1:** Sub-node **c** is *optional*. (Sub-node **c** and sub-node **d** must have different values for the *match* parameter.)

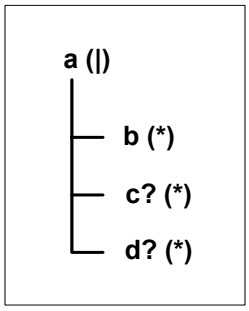
Figure 122 Optional Mode Property (Example 1)



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:** Both sub-node **c** and sub-node **d** are *optional*.

Figure 123 Optional Mode Property (Example 2)

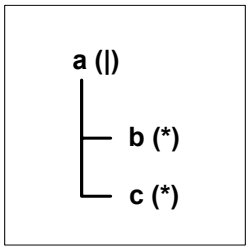


Option	Input	Output
never	**b|**	**b|**
allow	**b|**, **b*|**, or **b**|**	**b|**
cheer	**b|**, **b*|**, or **b**|**	**b**|**
force	**b**|**	**b**|**

Terminator Mode

Consider the tree structure shown in Figure 124, where the node **a** has a pipe (|) as its delimiter, the sub-nodes **b** and **c** have asterisks (*) as their delimiters.

Figure 124 Terminal Mode Property Example



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

Escape Sequences

The following escape sequences are supported in delimiters.

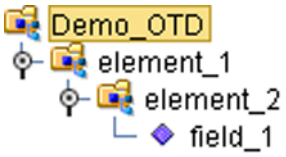
Table 45 Escape Sequences

Sequence	Description
\\	Backslash
\b	Backspace
\f	Linefeed
\n	Newline
\r	Carriage return
\t	Tab
\ddd	Octal number
\xdd	Hexadecimal number

6.5.6 Creating a Delimiter List

As an example, we shall create a delimiter list for the simple OTD structure shown in Figure 125.

Figure 125 Demo 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 **delim** property appears as *not set*, as shown in Figure 126.

Figure 126 Initial Root Node Properties

Properties

Name	Value
name	Demo_OTD
javaName	Demo_OTD
comment	
delim	not set
nodeType	group
public	false
top	true

2 Double-click in the **delim** value field to activate it for editing; an ellipsis (**...**) button appears, as shown in Figure 127. The **showDelim** property fields also appear.

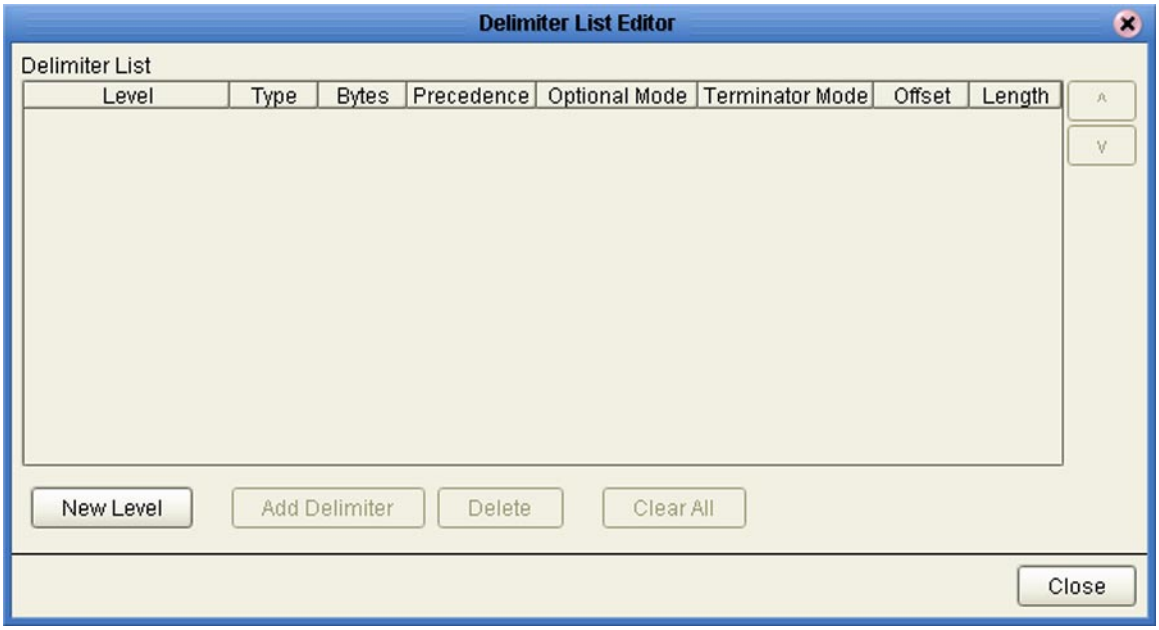
Figure 127 Activated *delim* Value Field

Properties

Name	Value
name	Demo_OTD
javaName	Demo_OTD
comment	
delim	not set ...
nodeType	group
public	false
top	true

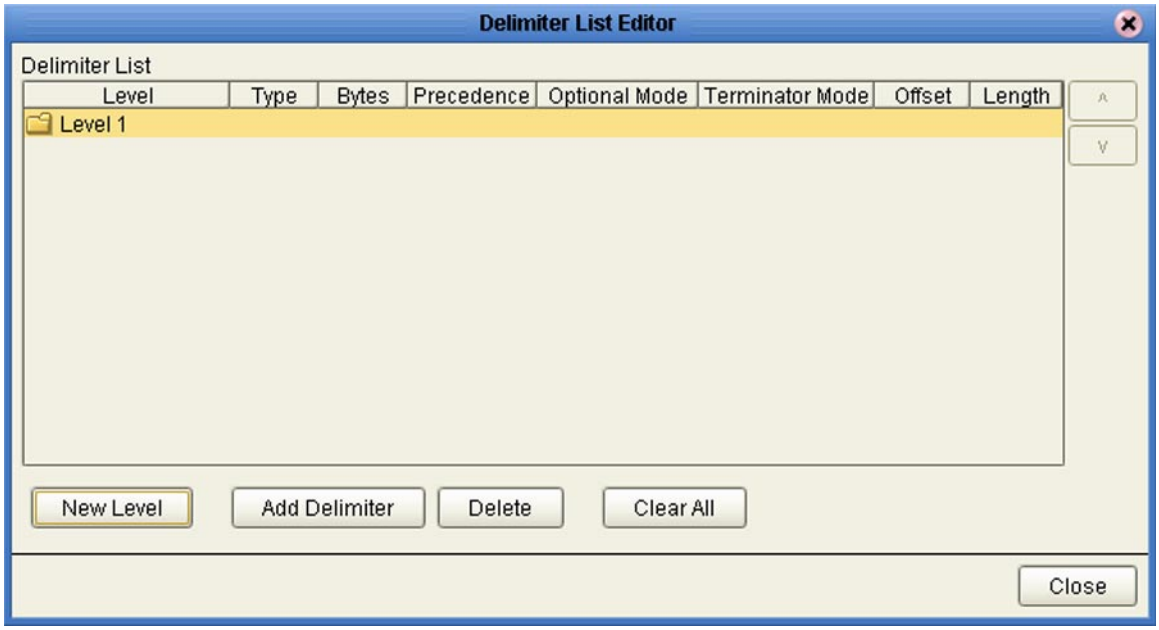
3 Click the ellipsis button to display the Delimiter List Editor, which is initially blank (see Figure 128).

Figure 128 Delimiter List Editor - Delimiters Not Set



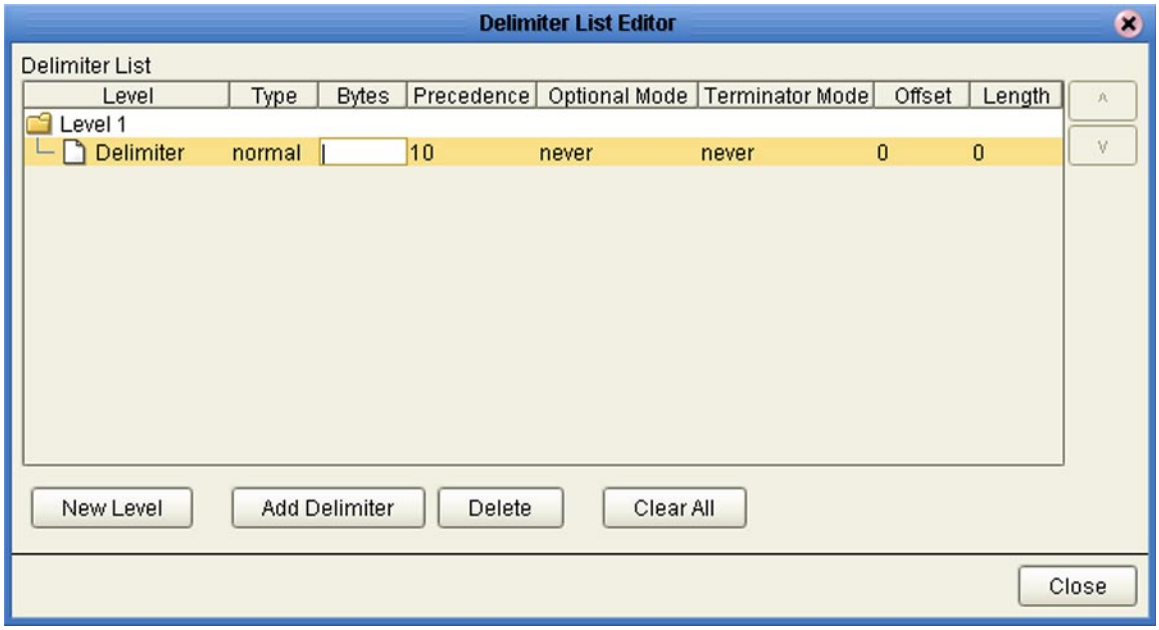
4 Click **New Level** to add a level to the delimiter list, as shown in Figure 129.

Figure 129 Delimiter List Editor - Insert Level



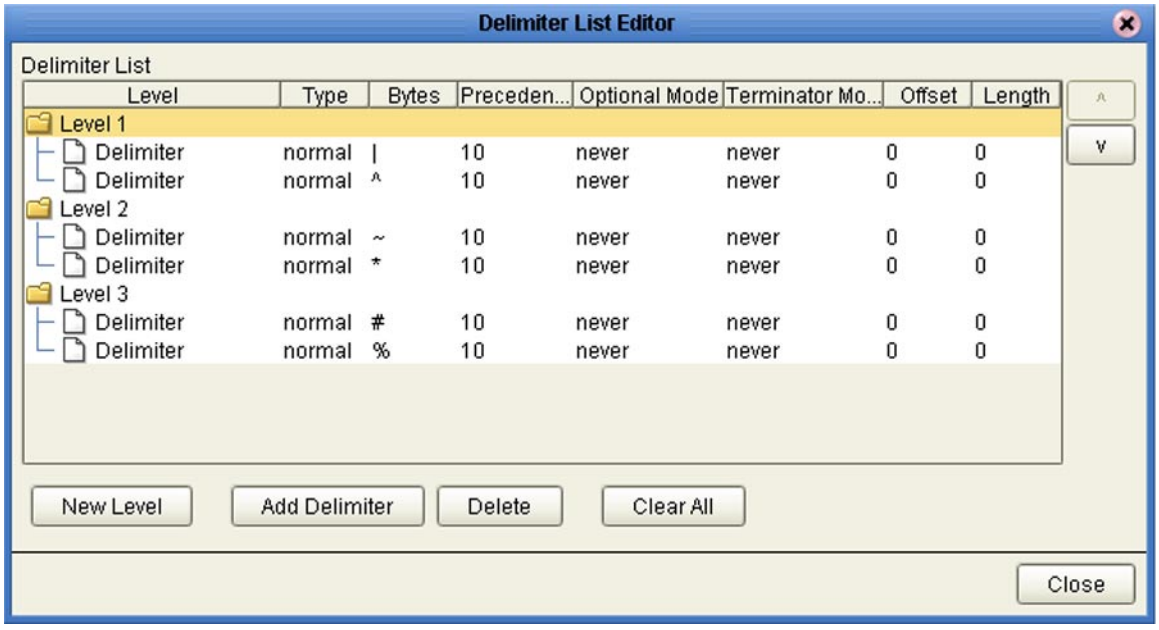
5 Select a level and click **Add Delimiter** to add a delimiter to the selected level. Double-click in the Bytes field and type in the delimiter characters (see Figure 130).

Figure 130 Delimiter List Editor - Add Delimiter



6 Continue adding levels and delimiters as required (see Figure 131).

Figure 131 Delimiter List Editor - Add Levels and Delimiters



7 Click **Close** to close the editor and save your work.

8 The value for the **delim** property will now appear as *specified*, as shown in Figure 132.

Figure 132 Root Node - Delimiter Specified

Properties

Name	Value
name	Demo_OTD
javaName	Demo_OTD
comment	
delim	specified
nodeType	group
public	false
top	true

9 Since the **nodeType** values for elements and fields are *delim* by default, they will automatically display the delimiters for their respective levels. To exclude any of them from the effect of the delimiter list, you must reset their **nodeType** properties to a value other than *delim*.

A The delimiters for *Element_1* are now displayed as shown in Figure 133—compare the delimiters with the list shown in Figure 131. Note that the **delim** property remains *not set*, since the delimiter list is defined for the *root* node. Defining another delimiter list here will override the original list.

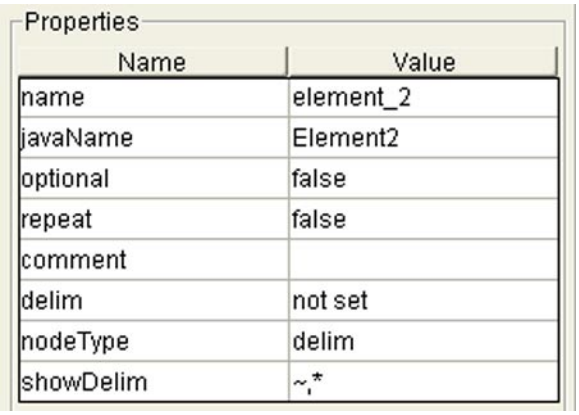
Figure 133 Element_1 Node Properties

Properties

Name	Value
name	element_1
javaName	Element1
optional	false
repeat	false
comment	
delim	not set
nodeType	delim
showDelim	|,^

B The delimiters for *Element_2* are now displayed as shown in Figure 134—compare the delimiters with the list shown in Figure 131. Again, the **delim** property remains *not set*.

Figure 134 Element_2 Node Properties



C The delimiters for *Field_1* are now displayed as shown in Figure 135— compare the delimiters with the list shown in Figure 131. Again, the **delim** property remains *not set*.

Figure 135 Field_1 Node Properties

10 Once you have defined your delimiter list, you should test the OTD to verify that it parses correctly (see **Using the OTD Tester** on page 123).

6.6 OTD Libraries

Several libraries containing large numbers of OTDs for specific purposes are available to use with eGate Integrator. These OTDs provide templates corresponding to standard message types defined by various organizations. The templates are pre-defined and can be used as-is or modified using the OTD Editor, whichever is appropriate to your application.

6.6.1 CBO OTD Library

The Canonical Business Object (CBO) OTD Library corresponds to the Open Applications Group's Integration Specification (OAGIS), and is packaged with eGate Integrator. The library is contained in the archive file **CBO_OTD_v1_0.sar**, which can be downloaded from the installation CD-ROM.

OAGIS provides a horizontal business language that provides an open interface for interaction with vertical industry standards. Extensive information is available from the Open Applications Group's Web site **http://www.openapplications.org.**

6.6.2 Add-on OTD Libraries

Several other OTD libraries are available from SeeBeyond as add-ons to eGate Integrator. These include libraries for SWIFT and HL7.

Chapter 7

Collaboration Definitions (Java)

This chapter describes the process for building Java-based Collaboration Definitions.

7.1 Overview

Collaborations use Collaboration Definitions to define how data should be processed and routed between Project components. Collaborations also define how databases should be queried in response to requests and how APIs to one or more applications should be invoked. Collaborations are used when data translation is required.

The Enterprise Designer includes two tools, the Collaboration Definition Wizard (Java) and Collaboration Editor (Java), that are used to create and customize your Java-based Collaboration Definitions. You must have OTDs available to use as the foundation for creating a Java-based Collaboration Definition. See **Object Type Definitions** on page 114 for more details.

Important: *If you delete an OTD in the Project Explorer, any Java-based 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* **Impact Analyzer** *on page 81**).*

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.*

7.2 Using the Collaboration Definition Wizard (Java)

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). The user interface is highly self-explanatory, but details of the navigation buttons are listed in Table 46 for your reference.

Table 46 Wizard Navigation Buttons

Button	Function
< Back	Returns to the previous step in the wizard. This button is disabled on the first step.
Next >	Goes to the next step in the wizard. This button is disabled on the last step.
Finish	Saves all Collaboration Definition settings and closes the wizard. This button is only enabled on the last step.
Cancel	Closes the wizard without saving the Collaboration Definition.
Help	Displays the online help documentation for the Collaboration Definition Wizard dialog box.
Add	Adds a selected Object Type Definition to the Collaboration Definition.
Remove	Removes a selected Object Type Definition from the Collaboration Definition.

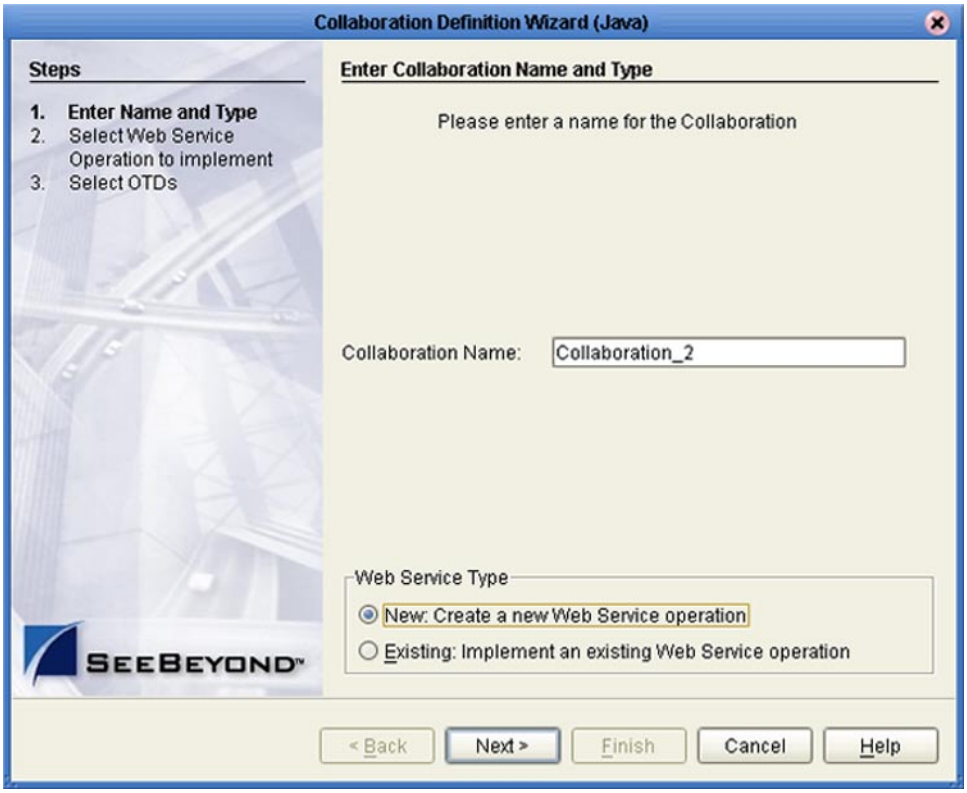
7.2.1 Creating a Java-based Collaboration Definition

1 Right-click on a Project in the Enterprise Explorer to display the Project context menu.

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

3 Enter a **Name** for your Collaboration, as shown in Figure 136.

Figure 136 Initial Wizard Dialog



4 Select a Web service, which can be either:

- A New Web Service.
- An Existing Web Service (for example, an eInsight process or an OTD).

5 Click **Next** to proceed to the next Wizard dialog, which is dependent upon your Web Service selection.

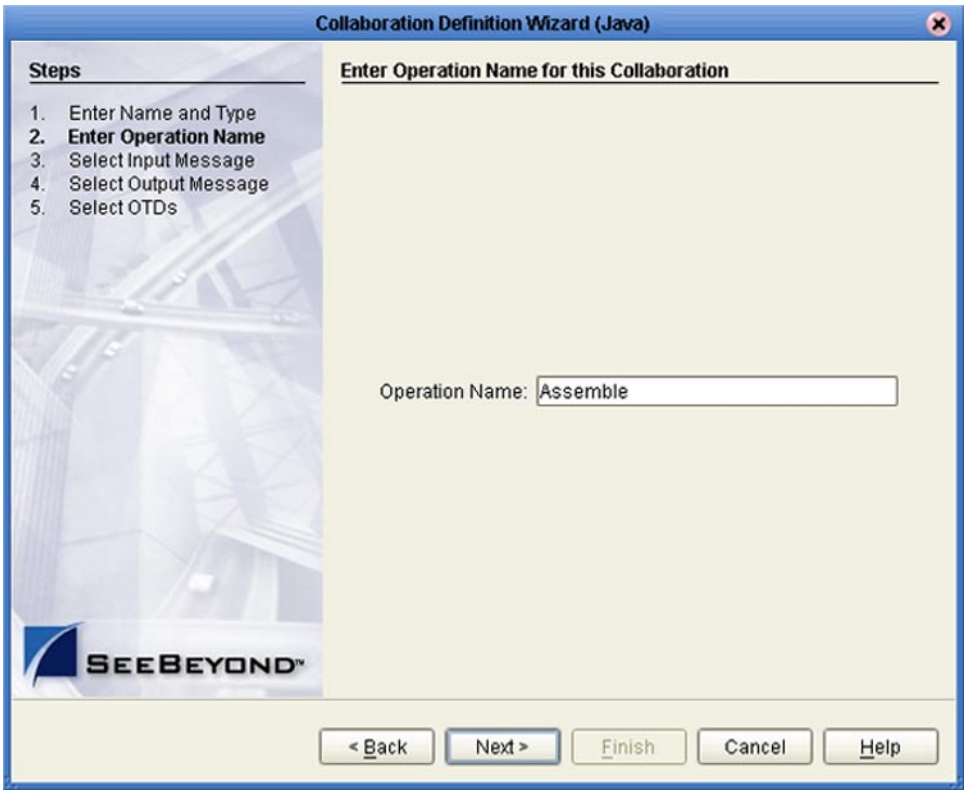
Note: *Selecting New Web Service enables the Collaboration Tester (see* **Collaboration Tester** *on page 186).*

New Web Service

If you selected a New Web Service, you will be presented with the following set of Wizard dialogs.

1 Enter an operation name, as shown in Figure 137. This will become the *method* that can be used to invoke the Java-based Collaboration as a Web service.

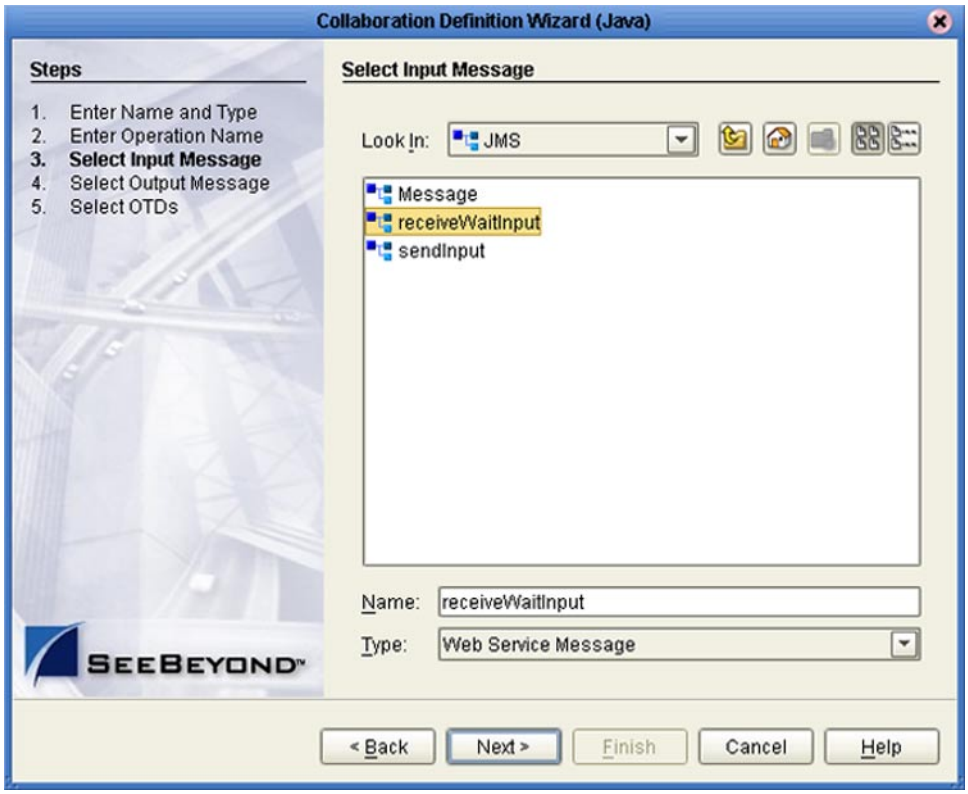
Figure 137 New Web Service: Operation Name



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

3 Select the input Web service message, as shown in Figure 138.

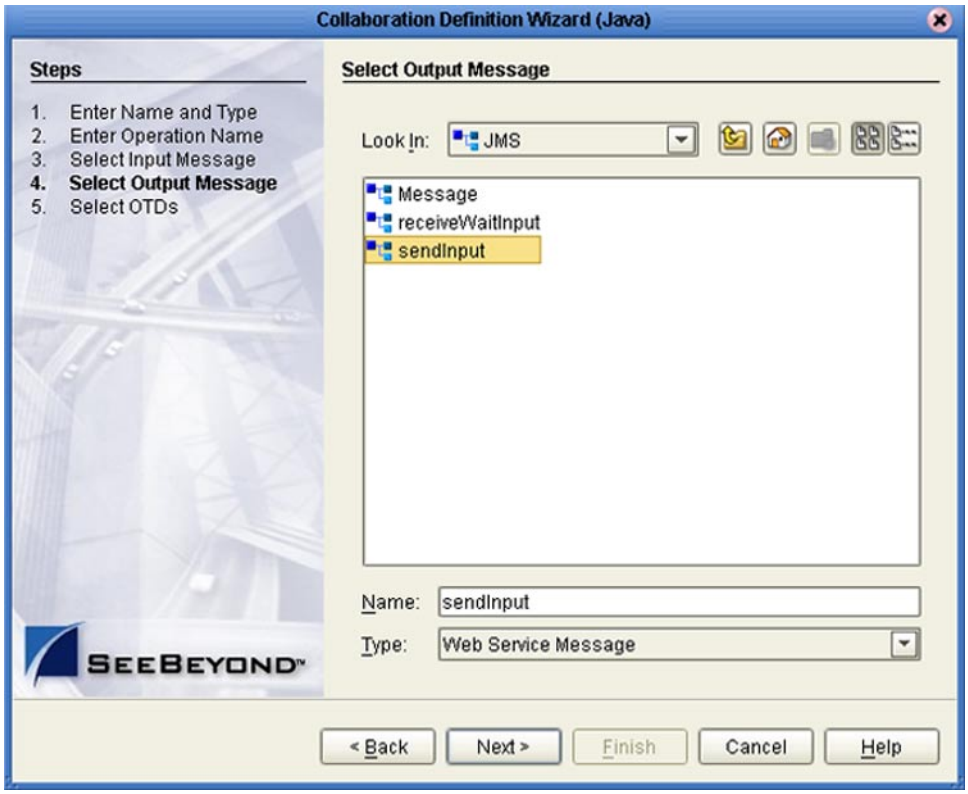
Figure 138 New Web Service: Input Message



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

5 Select the output Web service message, as shown in Figure 139.

Figure 139 New Web Service: Output Message

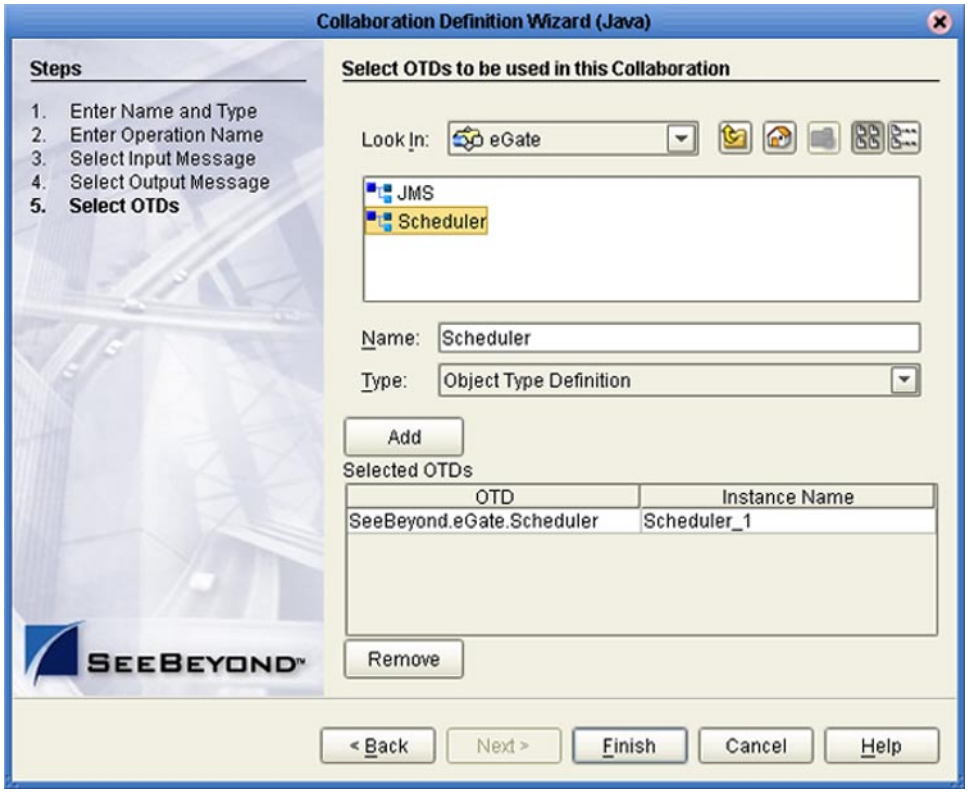


6 Click **Next** to proceed to the next Wizard dialog.

7 Select an auxiliary OTD, as shown in Figure 140. This step is optional, and is intended to support additional functionality such as a database lookup.

Note: *Use caution here, since you may already have OTDs selected in the preceding steps.*

Figure 140 New Web Service: Auxiliary OTD



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

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

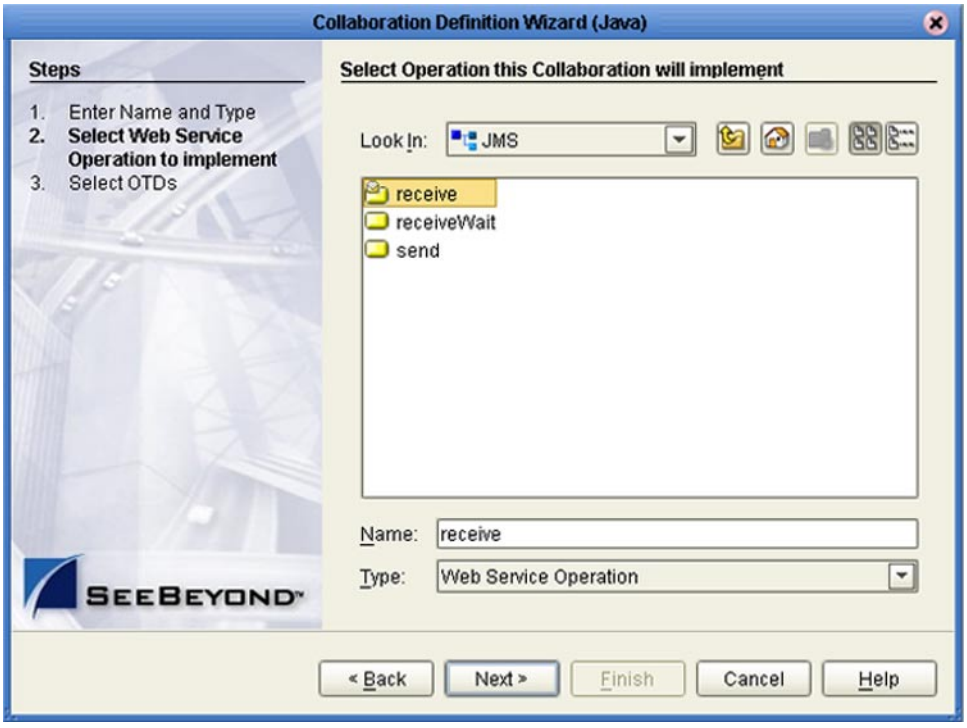
Existing Web Service

If you selected an Existing Web Service, you will be presented with the following set of Wizard dialogs.

1 Select a Web service operation, which can be either:

- An installed ICAN Web Service (for example, a JMS *receive* Web service - see Figure 141).
- A custom Web Service (for example, something that has been created in an eGate Project).

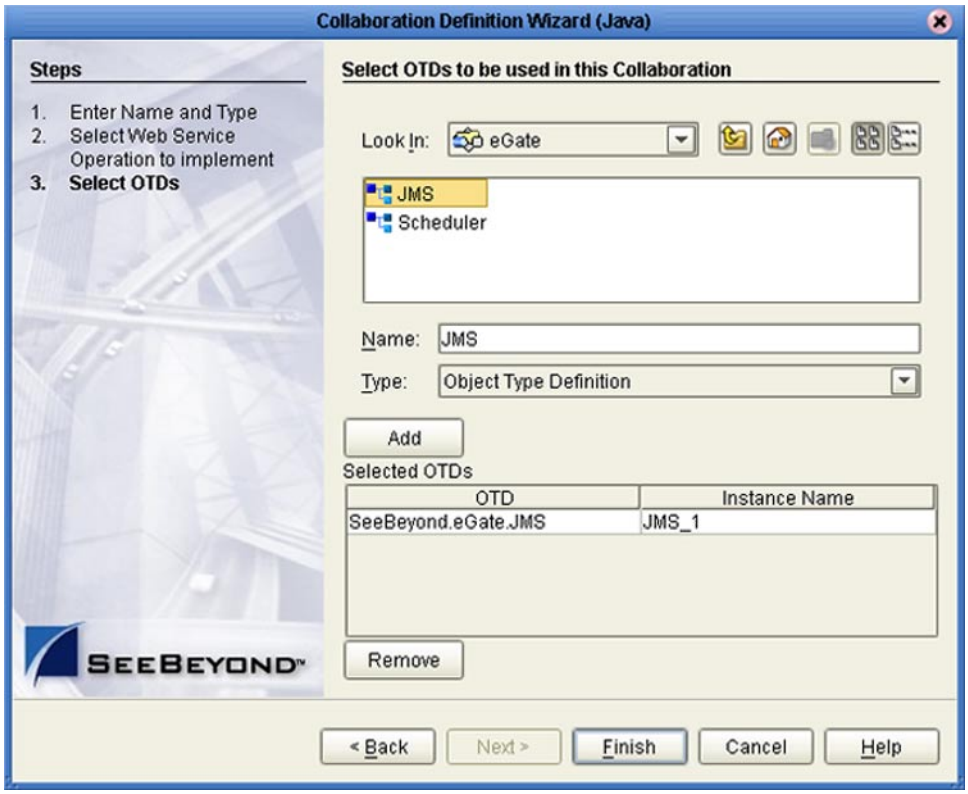
Figure 141 Existing Web Service: Select Operation



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

3 Select an OTD, as shown in Figure 142.

Figure 142 Existing Web Service: Select OTD



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

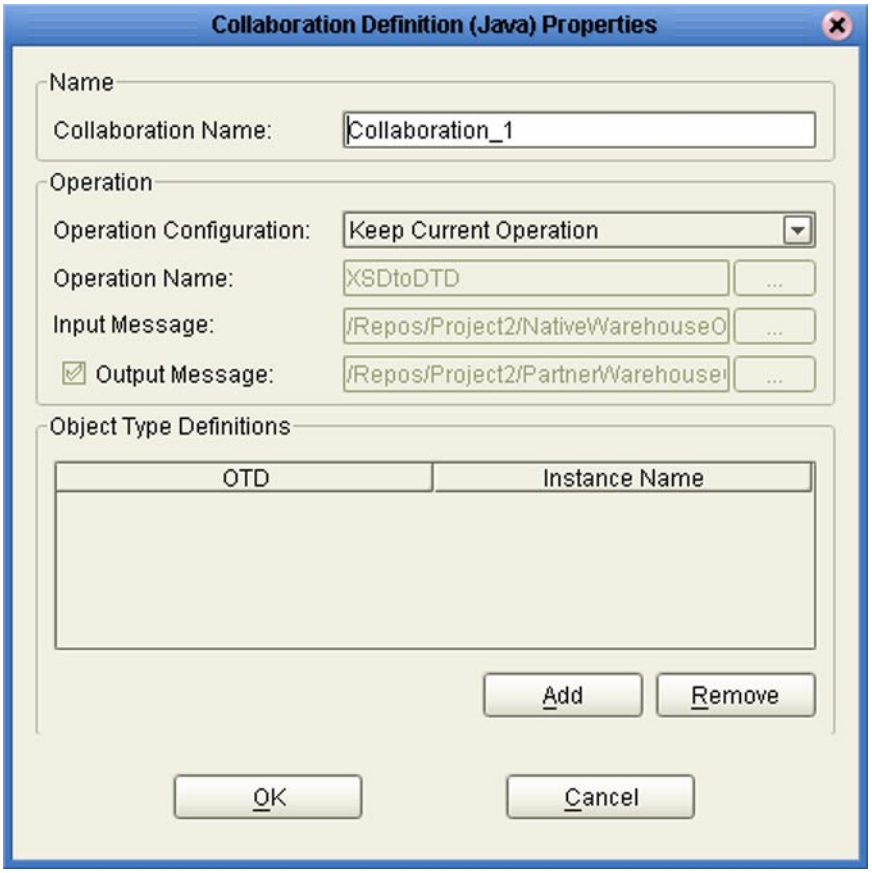
5 Click **Next** to proceed to the Collaboration Editor (Java).

7.3 Editing Collaboration Definition Properties

Right-clicking a Collaboration Definition (Java) in the Project Explorer displays the context menu shown in Figure 69 on page 100. Selecting **Properties** from the menu displays the Collaboration Definition (Java) Properties dialog box for the selected Collaboration Definition (see Figure 143).

This dialog box resembles 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.

Figure 143 Collaboration Definition (Java) Properties



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 box, as described in the following procedure.*

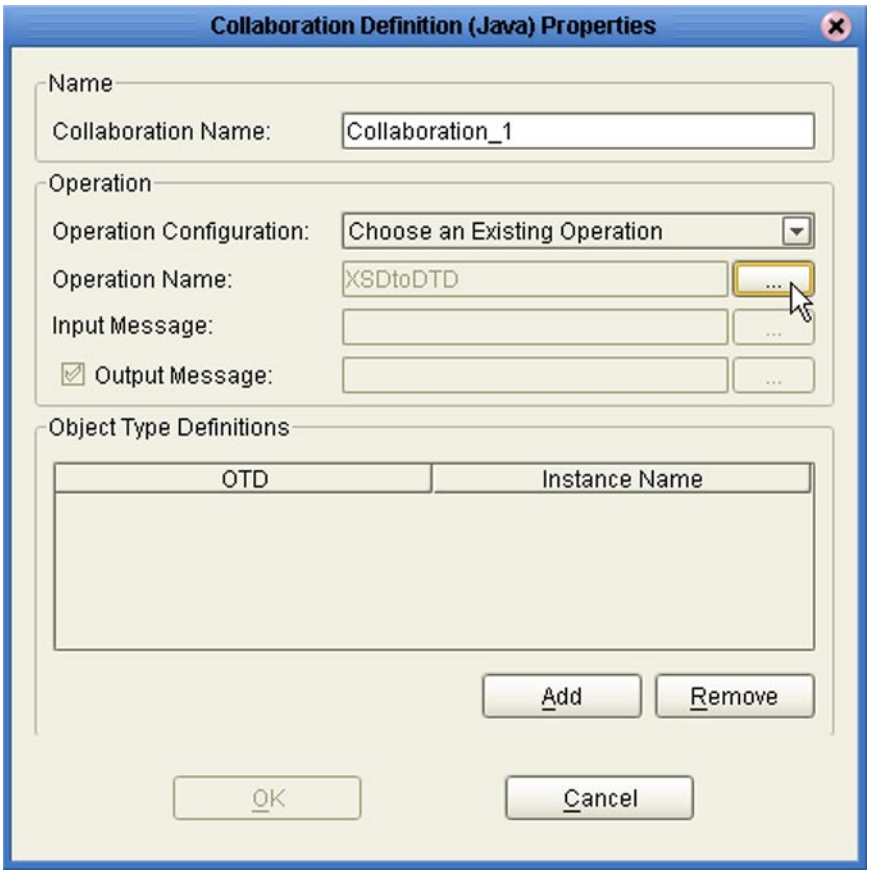
To reset the configuration properties

1 In the Explorer, right-click the Collaboration Definition that has been changed.

2 Select **Properties** from the Collaboration Definition context menu (see **Figure 69 on page 100**).

3 Select **Choose an Existing Operation** from the Operation Configuration list (see Figure 144).

Figure 144 Collaboration Definition (Java) Properties



4 Select the same Web Service Operation as before in **Operation Name** (click the ellipsis **[...]** button, as shown in Figure 144, to display a dialog box for selecting the operation).

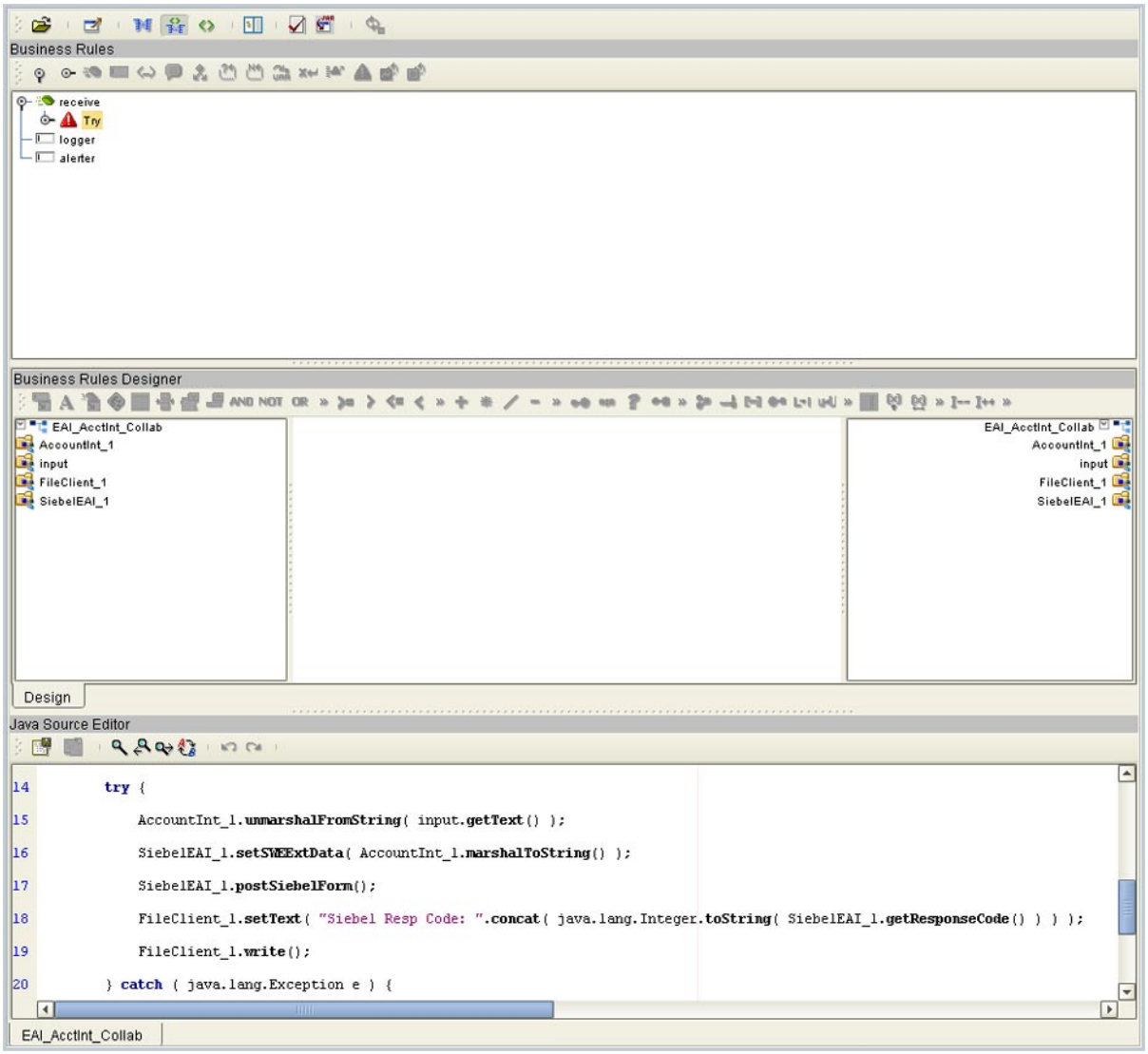
5 Select the appropriate input and output messages.

6 Click **OK** to finish.

7.4 Using the Collaboration Editor (Java)

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 145).

Figure 145 Collaboration Editor (Java)



The three parts of the Collaboration Editor are described in the following sections:

- **Business Rules Editor** on page 181.
- **Business Rules Designer** on page 187.
- **Java Source Editor** on page 184.

Important: *The Collaboration Editor (Java) does not support synchronized blocks.*

7.4.1 Java Toolbar Icons

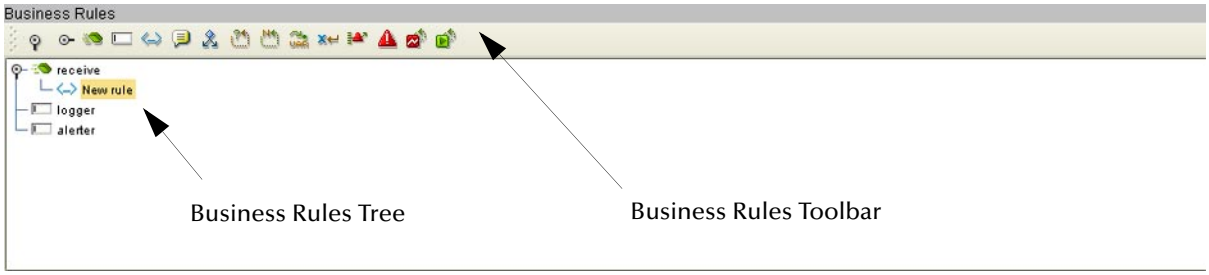
Table 47 Java Toolbar Icons

Icon	Command	Function
	Import a Local File	Displays the **Open** dialog box, which you can use to locate and select a Collaboration Definition (Java) 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.
	Export Java Rule to a Local File	Displays the **Save** dialog box, which you can use to save the selected Collaboration Definition (Java) to a file.
	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 and Java Source Editors only.
	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.
	Import JAR File	Displays the **Add/Remove JAR Files** dialog box, which you can use to import a .jar file.
	Refresh Collaboration	Refreshes the Collaborations from the Repository.

7.4.2 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 146). You add rules simply by dragging and dropping the rules into the Business Rules tree. Click the **Commit Changes** button when you are finished.

Figure 146 Business Rules Editor



Note: *The Enterprise Designer automatically disables the Business Rules Editor when you are actively using the Java Source Editor. This is a safety feature that prevents you from inadvertently making contradictory changes. To re-enable the Business Rules Editor, click the* ***Commit Changes*** *button in the main toolbar.*

Commands

The **Method** command is used 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).

The **Field** command is used when you want to create a field 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 in the Collaboration are able to read or write from it. A field can be used with the Collaboration (access type = local) or by any other Collaboration (access type = public).

The remaining commands, as represented in the toolbar, are explained in Table 48.

Business Rules Toolbar Icons

Table 48 Business Rules Toolbar Buttons

Icon	Command	Function
	Expand All Rules	Expands the tree to show all rules.
	Collapse All Rules	Collapses all rules in the tree.
	Method	Displays the *Add Method* dialog box, which you can use to add a new business rule method to the Business Rules tree. The Business Rules tree includes a default method called **executeBusinessRules**.
	Field	Displays the *Create a Parameter* dialog box, which you can use to add a new field to the Business Rules tree.
	Rule	Adds a new (empty) **rule** statement to the Business Rules tree. Use this command as a safeguard regarding positioning.
	Comment	Displays a dialog box in which you can enter a comment.
	If-Then	Adds an **if-then** statement to the Business Rules tree.
	While	Adds a **while** statement to the Business Rules tree, starting a specific iteration (repetition) of business rules. You can configure the condition using drag-and-drop when the **while** statement is selected.
	For	Adds a **for** statement to the Business Rules tree, starting a specific iteration (repetition) of business rules.
	Local Variable	Displays the *Create a Variable* dialog box, which you can use to add a local variable to the Business Rules tree.
	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. A **finally** statement is added automatically; you must right-click on the rule to add a **catch** statement (see **Using Try-Catch** on page 217).

Table 48 Business Rules Toolbar Buttons

Icon	Command	Function
	Break	Breaks out of a business rule **while**, **for**, or **switch** loop.
	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. The rule, if-then, while, for, local variable, return, and try statements can be added to a method as subnodes. Statements cannot be added to fields.

The rules for placing subnodes in a method are described in Table 49.

Table 49 Rules for Placement of Subnodes

Subnode	Description
if	Place as a sibling after the last rule in the **if-then-else** block.
then	Place as a child after the last rule in the **then** code block.
else	Place as a child after the last rule in the **else** code block.
while	Place as a sibling after the last rule in the **while** node, just after the closing parenthesis.
while (code block)	The target node is represented by a code block node under the **while** node. The source rule becomes a child as the last rule in the block.
for	Place as a sibling after the last rule in the **for** node, just after the closing parenthesis.
for (code block)	The target node is represented by a code block node under the **for** node. The source rule becomes a child as the last rule in the block.
local variable	Place as a sibling after the last **local variable** block.
return	Place as a sibling after the last **return** block.
try	Place as a sibling after the last rule in the **try-catch-finally** block.
catch	Place as a child after the last rule in the **catch** code block.
finally	Place as a child after the last rule in the **finally** code block.

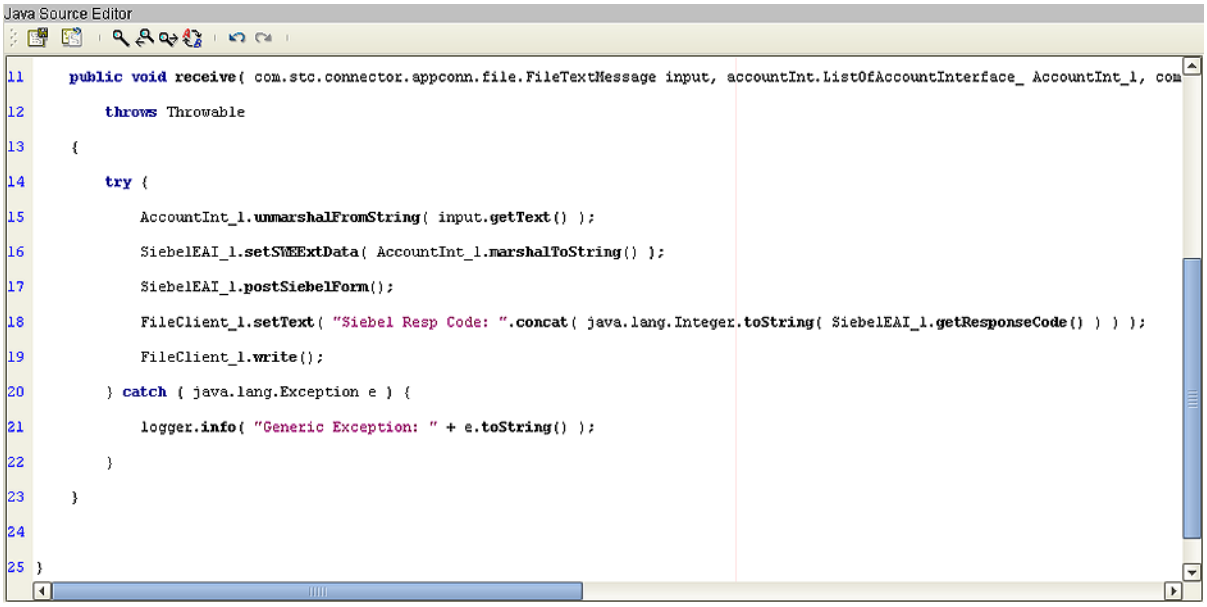
Note: *When a subnode is moved to another method, or node, it becomes the first subnode listed under that node.*

7.4.3 Java Source Editor

The Java Source Editor (see Figure 147) displays the actual Java code as you map the Collaboration Definition. 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: *The Java Source Editor is not displayed by default—you must click the Source Code Mode or Advanced Mode toolbar buttons to display it.*

Figure 147 Java Source Editor



Note: *The Enterprise Designer automatically disables the Business Rules Editor and the Business Rules Designer when you are actively using the Java Source Editor. This is a safety feature that prevents you from inadvertently making contradictory changes. You must click the* ***Commit Changes*** *button in the toolbar to re-enable the Business Rules Editor and the Business Rules Designer.*

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 50 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.

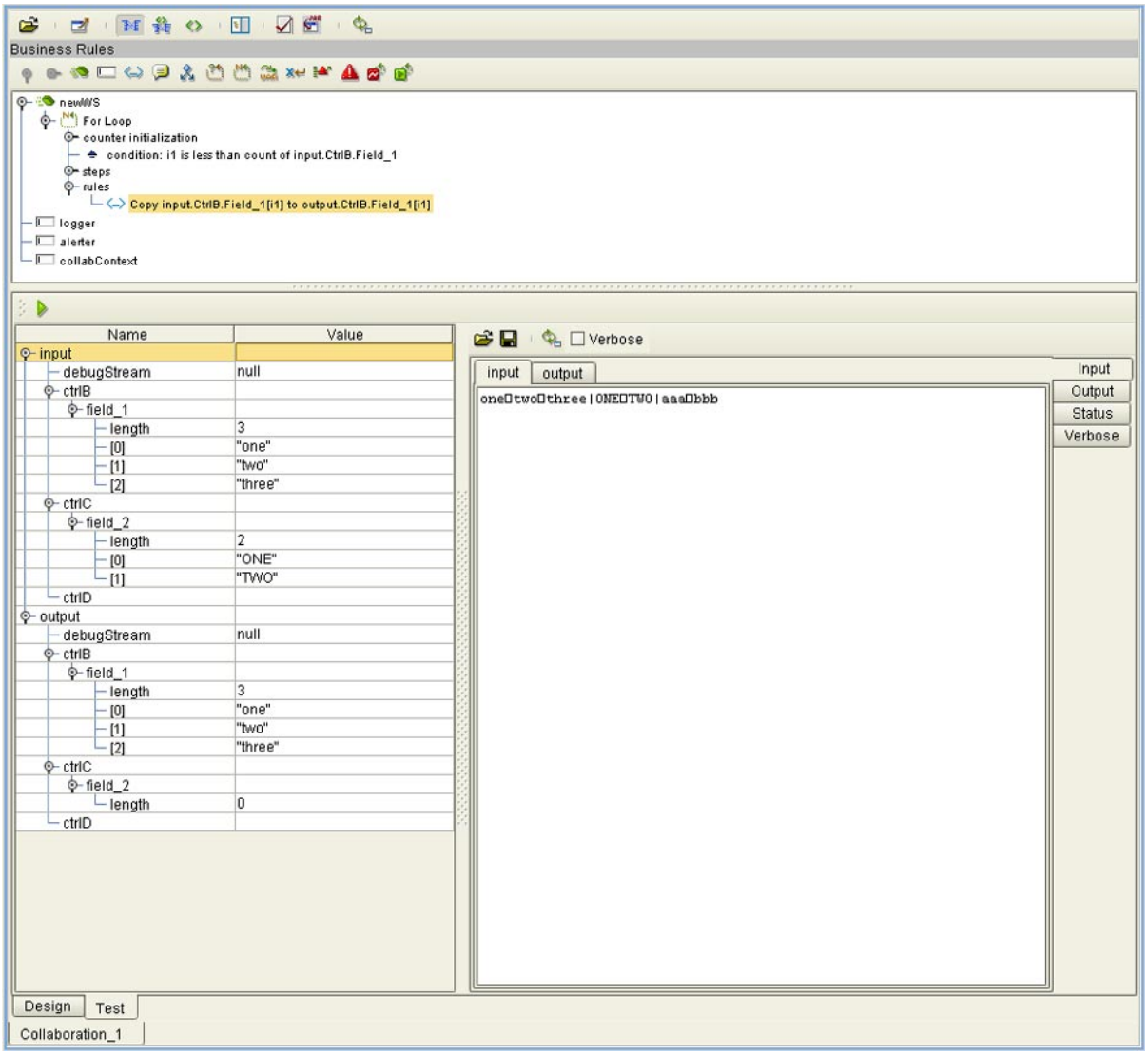
Table 50 Java Source Editor Toolbar Icons

Icon	Command	Function
	Roll Back Changes	Cancels changes made in the Java Source Editor and returns the Java-based Collaboration Definition to the last saved state.
	Find	Opens the **Find** dialog box, where you can enter text to search for in the Java Source Editor. 
	Find Previous	Searches backward for a previous instance of the text entered in the Find dialog box.
	Find Next	Searches forward for the next instance of the text entered in the Find dialog box.
	Replace	Opens the **Replace** dialog box, where you can enter text to search for, and to replace with, in the Java Source Editor. 
	Undo	Undoes the last operation.
	Redo	Restores the last operation, if undone.

7.4.4 Collaboration Tester

The JCE Tester (see Figure 148) is enabled only when you create a Collaboration as a new Web Service Type. The tester works in the same manner as the OTD Tester; please see **OTD Tester** on page 119 for more information.

Figure 148 JCE Tester

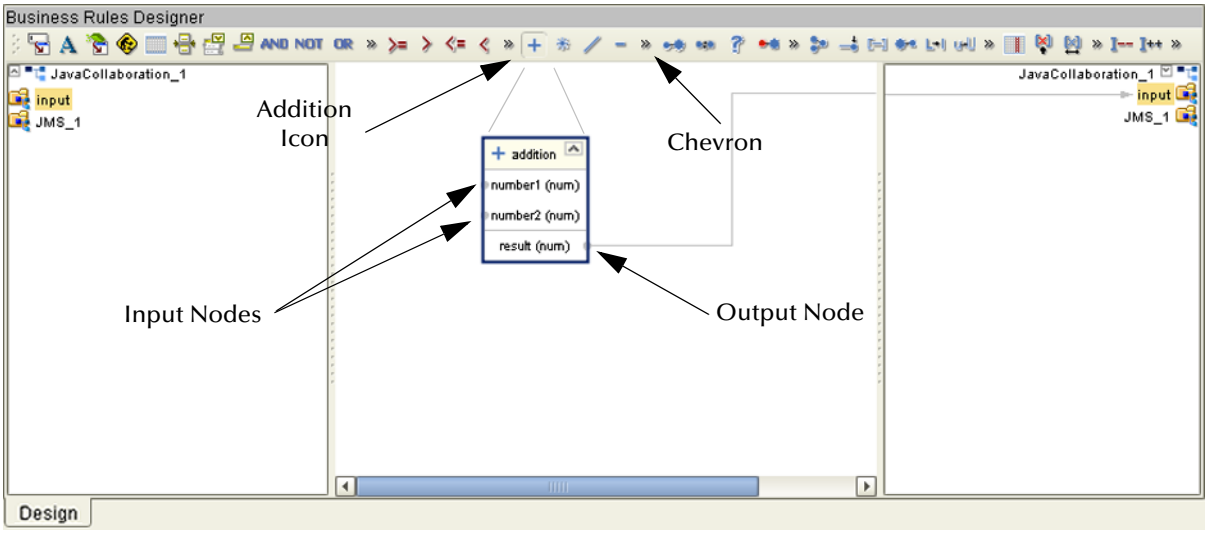


7.4.5 Business Rules Designer

The Business Rules Designer (see Figure 149) lists each field included in the Java-based Collaboration Definition. This area has its own toolbar and method palette which you use to map fields and add methods.

Note: *The Enterprise Designer automatically disables the Business Rules Designer when you are actively using the Java Source Editor. This is a safety feature that prevents you from inadvertently making contradictory changes. To re-enable the Business Rules Designer, click the* ***Commit Changes*** *button in the main toolbar.*

Figure 149 Business Rules Designer: Addition Method



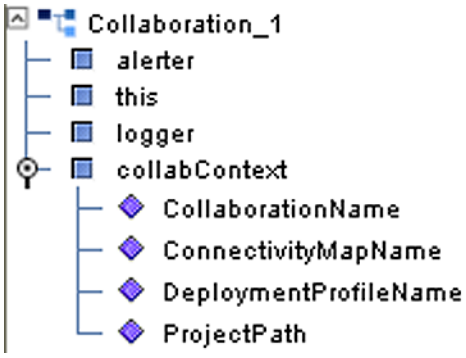
Note: *If you encounter an error while using the addition, subtraction, multiplication, or division operators, try manually converting the string to integers.*

Method-Access Nodes

When you create a Java-based Collaboration, several method-access nodes are added automatically in the Business Rules Designer (see Figure 150).

Note: *You must double-click the Collaboration to display these nodes.*

Figure 150 Method-Access Nodes



To call one of these methods, right-click on the node to display the context menu and click *Select a method to call ...* .

Table 51 Method Access Nodes

Node	Subnode	Description
alerter		Allows access to **alert** method, as described in **Creating Alerts** on page 240.
this		Allows you to call other methods defined for *this* Collaboration without having to edit the Java code.
logger		Allows access to **logging** method, as described in **Creating Log Entries** on page 243.
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.

Business Rules Designer Toolbar Icons

Table 52 Business Rules Designer Toolbar Icons

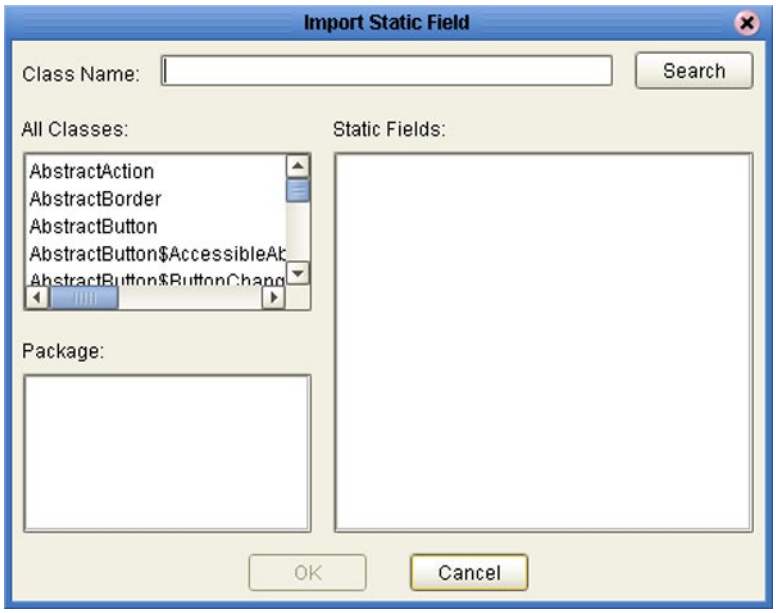
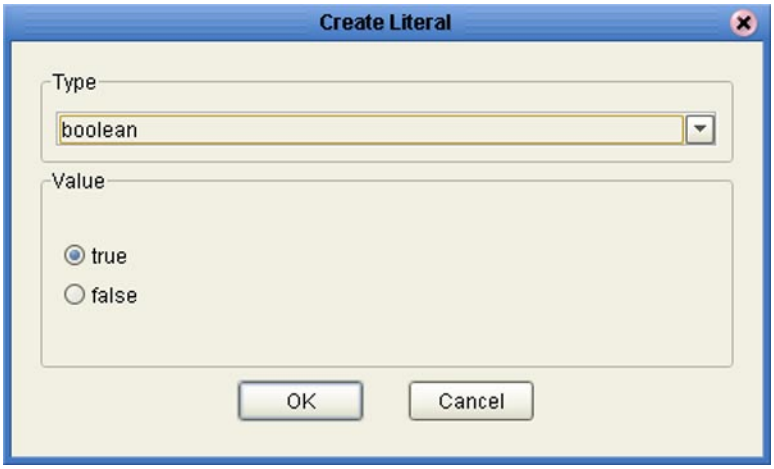
Icon	Command	Function
	Import Static Field	Displays the *Import Static Field* dialog box, which you can use to select a field to add to the Collaboration Definition. 
	Create Literal	Displays the *Create Literal* dialog box, which you can use to select a literal and enter a value to add to the Collaboration Definition.  Note that a space character can be created by selecting **Char** as a type, and simply entering a space as a value. Although nothing will be visible in the Value field of the dialog box, the character will appear in the Business Rules canvas, as follows: 

Table 52 Business Rules Designer Toolbar Icons

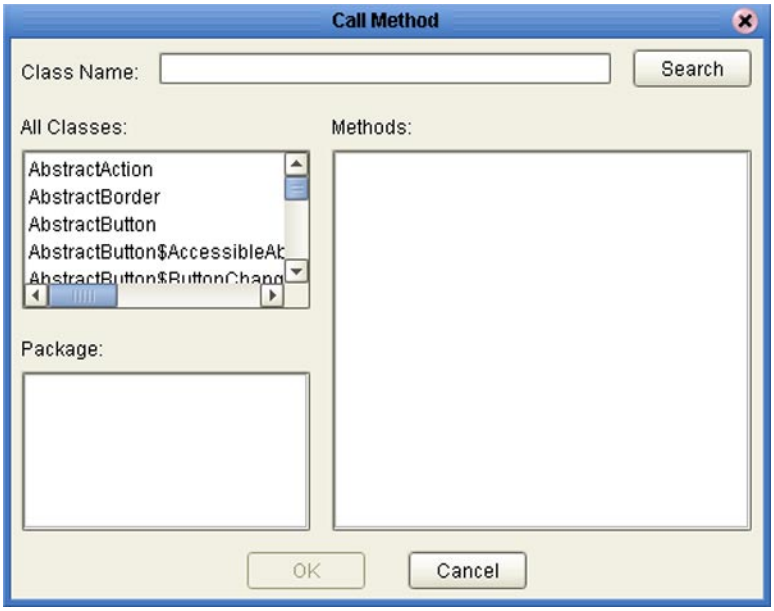
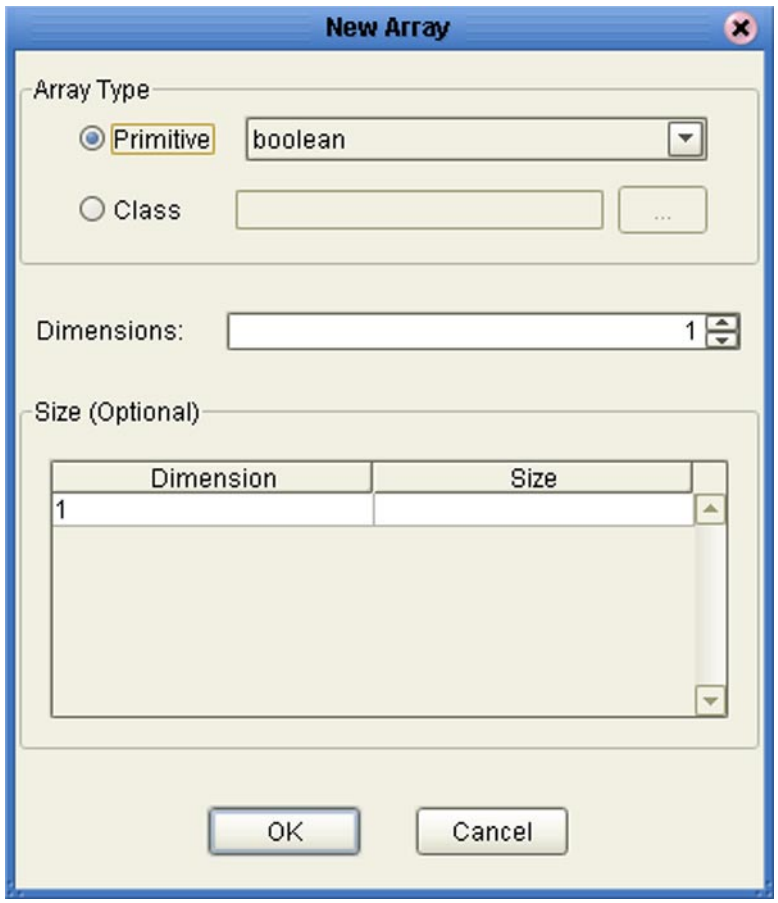
Icon	Command	Function
	Call Java Method	Displays the *Call Java Method* dialog box , which you can use to select a Java method to add to the Collaboration Definition. 
	Call New Constructor	Displays the *Call New Constructor* dialog box, which you can use to select a Java class to the Collaboration Definition. 

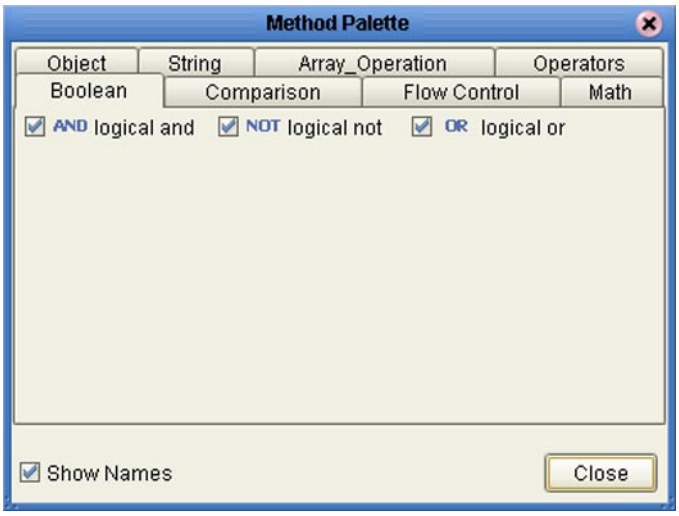
Table 52 Business Rules Designer Toolbar Icons

Icon	Command	Function
	New Array	Displays the *New Array* dialog box, which you can use to add an array to the Collaboration Definition. 
	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.

Collaboration Method Palette (Java)

The Collaboration Method Palette includes a series of method icons that you can drag onto the Business Rules Designer mapping area. Click the Chevron (>>) to the right of the method groups to display the dialog box shown in Figure 151. Select a check box to add the method to the toolbar; clear a check box to remove the method from the toolbar. The methods are described in **Collaboration Methods (Java)** on page 193.

Figure 151 Java Collaboration Method Palette Dialog Box



Collaboration Method Boxes (Java)

The method boxes are placed in the mapping panel of the Business Rules Designer by dragging the corresponding icon from the method palette toolbar. As shown in **Figure 149 on page 187**, the method boxes typically have input and output nodes that you link to fields in the left and right panels, respectively. The method boxes are expanded by default (see Figure 152); you can collapse them (see Figure 153) 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 152 Expanded Method Box

Figure 153 Collapsed Method Box

7.5 Collaboration Methods (Java)

7.5.1 Boolean Methods

Figure 154 Method Palette: Boolean Methods

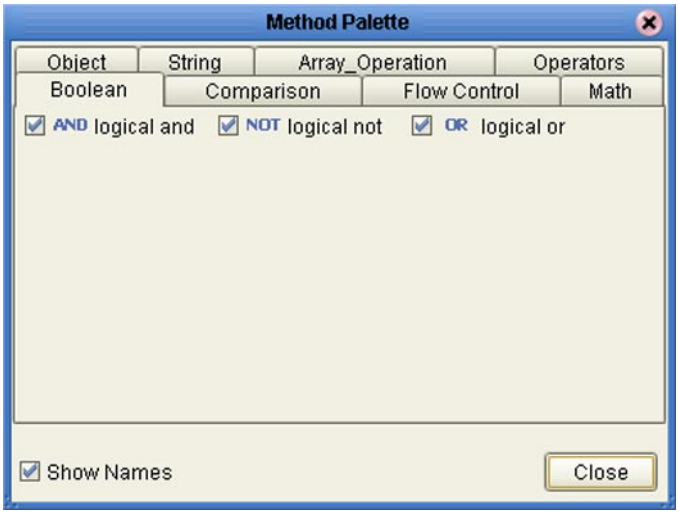
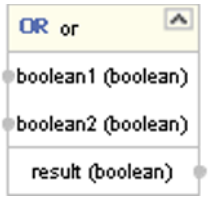
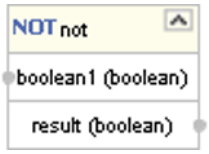


Table 53 Boolean Collaboration Methods (Java)

Method Box	Description/Usage
	The **AND** method returns Boolean true if both *boolean1* and *boolean2* are true; otherwise, returns Boolean false.
	The **OR** method returns Boolean false if both *boolean1* and *boolean2* are false; otherwise, returns Boolean true.
	The **NOT** method returns the inverse of *boolean1*.

7.5.2 Comparison Methods

Figure 155 Method Palette: Comparison Methods

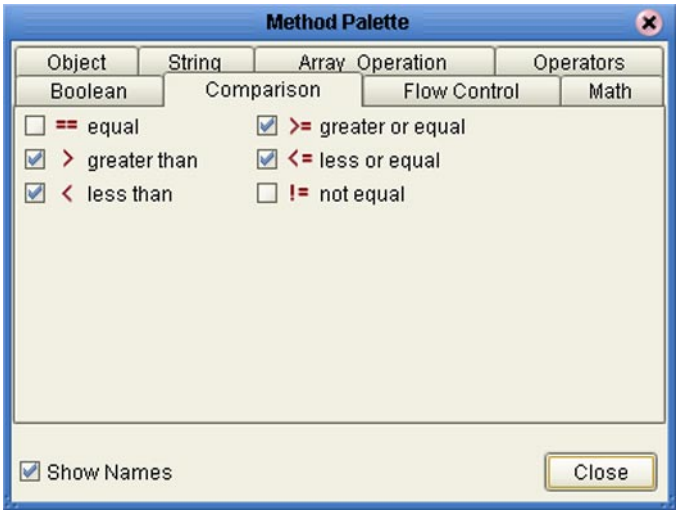


Table 54 Comparison Collaboration Methods (Java)

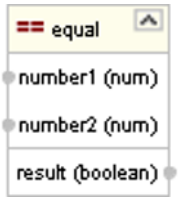
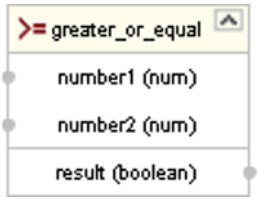
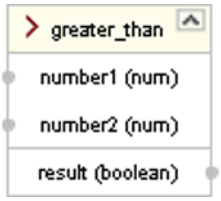
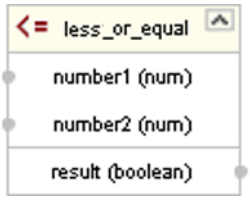
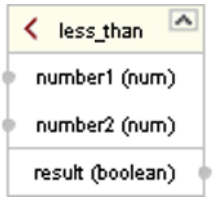
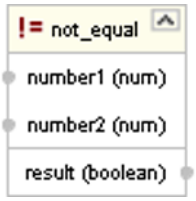
Method Box	Description/Usage
	The **equal** method returns Boolean true if *number1* is equal to *number2*; otherwise, returns Boolean false.
	The **greater or equal** method returns Boolean true if *number1* is greater than or equal to *number2*; otherwise, returns Boolean false.
	The **greater than** method returns Boolean true if *number1* is greater than *number2*; otherwise, returns Boolean false.

Table 54 Comparison Collaboration Methods (Java)

Method Box	Description/Usage
	The **less or equal** method returns Boolean true if *number1* is less than or equal to *number2*; otherwise, returns Boolean false.
	The **less than** method returns Boolean true if *number1* is less than *number2*; otherwise, returns Boolean false.
	The **not equal** method returns Boolean true if *number1* is not equal to *number2*; otherwise, returns Boolean false.

7.5.3 Flow Control Methods

Figure 156 Method Palette: Flow Control Methods

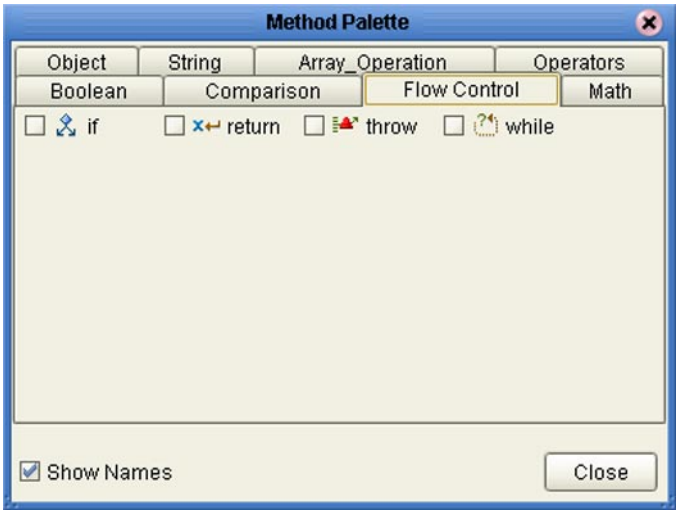
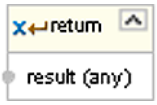


Table 55 Math Collaboration Methods (Java)

Method Box	Description/Usage
	The **if** method provides the boolean IF condition.
	The **return** method returns the result, if any.
	The **throw** method throws an exception.
	The **while** method provides the boolean WHILE condition.

7.5.4 Math Methods

Figure 157 Method Palette: Math Methods

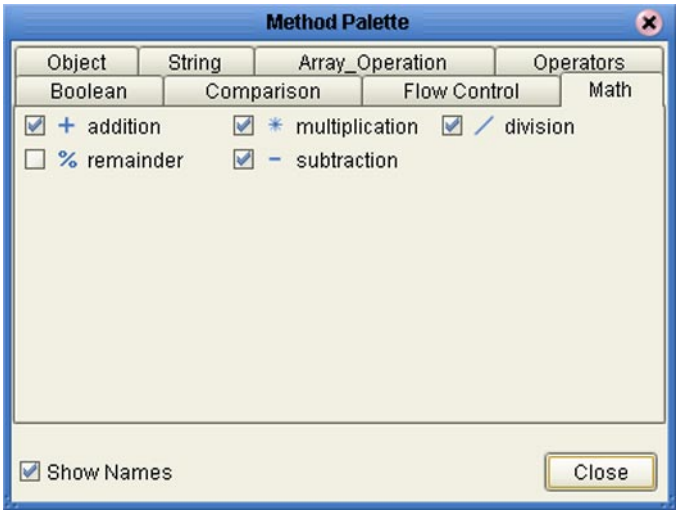


Table 56 Math Collaboration Methods (Java)

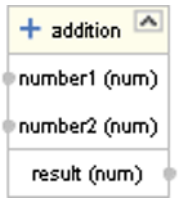
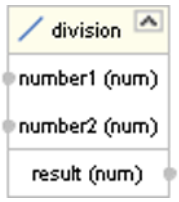
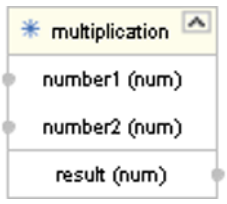
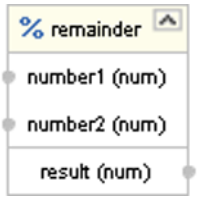
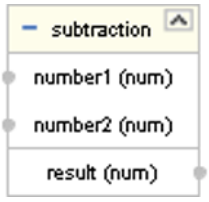
Method Box	Description/Usage
	The **addition** method adds the value of *number1* to the value of *number2*, returns the sum.
	The **division** method divides the value of *number1* by the value of *number2*, returns the quotient.
	The **multiplication** method multiplies the value of *number1* by the value of *number2*, returns the product.

Table 56 Math Collaboration Methods (Java)

Method Box	Description/Usage
	The **remainder** method divides the numerical value of *number1* by the numerical value of *number2*, returns the remainder.
	The **subtraction** method subtracts the numerical value of *number2* from the numerical value of *number1*, returns the difference.

7.5.5 Object Methods

Figure 158 Method Palette: Object Methods

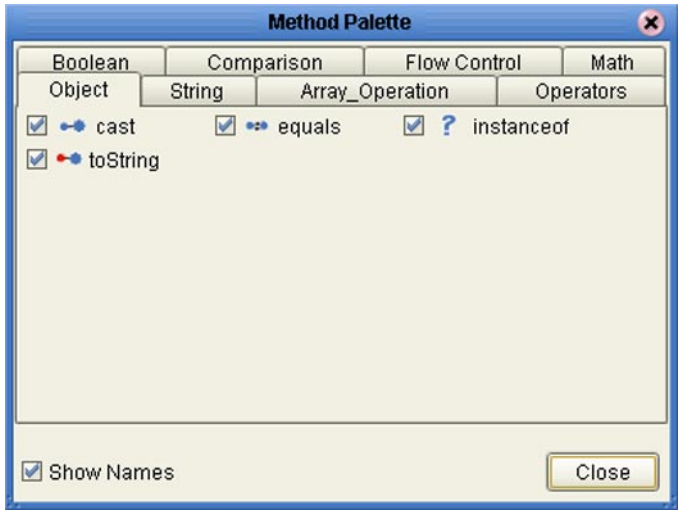


Table 57 Object Collaboration Methods (Java)

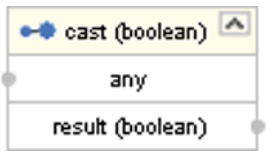
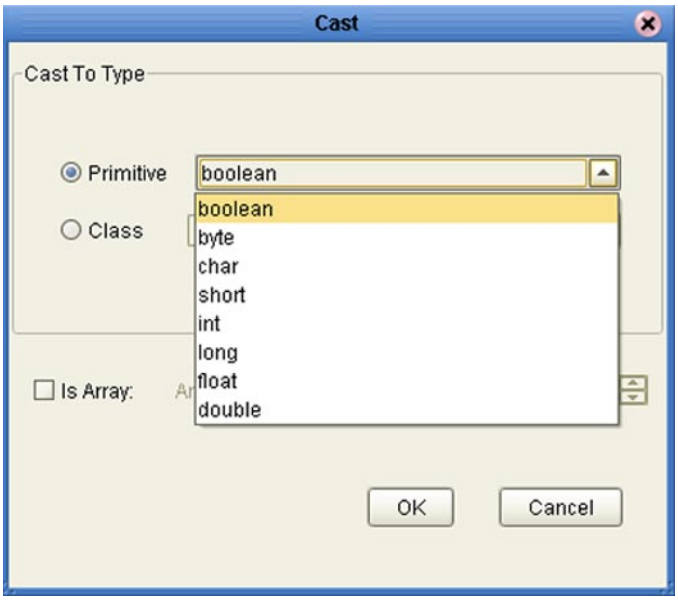
Method Box	Description/Usage
	Dragging the icon first displays the *CAST* dialog box, in which you specify the desired type (can be either primitive or class). The **cast** method then converts the given type (*any*) to the type specified in the CAST dialog box. 

Table 57 Object Collaboration Methods (Java)

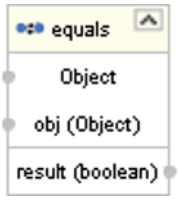
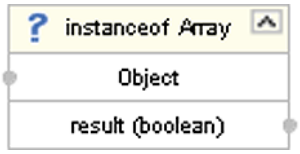
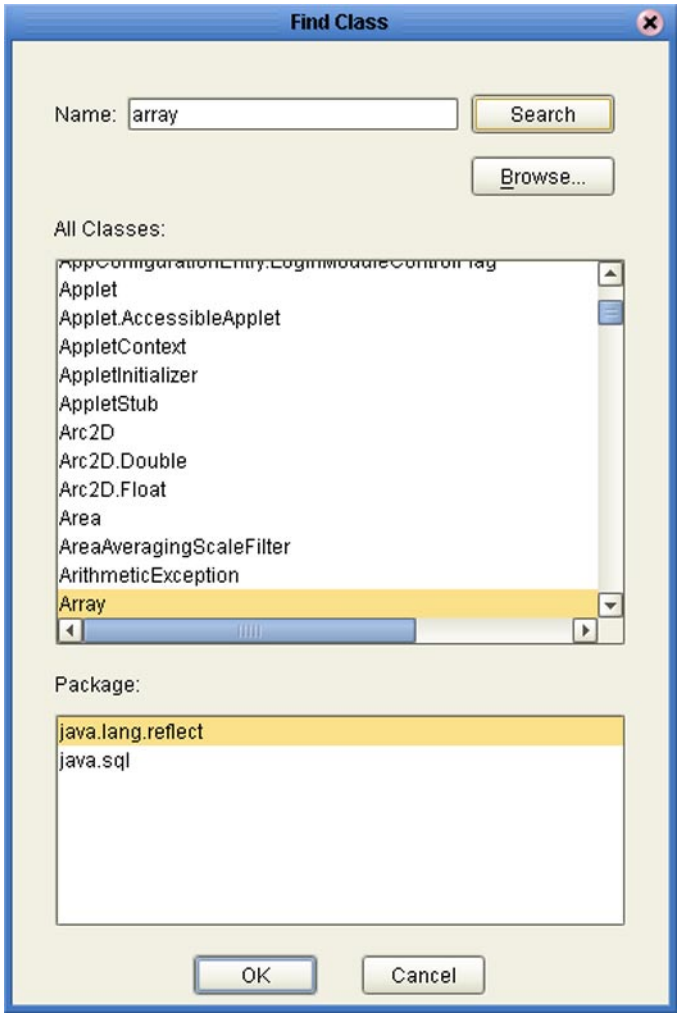
Method Box	Description/Usage
	The **equals** method returns Boolean true if *Object* is logically equal to *obj*, otherwise, returns Boolean false.
	Dragging the icon first displays the *Find Class* dialog box, in which you specify the desired class. The **instanceof** method returns Boolean true if *Object* is of the specified type; otherwise, returns Boolean false.  Clicking **Browse** displays the *Select Java Class* browser (see Figure 159), which allows you to browse for the class based on the package name, for example.

Table 57 Object Collaboration Methods (Java)

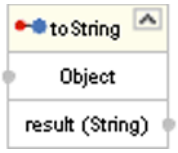
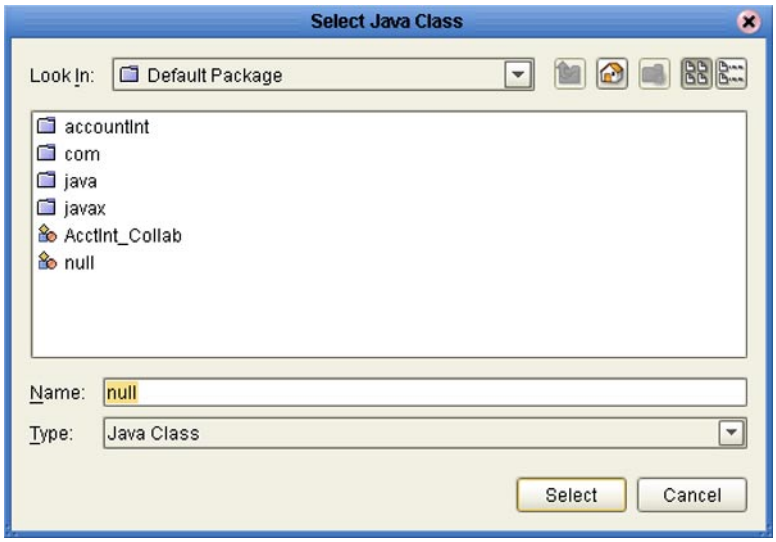
Method Box	Description/Usage
	The **toString** method converts the object in the input field into a string.

Figure 159 Select Java Class Browser



In the *Select Java Class* browser, note that the **Home** button selects the **Default Package**.

7.5.6 String Methods

Figure 160 Method Palette: String Methods

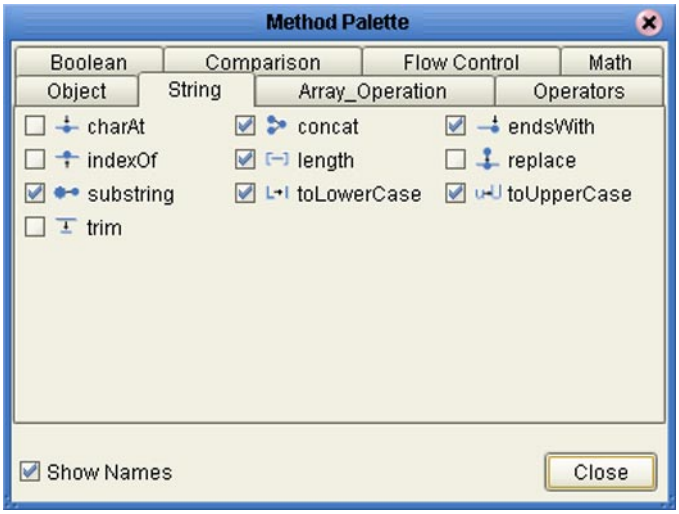


Table 58 String Collaboration Methods (Java)

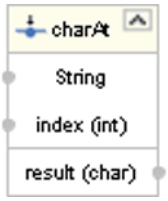
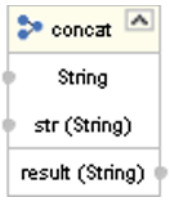
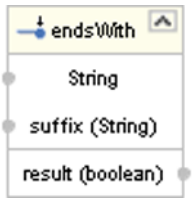
Method Box	Description/Usage
	The **charAt** method returns the character at the specified index, where *index* is the number of characters from the beginning of *String*.
	The **concat** method returns the string created by concatenating *str* to the end of *String*.
	The **endsWith** method returns Boolean true if *String* ends with the string *suffix*; otherwise, returns Boolean false.

Table 58 String Collaboration Methods (Java)

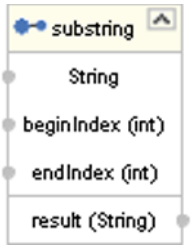
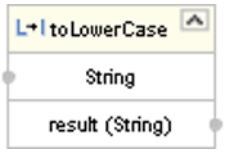
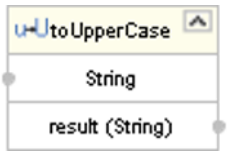
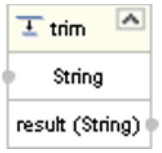
Method Box	Description/Usage
	The **indexOf** method returns the index within *String* corresponding to the location of the specified character (*ch*), where the index represents the number of characters from the beginning of *String*.
	The **length** method returns the length (number of characters) of *String*.
	The **replace** method returns a new string in which all occurrences of *oldChar* are replaced with *newChar*.
	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*.
	The **toLowerCase** method converts all characters in *String* to lower case, using the rules of the default locale.
	The **toUpperCase** method converts all characters in *String* to upper case, using the rules of the default locale.

Table 58 String Collaboration Methods (Java)

Method Box	Description/Usage
	The **trim** method trims leading and trailing whitespace from *String*.

7.5.7 Array Operation Methods

Figure 161 Method Palette: Array Operation Methods

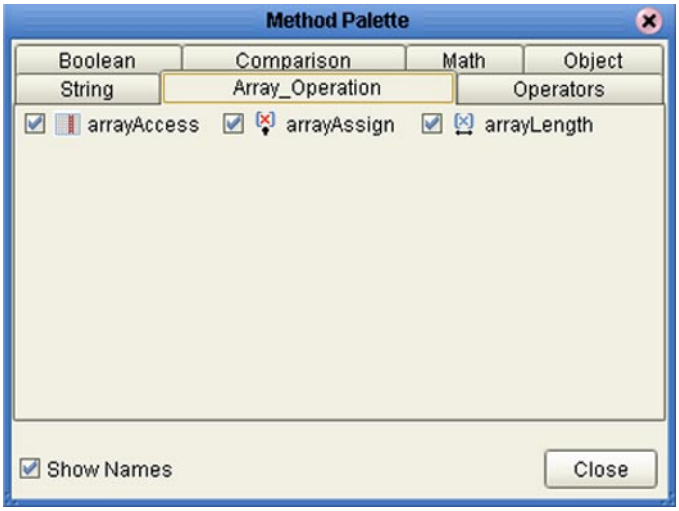


Table 59 Array Operation Collaboration Methods (Java)

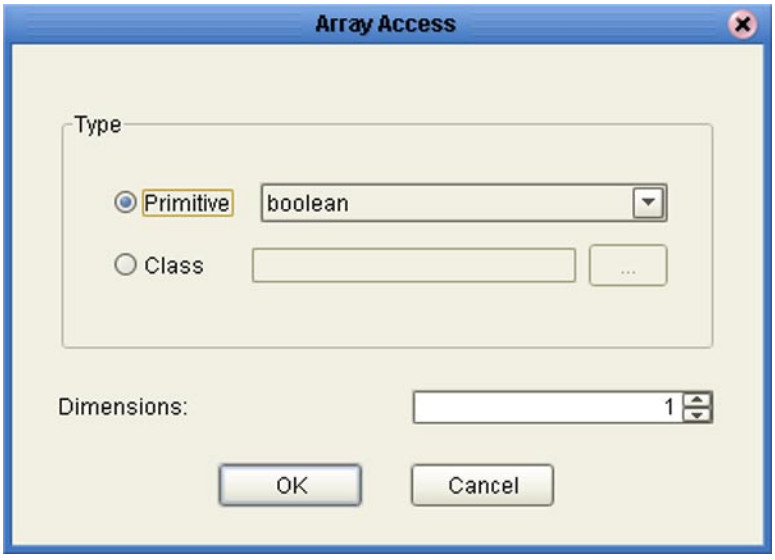
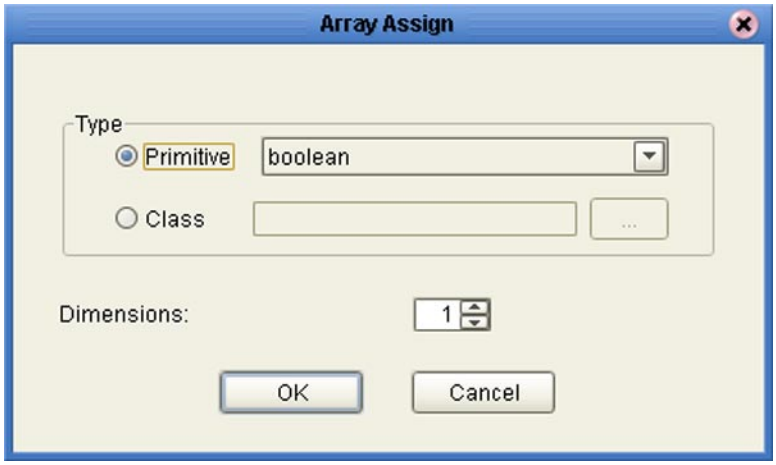
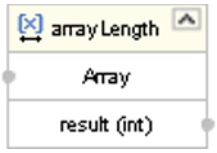
Method Box	Description/Usage
	Dragging the icon first displays the *Array Access* dialog box, in which you specify the desired array type (can be either primitive or class) and dimension. The **arrayAccess** method then returns the element located at the specified index within the array, where the index represents the number of elements from the beginning of the array. 

Table 59 Array Operation Collaboration Methods (Java)

Method Box	Description/Usage
	Dragging the icon first displays the *Array Assign* dialog box, in which you specify the desired array type (can be either primitive or class) and dimension. The **arrayAssign** method then 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. 
	The **arrayLength** method returns the length (number of elements) of the array.

7.5.8 Operators

Figure 162 Method Palette: Operators

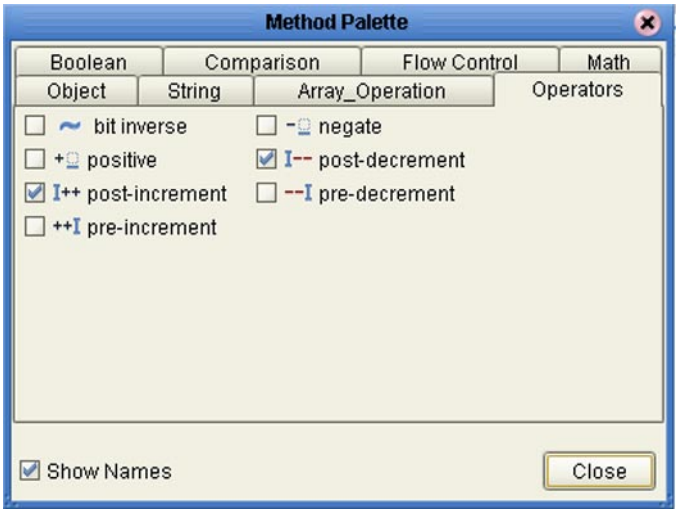


Table 60 Operators (Java)

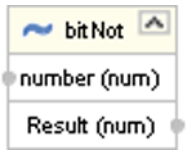
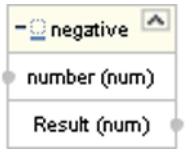
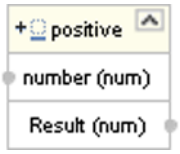
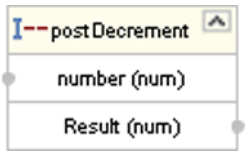
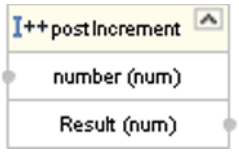
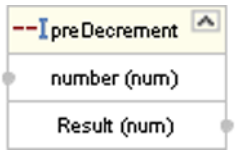
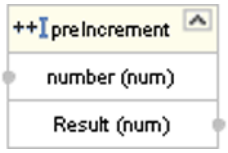
Operator Box	Description/Usage
	The **bitNot** operator returns the inverse of the bit value of the input number.
	The **negative** operator returns the arithmetic negation of the input number.
	The **positive** operator returns the value of the input number.
	The **postDecrement** operator decrements the input number by one, after other operations.

Table 60 Operators (Java)

Operator Box	Description/Usage
	The **postIncrement** operator increments the input number by one, after other operations.
	The **preDecrement** operator decrements the input number by one, before other operations.
	The **preIncrement** operator increments the input number by one, before other operations.

7.6 Version Control

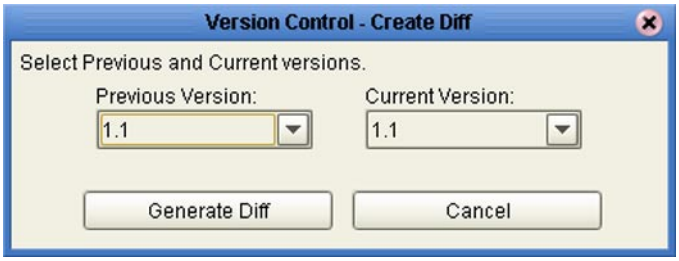
7.6.1 Creating a Modified Collaboration Definition (Java)

Java-based Collaboration Definitions can be saved to a Diff (**.sdf**) file. This feature allows two sites working with the same Java-based Collaboration Definition to seamlessly merge changes.

To create a *different* version of a Java-based Collaboration Definition file

1. In Project Explorer, select a **Collaboration Definition (Java)** icon.
2. Right-click to display the context menu (see **Figure 69 on page 100**).
3. Select **Create Diff** to display the dialog box shown in Figure 163.

Figure 163 Version Control - Create Diff Dialog Box



4. Click **Generate Diff** to display the *Specify Name ...* dialog box.
5. Enter a name for the Diff file in the **File Name** box (or use the default file name).
6. Click **Save** to save the Diff file.

7.6.2 Merging Two Versions of a Collaboration Definition (Java)

To merge two versions of a Java-based Collaboration Definition

1. From the Project Explorer tab, select a **Collaboration Definition (Java)** icon.
2. Right-click to display the Collaboration Definition (Java) context menu.
3. Select **Check Out**.
4. Right-click to again display the context menu.
5. Select **Merge Diff ...** to display the dialog box shown in Figure 164.

Figure 164 Version Control - Merge Changes Dialog Box

6. Click the **Ellipsis** (**...**) button to display the *Specify Name ...* dialog box.
7. Locate and select the Diff file.
8. Click **Open** add the Diff file to the dialog box shown in Figure 164.
9. Click **Merge** to merge the Diff file with the corresponding Collaboration Definition (Java) in your Project.
10. Resolve any conflicts from the merge.
11. **Commit** the merged Collaboration.
12. **Save** the merged Collaboration.

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

7.7 Setting Up Collaboration Definition Variables

The Collaboration Editor (Java) includes a variable creation feature. The following sections describe how to set up and assign a constructor to a variable.

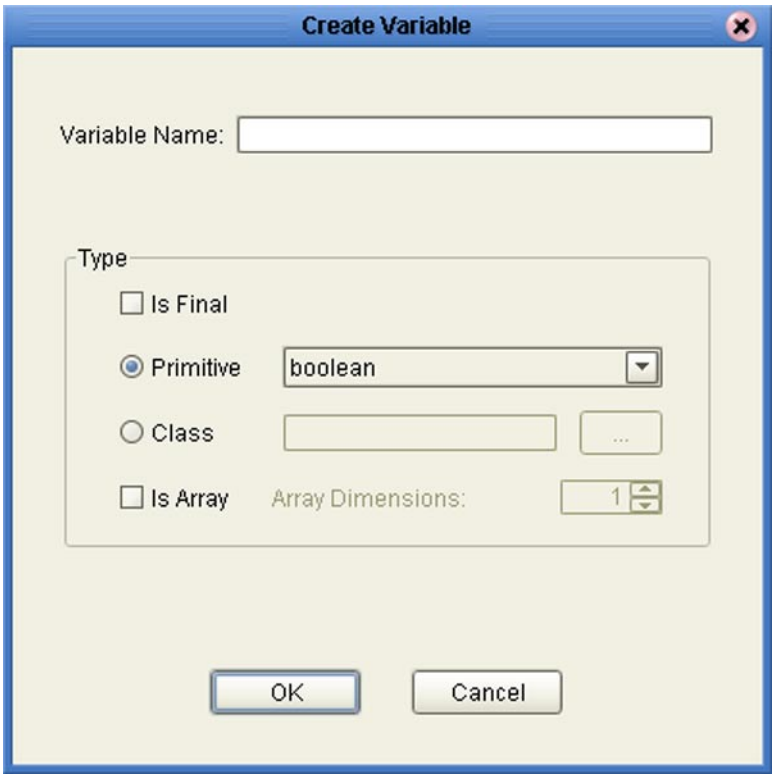
7.7.1 Creating a Variable

This section describes how to add a variable to a Java-based Collaboration Definition.

To create the variable

1 In the Business Rules area, click the **Local Variable** button to display the dialog box shown in Figure 165.

Figure 165 Create Variable Dialog Box

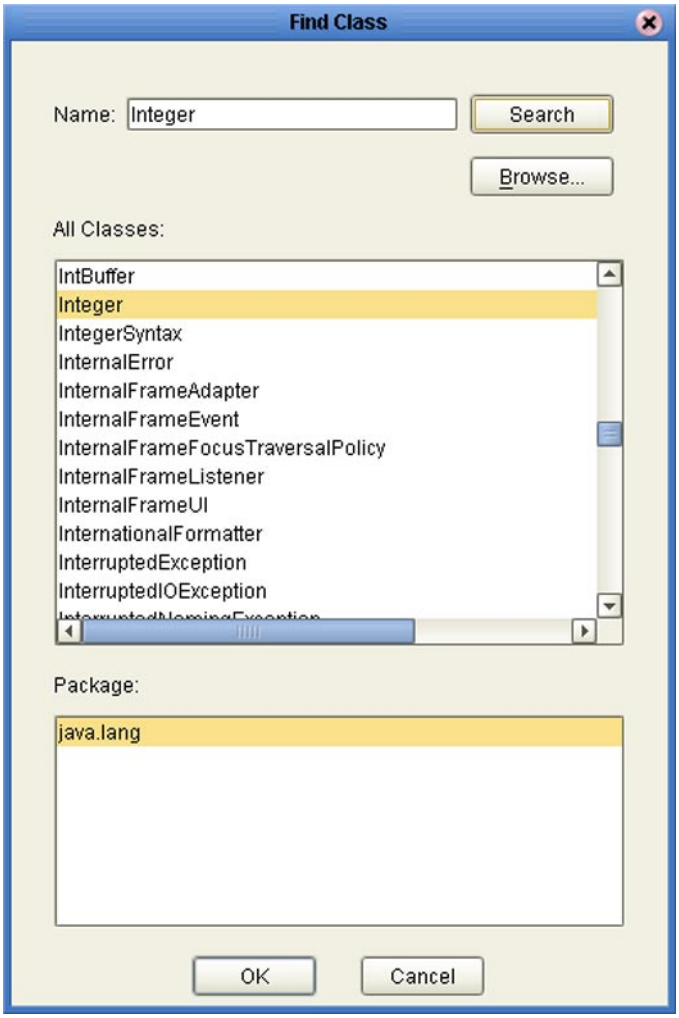


2 Enter a name for the variable in the **Variable Name** box.

3 Select the **Class** option button.

4 Click the **Ellipsis** (**...**) button to display the **Find Class** dialog box shown in Figure 166.

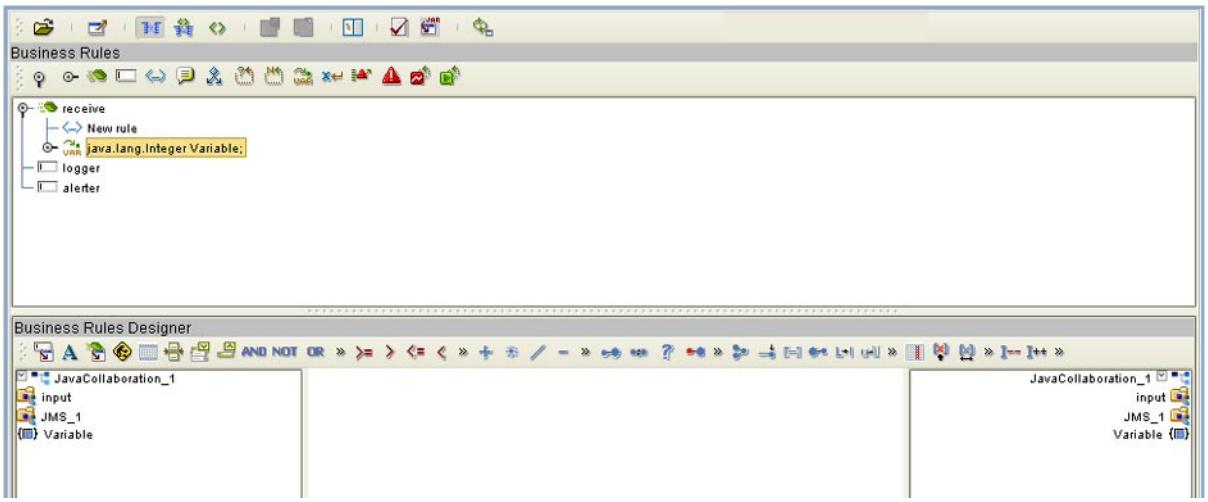
Figure 166 Find Class Dialog Box



5 Select a class from the **Class Name** list.

6 Click the **OK** button to add the variable, with selected class, to the Collaboration Editor (Java) window as shown in Figure 167.

Figure 167 Collaboration Definition (Java) with Variable



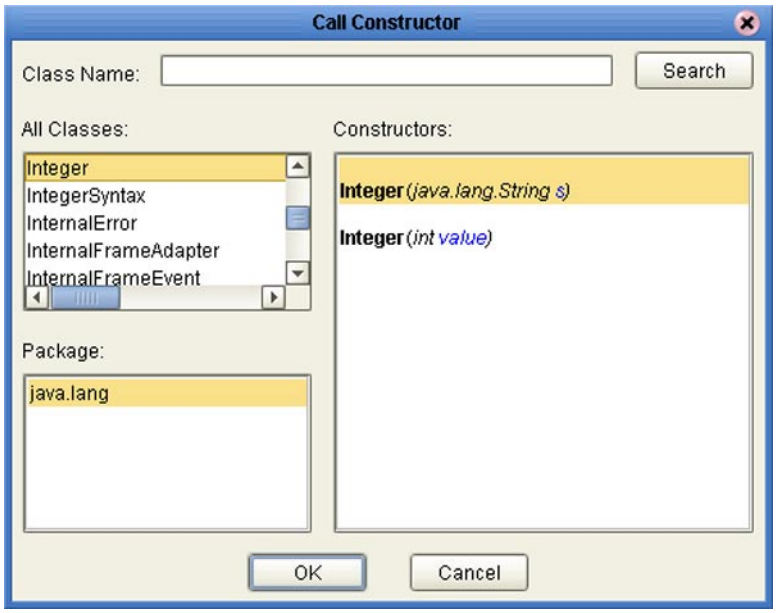
7.7.2 Invoking the Variable Constructor

This section describes how to invoke a constructor for the variable.

To invoke a constructor

1 In the Business Rules Designer area, click the **Call New Constructor** button to display the dialog box shown in Figure 168.

Figure 168 Call New Constructor Dialog Box

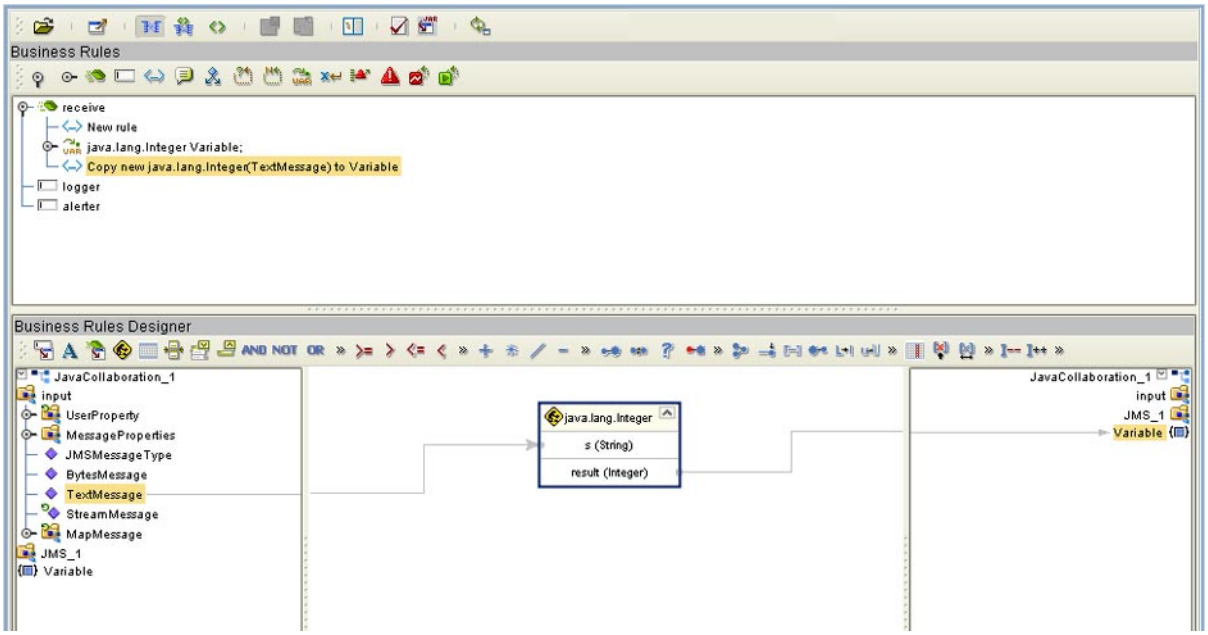


2 Select the class used for the variable in the **All Classes** list.

3 Click **OK** to add an integer box to the Business Rules Designer area as shown in Figure 169.

4 Link an item in the left column through the integer to an item in the right column.

Figure 169 Collaboration Definition (Java) with Constructor



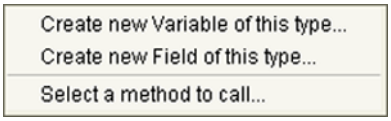
7.7.3 Displaying Method Classes

This section describes how to display a list of method classes.

To view the available method classes

1. Select a Variable in the Business Rules Designer area.
2. Right-click to display its context menu, as shown in Figure 170

Figure 170 Variable Context Menu



3. Click **Select a Method to Call** to display a list of method classes as shown in Figure 171.

Figure 171 Method List Box

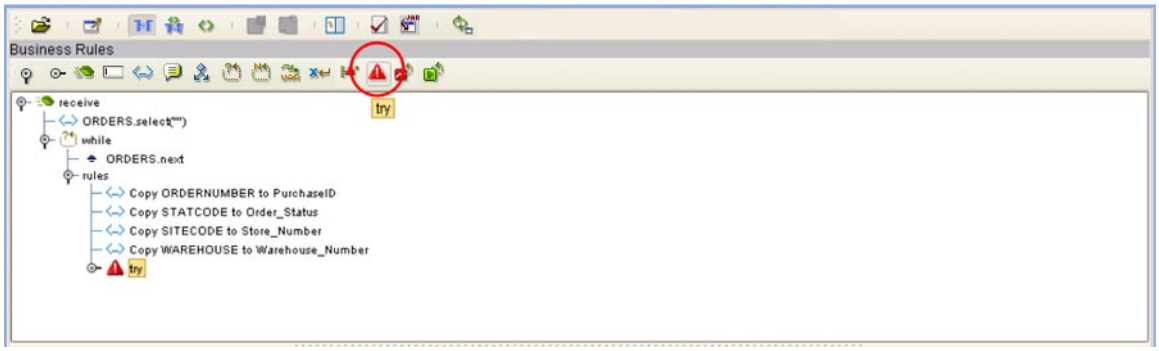
7.8 Using *Try-Catch*

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; if you want to perform a **try-catch**, you must use the following procedure.

To create a try-catch operation

1 Click the **try** icon in the toolbar (see Figure 172), which adds the **try** rule to the business rules tree.

Figure 172 Try Icon

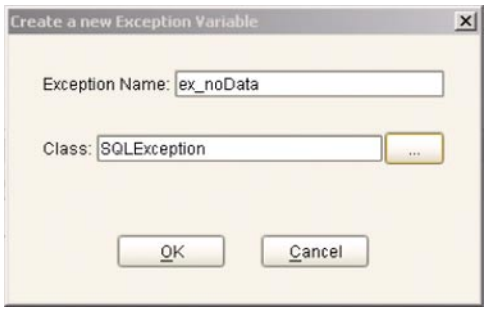


2 Right-click on the **try** rule to display the option menu (see Figure 173).

Figure 173 Try Option Menu (Partial)

3 Select the **catch** option to add an exception variable, which displays a dialog box (see Figure 174).

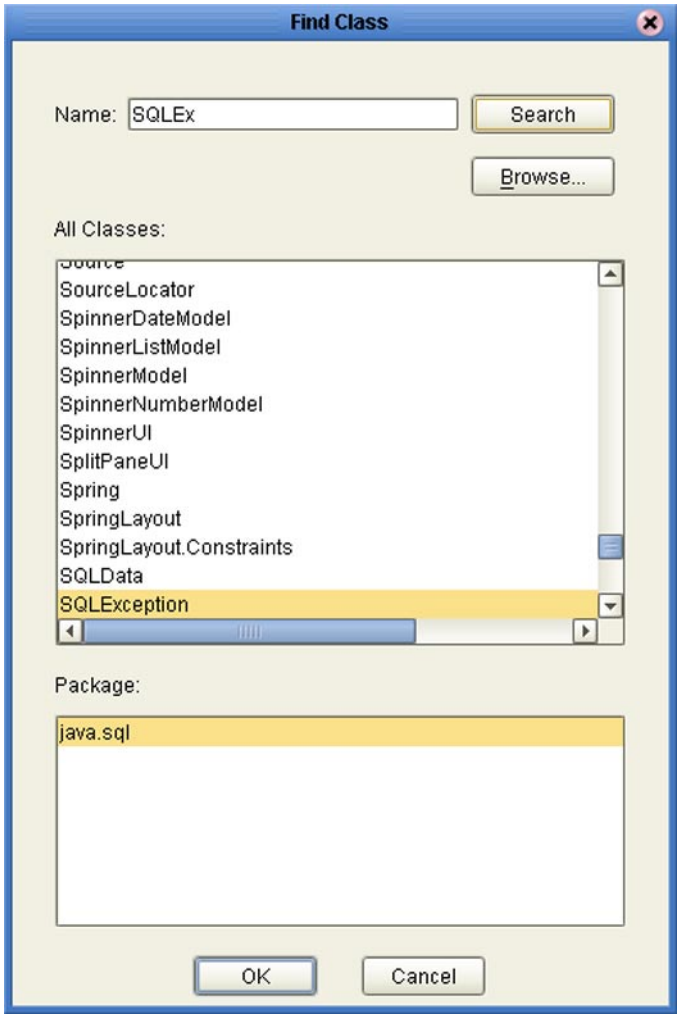
Figure 174 Create a New Exception Variable Dialog Box



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

5 Click the button to the right of the Class text box to display a list of available classes (see Figure 175).

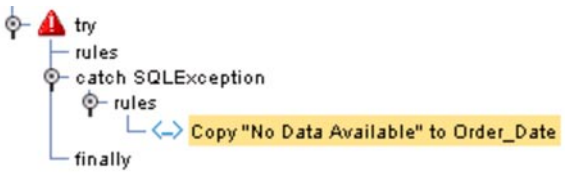
Figure 175 Find Class Dialog Box



6 To specify that the exception represents a database error, for example, select **SQLException** and click **OK** on both dialog boxes.

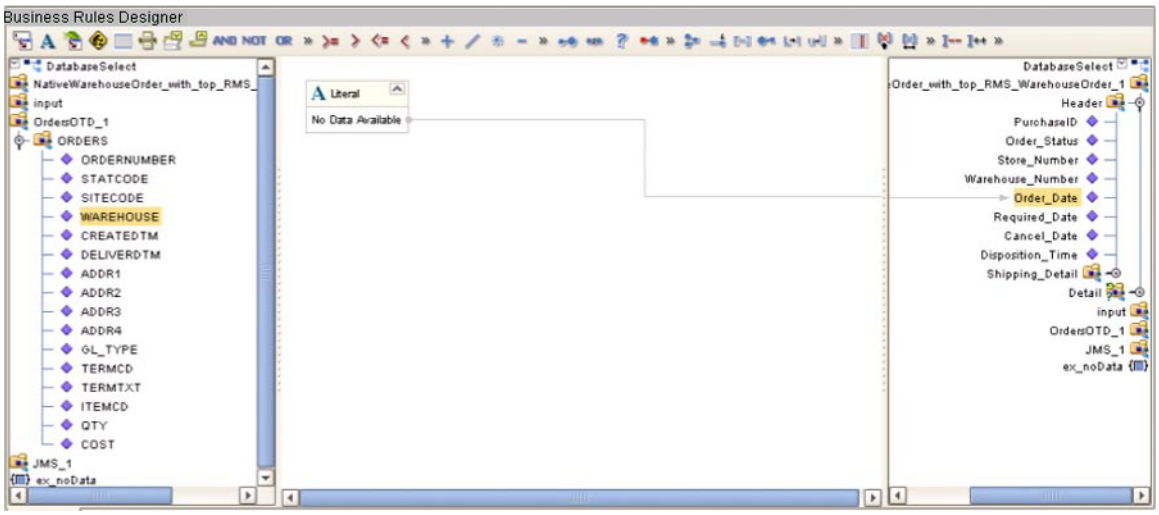
7 The catch rule is now added to the business rules tree (see Figure 176).

Figure 176 Catch SQLException Rule



8 The rule shown in Figure 176 contains a literal that represents the message that will be assigned to an outbound element Order_Date when the rule is executed (see Figure 177).

Figure 177 Exception Message



7.9 Adding and Using Third-Party Java Classes

Enterprise Designer allows you to add any file to the repository. Certain products in the ICAN Suite can then instruct the repository to distribute these files to Logical Hosts as needed. Selecting **New > File ...** in a Project context menu (see Figure 178) displays a file import dialog box (see Figure 179).

Figure 178 Project Context Menu: New File

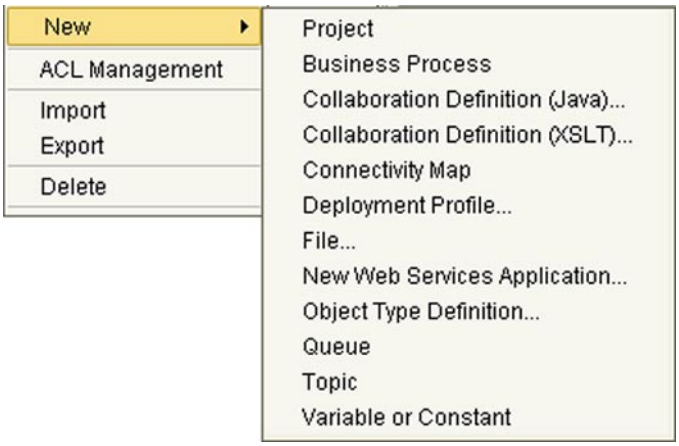
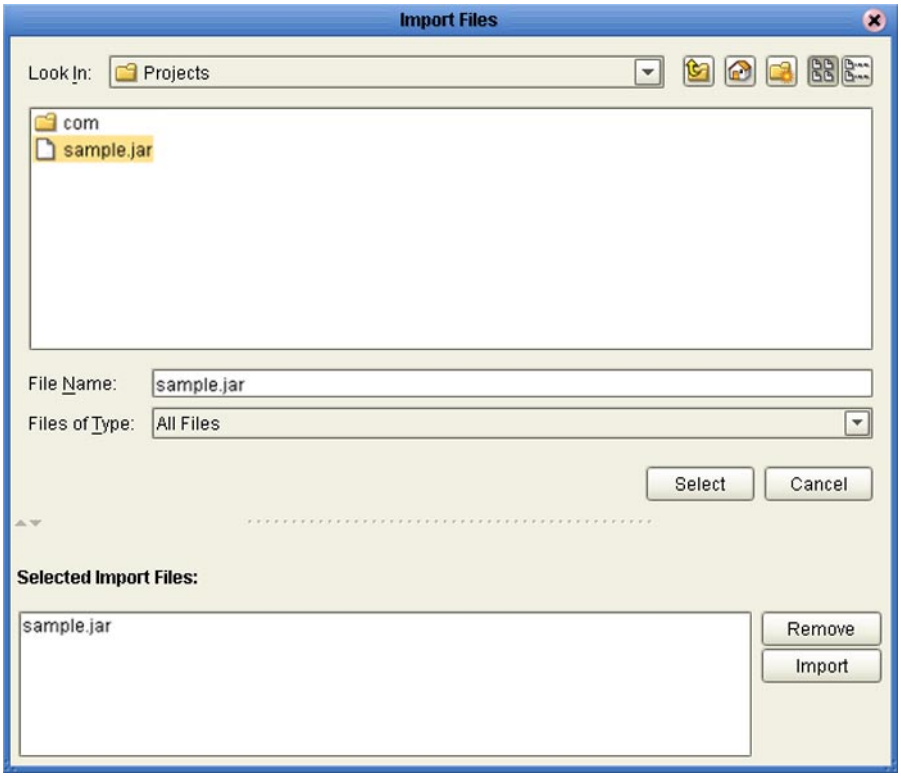


Figure 179 File Import Dialog Box



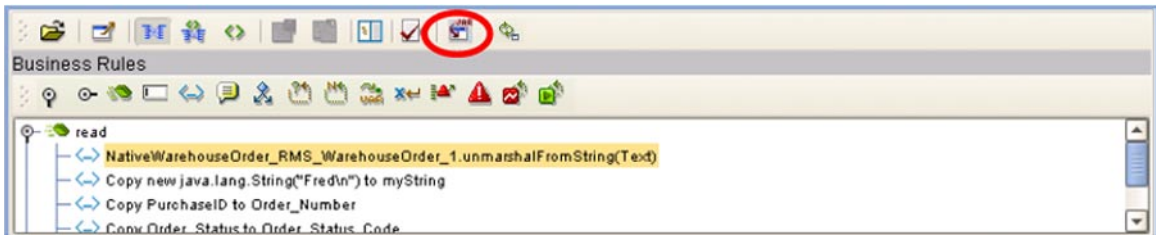
Locate the desired .jar files and click **Select** for each; then click **Import**.

To add the file(s) to a Collaboration Definition:

1 Open or create a Collaboration.

2 Click the **Import JAR File** icon in the toolbar (see Figure 180), which displays an *Add/Remove Jar Files* dialog box.

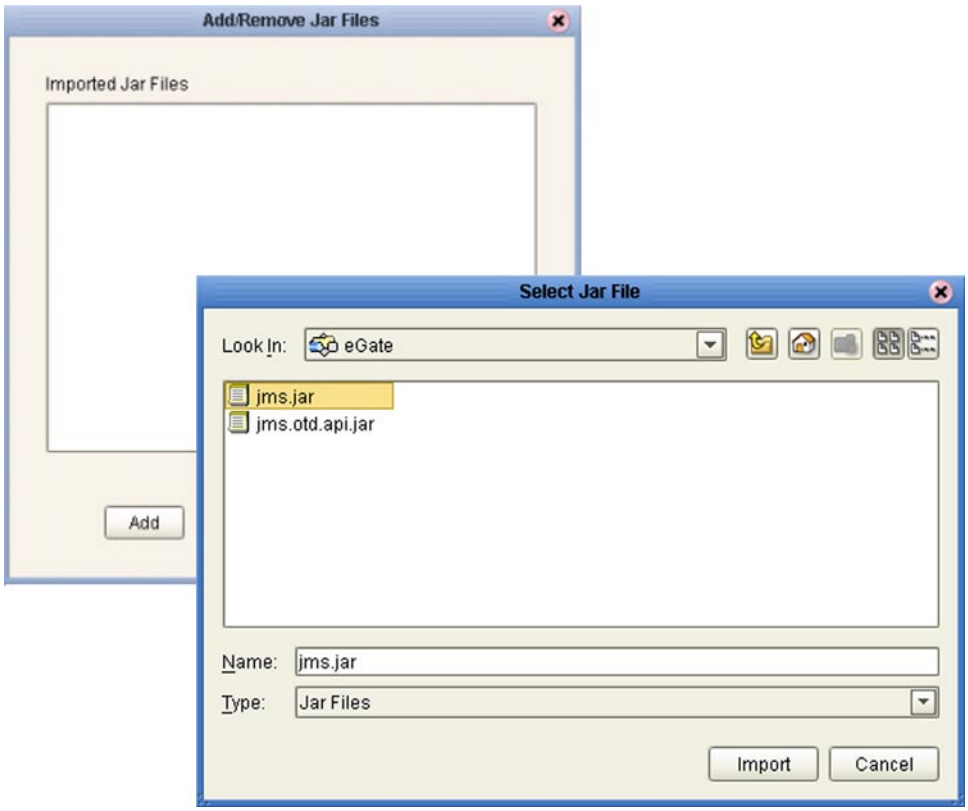
Figure 180 Import JAR File Icon



3 Click **Add** in the dialog box to display a file browser box (see Figure 182).

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

Figure 181 Add/Remove Jar File Dialog Box (1)

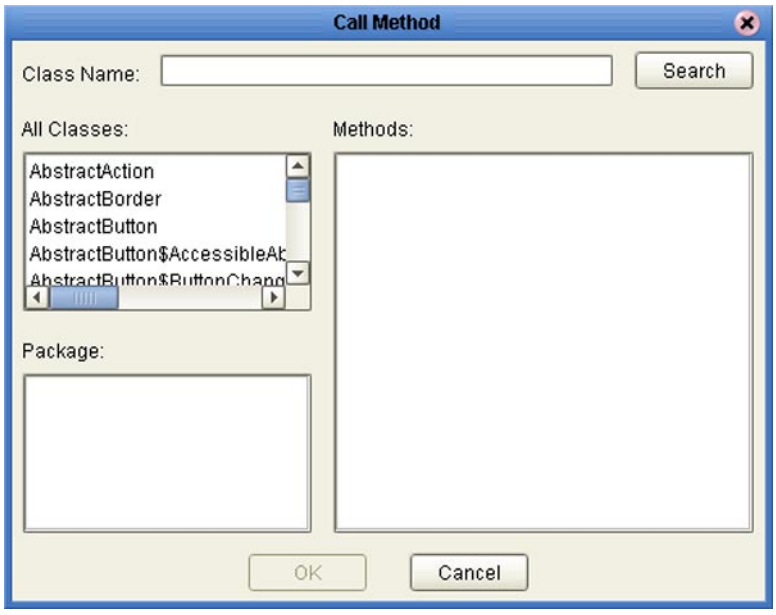


5 When all the **.jar** files are listed in the *Add/Remove Jar Files* dialog box (see Figure 182), click **Close**.

Figure 182 Add/Remove Jar File Dialog Box (2)

6 Click the **Call Java Method** icon in the Business Rules Designer, and a *Call Java Method* dialog box opens (see Figure 183).

Figure 183 Call Java Method Dialog Box



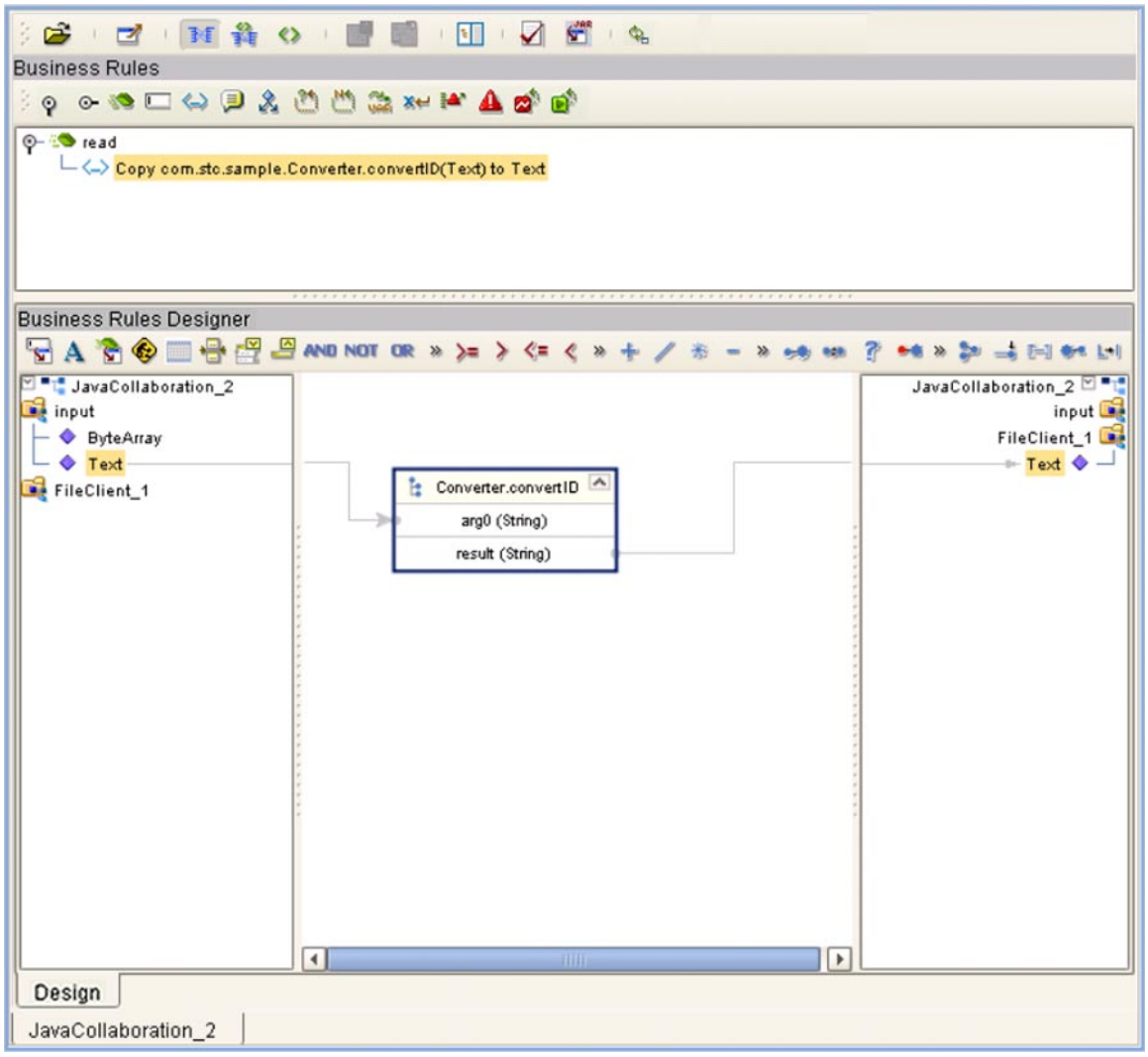
7 Select the desired class from the **All Classes** list.

8 Select the desired method from the **Methods** list.

9 Click **Ok**.

To use the third-party method, add the business rule containing the method. In the example shown, the method is a **convertID** operation that converts text from upper case to lower case. Connect an inbound text file to the input node of the method, and the outbound node to an outbound text file (see Figure 184).

Figure 184 Using the Third-Party Java Method



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

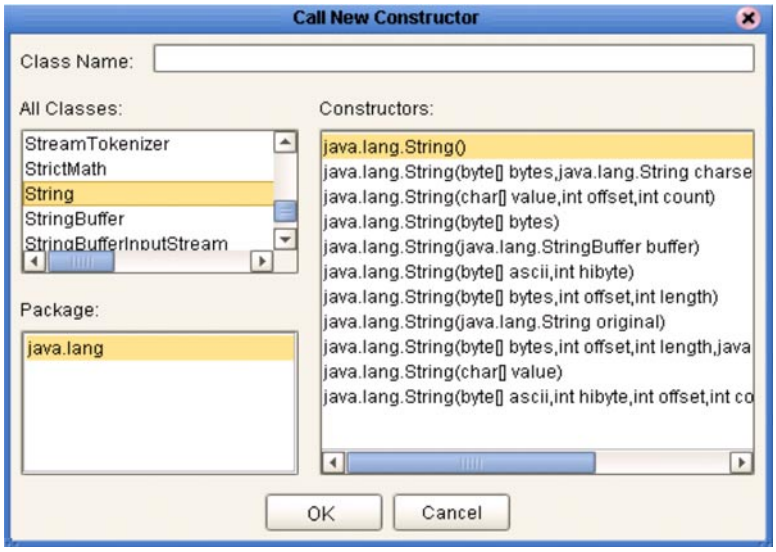
7.10 Adding Class Instances to a Collaboration

Instances of Java classes are added by means of the Java constructors, as described in the following procedure.

To add a class instance to a Java-based Collaboration Definition

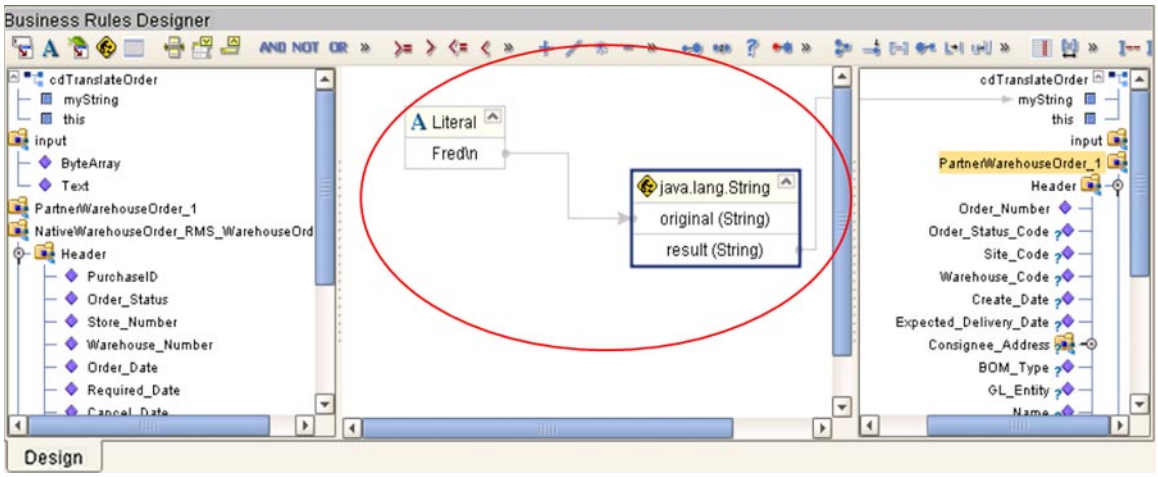
1. Create or open a Collaboration Definition and create a new rule.
2. Click the **Constructor** icon on the toolbar (see **Business Rules Designer Toolbar Icons** on page 189) to display the *Call New Constructor* dialog box (see Figure 185). This dialog box presents all available classes, including any third-party classes that have been uploaded.

Figure 185 Call New Constructor Dialog Box



3. Select a constructor method from the Constructors list and click **OK**. This method is placed on the Business Rules Designer's canvas, and can now be used as shown in Figure 186.

Figure 186 Constructor Example 1

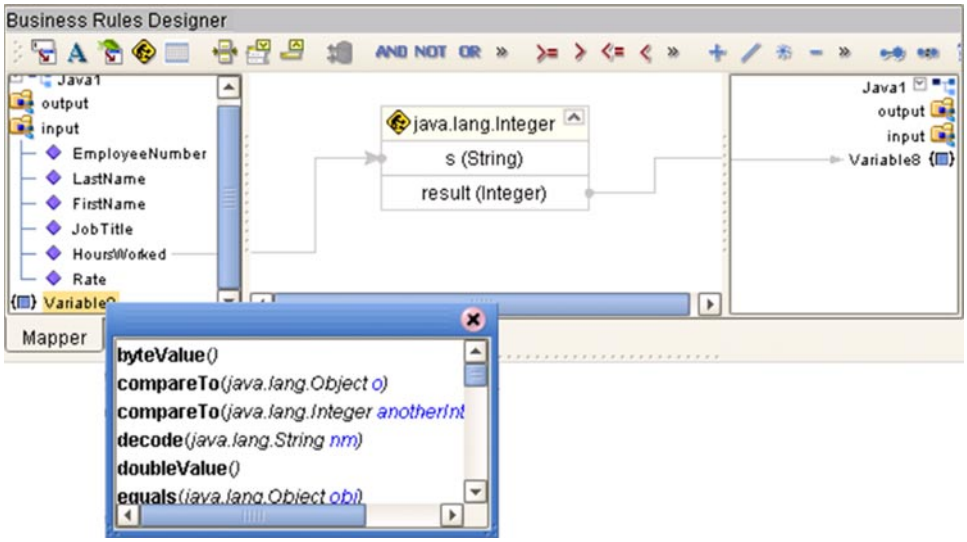


Alternative procedure to add a class instance to a Collaboration

An alternative way to invoke a constructor is to right-click on an element in the left panel of the Business Rules Designer and select a constructor method from the list box that appears, as illustrated in Figure 187. The procedure is:

1. Create a local string variable.
2. Right-click on the variable element.
3. Select a constructor from the list box.

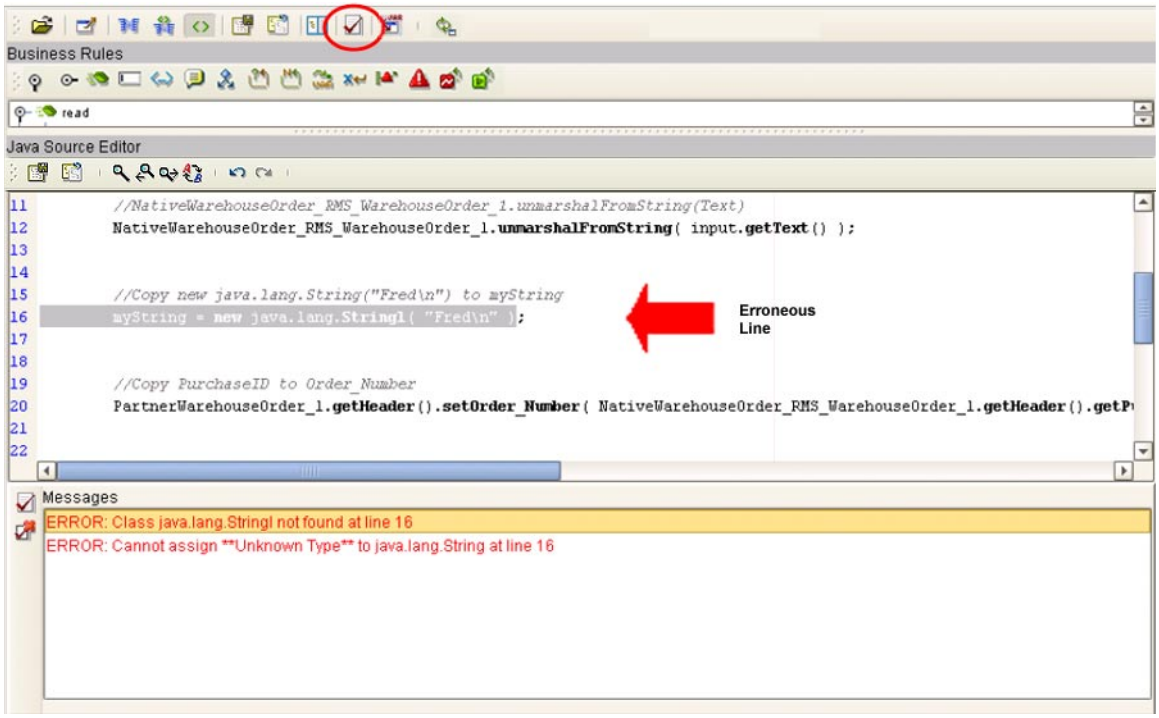
Figure 187 Constructor Example 2



7.11 Validating Java-based Collaboration Definitions

Clicking the **Validate** icon allows you to "precompile" the Java-based Collaboration Definition and display the errors in a validation panel, as shown in Figure 188. To locate the error, double-click on the error message and the Java Source Editor will be displayed, showing the erroneous line of code.

Figure 188 Validating a Collaboration Definition



7.12 Debugging Java-based Collaboration Definitions

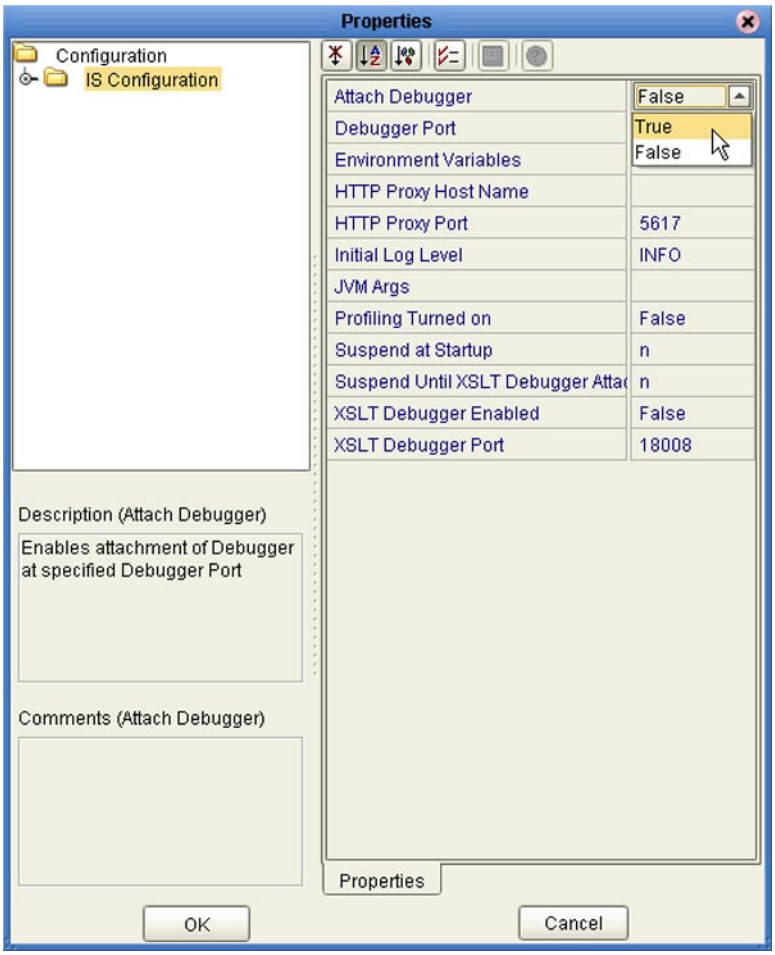
The Java Debugger enables you to debug Java-based Collaboration Definitions as deployed within an integration server on a Logical Host, and offers an alternative to creating logs and warnings in an Java-based Collaboration and subsequently inspecting them via the Enterprise Manager.

7.12.1 Enabling the Debugger

To enable the Java Debugger

1. In Enterprise Explorer, select the appropriate integration server.
2. Right-click to display the context menu.
3. Click **Properties** to display the Properties Dialog Box (see Figure 189).

Figure 189 Integration Server Properties Dialog Box

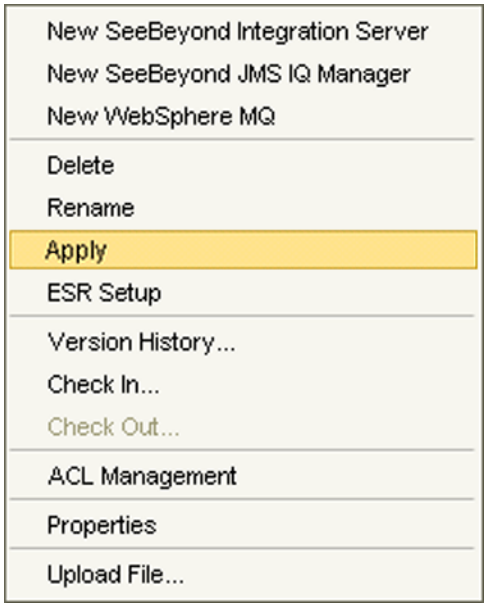


4. Set the value for *Attach Debugger* property to **True** and click **OK**.

5 In Enterprise Explorer, select the Logical Host containing the integration server you have just configured.

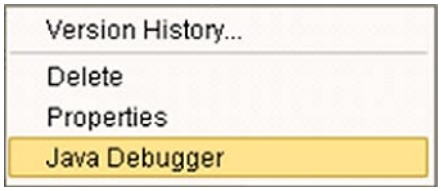
6 Right-click to display the context menu and click **Apply** (see Figure 190).

Figure 190 Logical Host Context Menu



7 The **Java Debugger** option now is enabled in the integration server context menu (see Figure 191).

Figure 191 Integration Server Context Menu

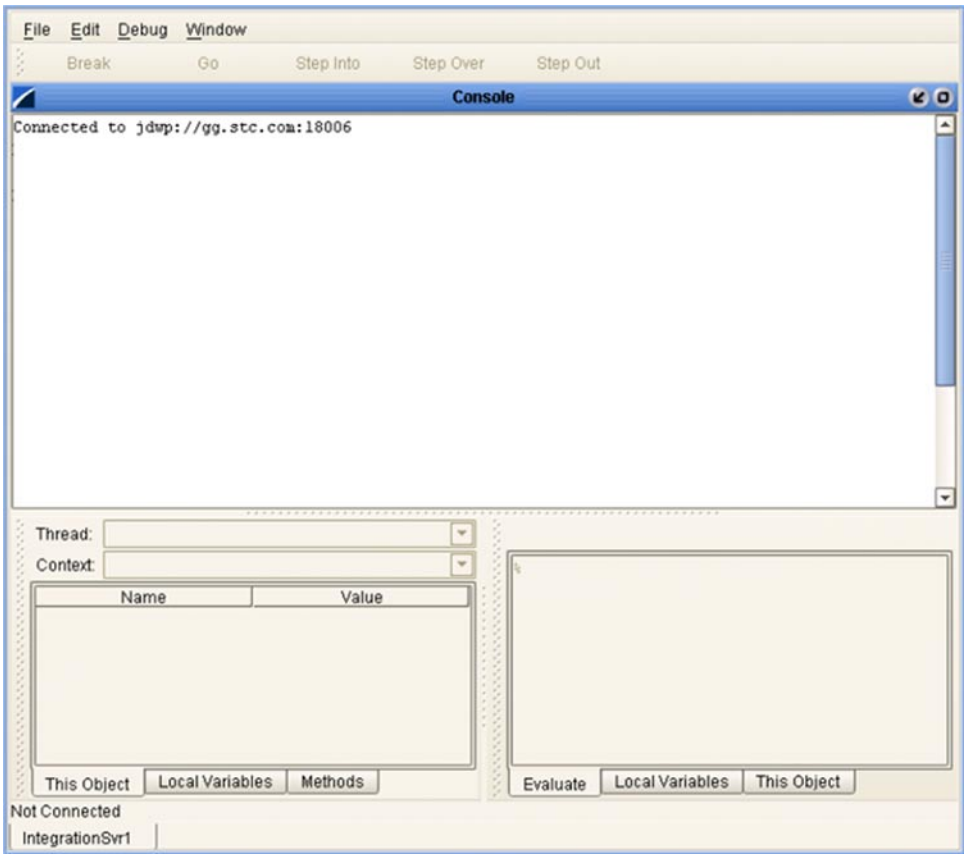


7.12.2 Invoking the Java Debugger

To invoke the Java Debugger

1. In Enterprise Explorer, select the appropriate integration server.
2. Right-click to display the context menu.
3. Click **Java Debugger** (see Figure 191), and the Java Debugger appears in the Enterprise Designer Editor panel (see Figure 192).

Figure 192 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 193), which displays the *Attach to JVM* dialog box (see Figure 194).

Figure 193 File Menu

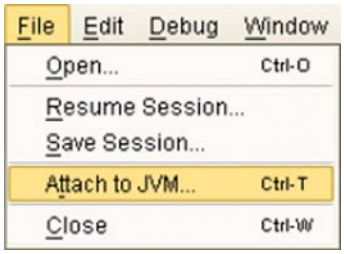
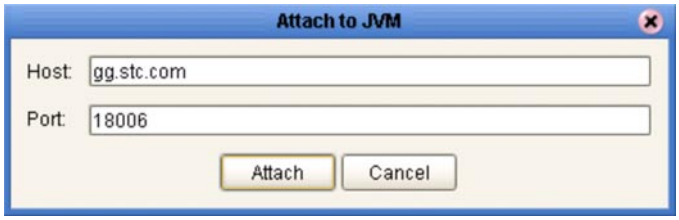


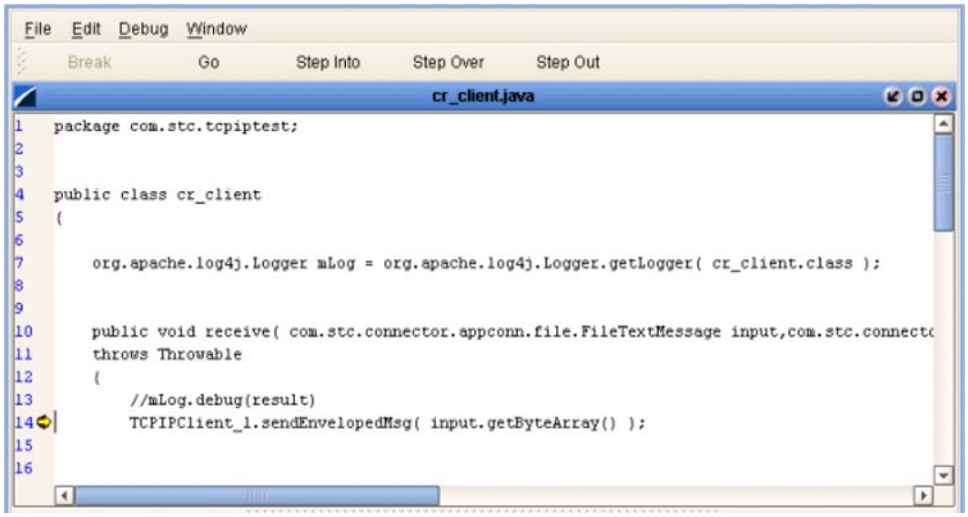
Figure 194 Attach to JVM Dialog Box



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 195).

Figure 195 Collaboration Source Code Display



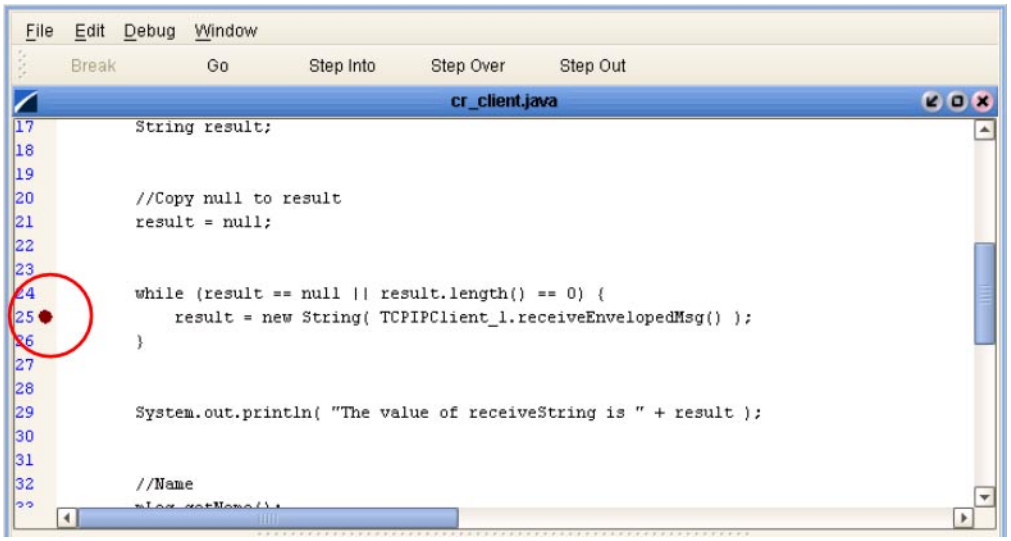
5 You can now set breakpoints to assist in examining and debugging the code.

7.12.3 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 196).

Figure 196 Breakpoint Example



2 Alternatively, you can set stops in a specific class or method, or have the debugger break on an exception. These options are available from the **Debug** menu.

A To set a stop in a method, for example, select **Stop in Method ...** from the Debug menu (see Figure 197). A *Stop in Method* dialog box is displayed, in which you can select the desired method (seeFigure 198).

Figure 197 Debug Menu

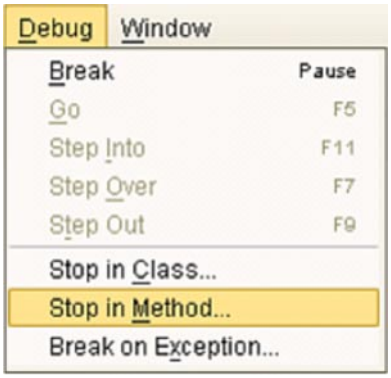
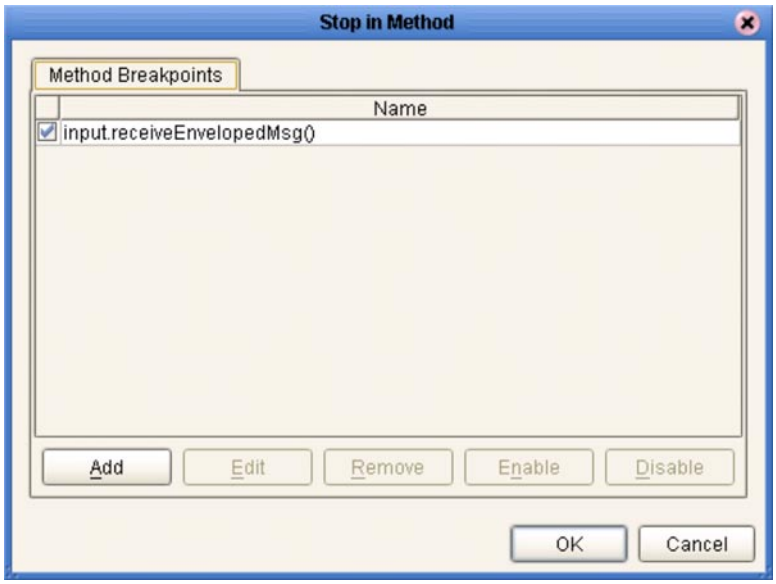


Figure 198 Stop in Method Dialog Box



B To break on an exception, select **Break on Exception ...** from the Debug menu, which displays a *Choose Exception* dialog box (see Figure 199). All occurrences of the specified exception are then trapped and reported (see Figure 200).

Figure 199 Choose Exception Dialog Box

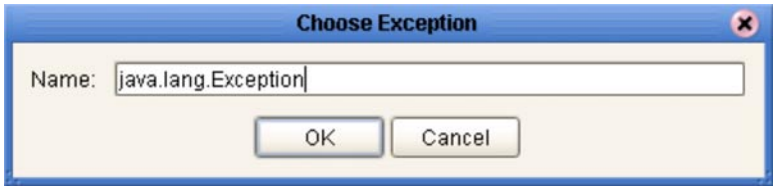
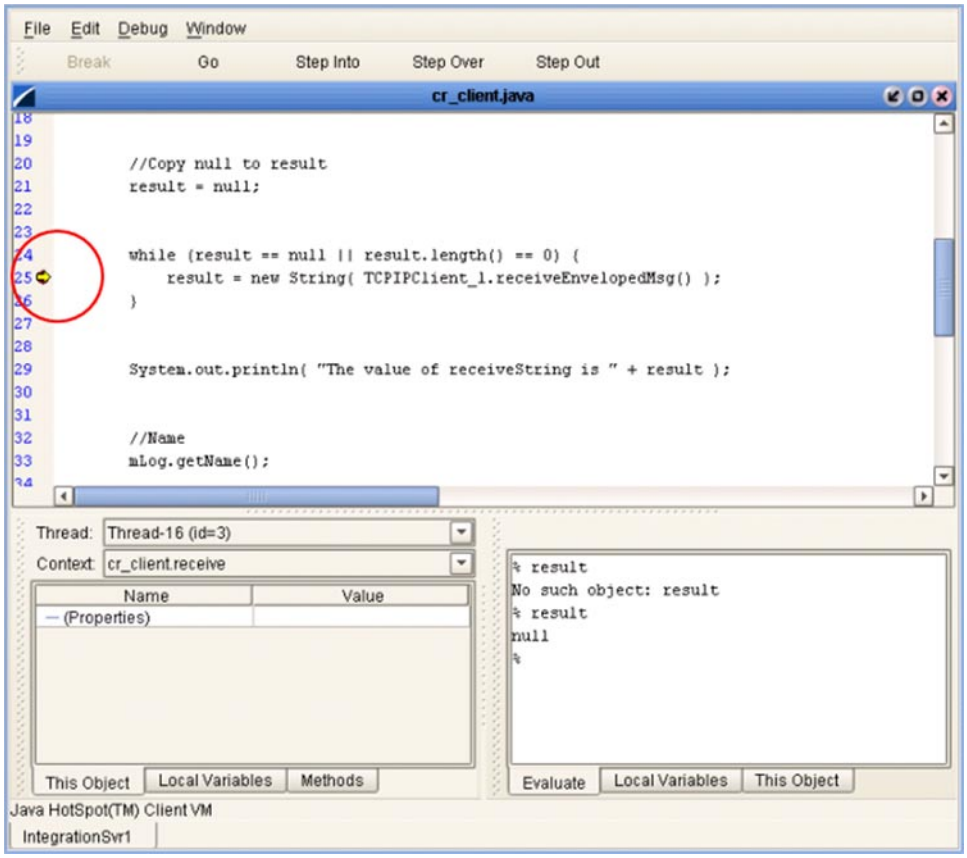


Figure 200 Break on Exception Dialog Box

7.12.4 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 201).

Figure 201 Breakpoint Indicator

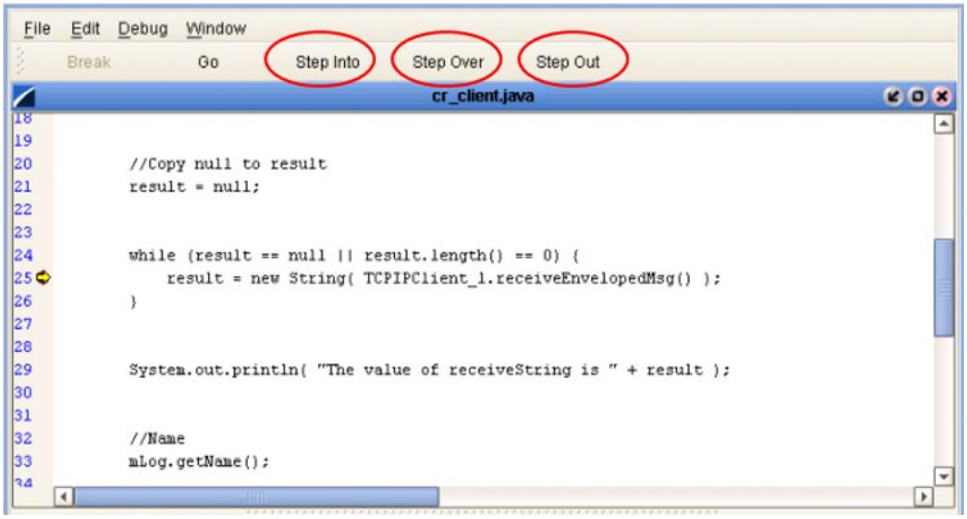


At this point, you can continue by (for example):

- Stepping Into
- Stepping Over
- Stepping Out
- Inspecting a local variable
- Setting a local variable

Stepping Into, Over, or Out

Figure 202 Stepping Into, Over, and Out Commands

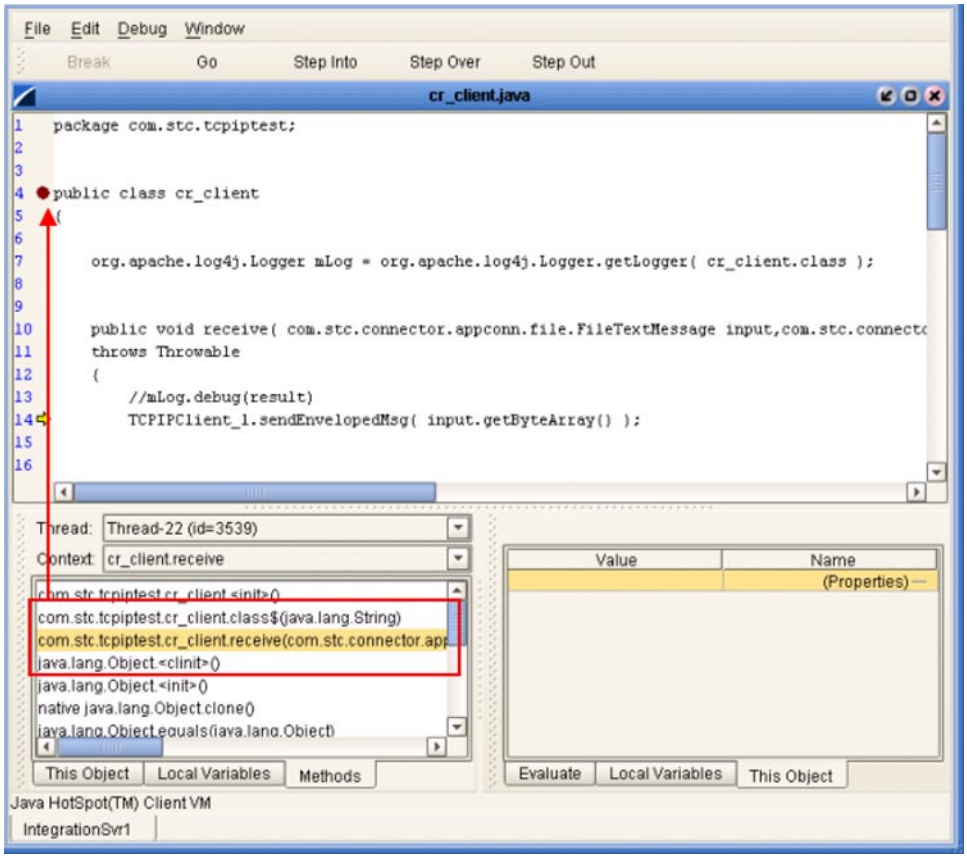


- By selecting the **Step Into** option (see Figure 202), the breakpoint is lifted and execution of the Collaboration will continue, *including* the line of code at the breakpoint.
- By selecting the **Step Over** option, the breakpoint is lifted and execution of the Collaboration will continue, *ignoring* 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 203).

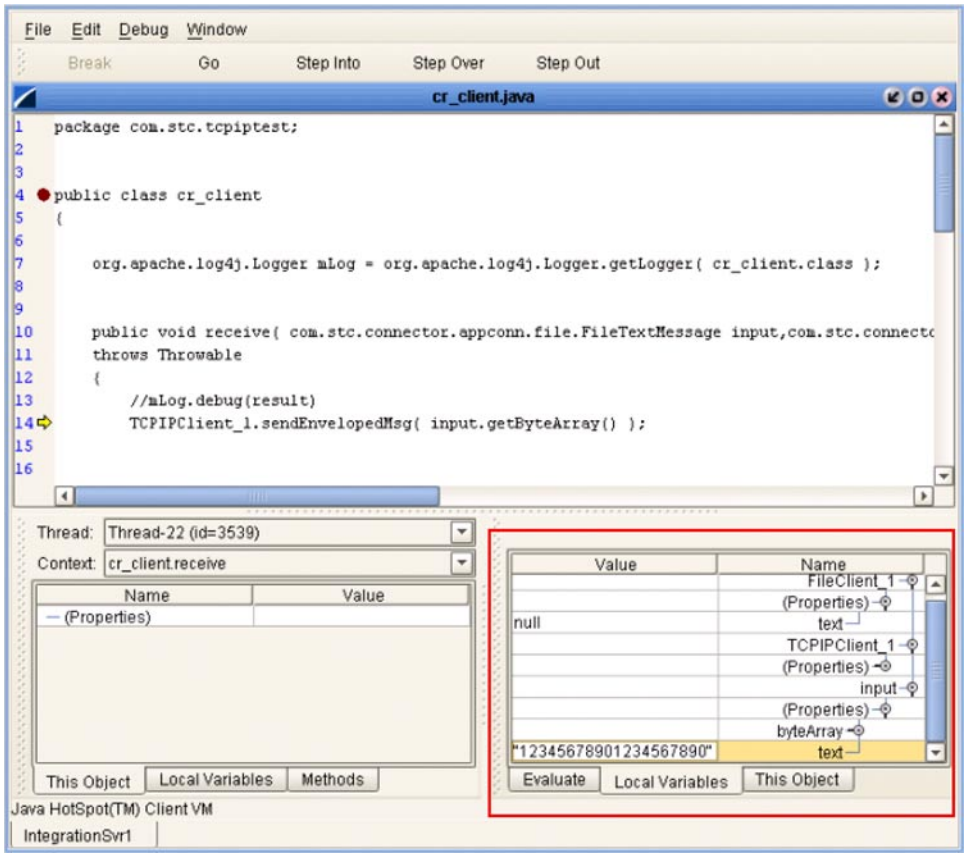
Figure 203 Java Thread and Method Display



Inspecting a Local Variable or Method

You can inspect a local variable by selecting the **Local Variables** tab in the right bottom panel of the debugger (see Figure 204). 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 204 Local Variables Tab

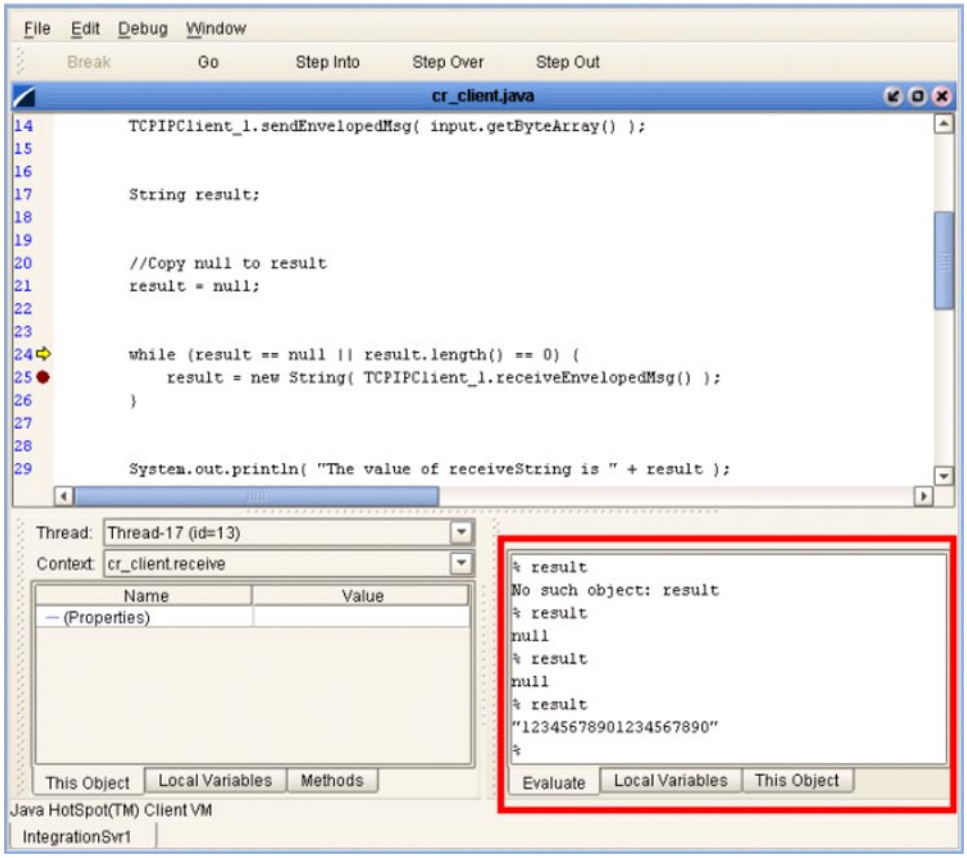


You also can inspect a local variable by selecting the **Evaluate** tab and entering the variable name in the panel using the following syntax:

```
% <variable_name>
```

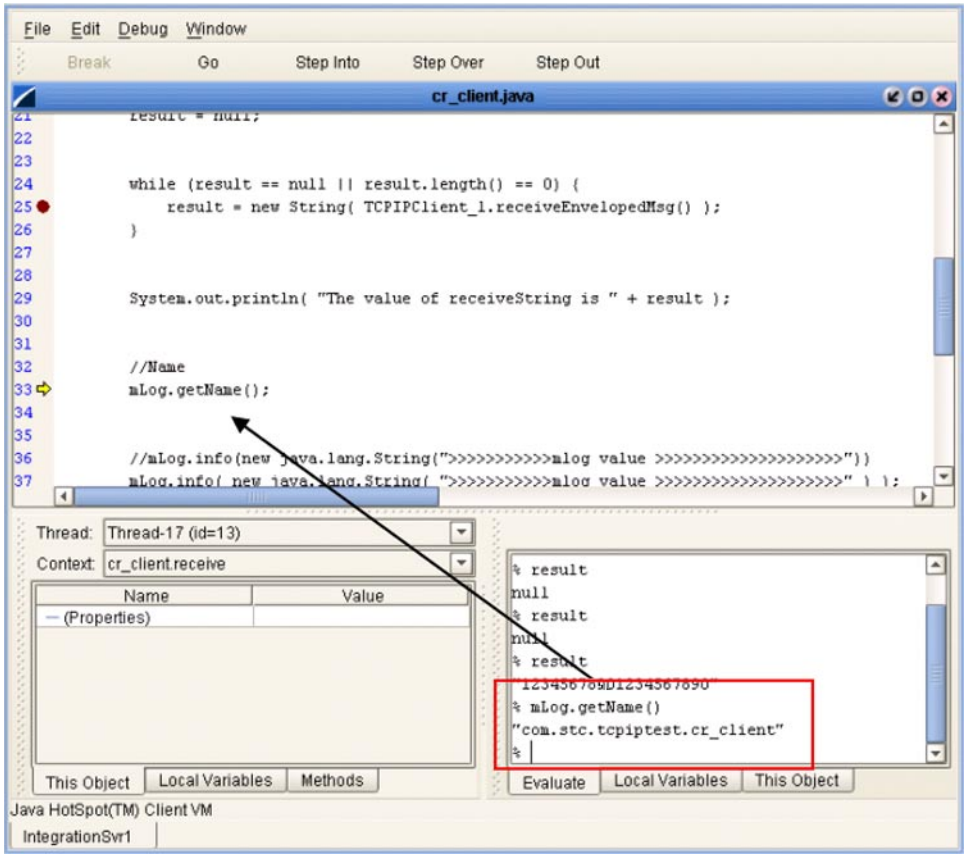
If the code has not initialized a variable, *No such object* is displayed; otherwise, the current value of the variable is displayed (see Figure 205).

Figure 205 Evaluate Local Variable



You can also inspect the result of a method by entering the method name, following the same syntax format (see Figure 206).

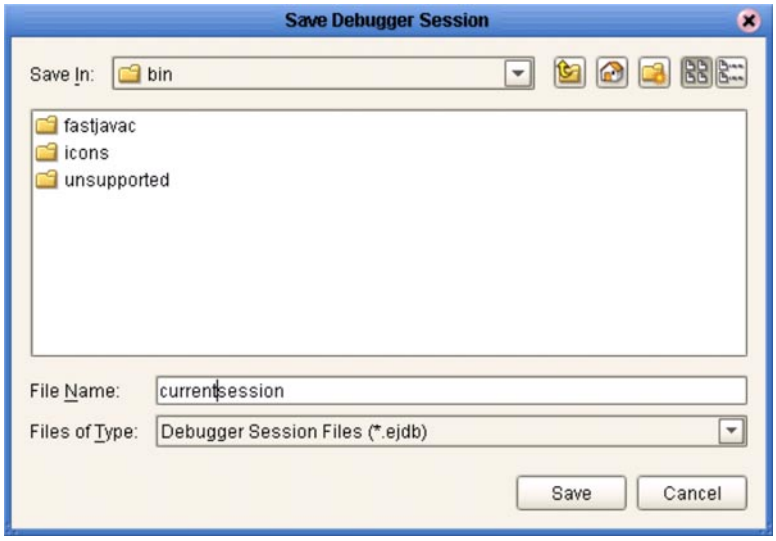
Figure 206 Evaluate Method



Saving and Resuming Debug Sessions

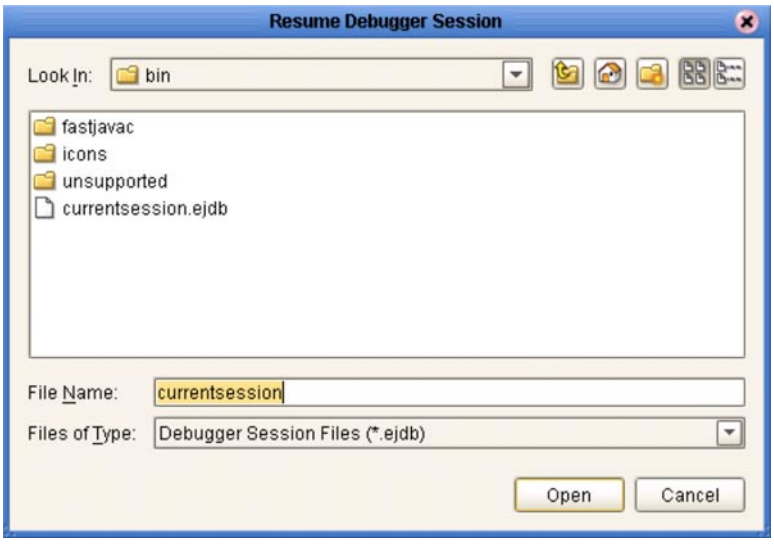
You can pause the debugging process by saving the session to a file. Selecting **Save Session** from the debugger File menu displays the *Save Debugger Session* dialog box (see Figure 207), in which you specify the file name and location.

Figure 207 Save Debugger Session Dialog Box



You can continue a debugging process that was paused by saving to a file. Selecting **Resume Session** from the debugger File menu displays the *Resume Debugger Session* dialog box (see Figure 208), in which you specify the file name and location.

Figure 208 Resume Debugger Session Dialog Box



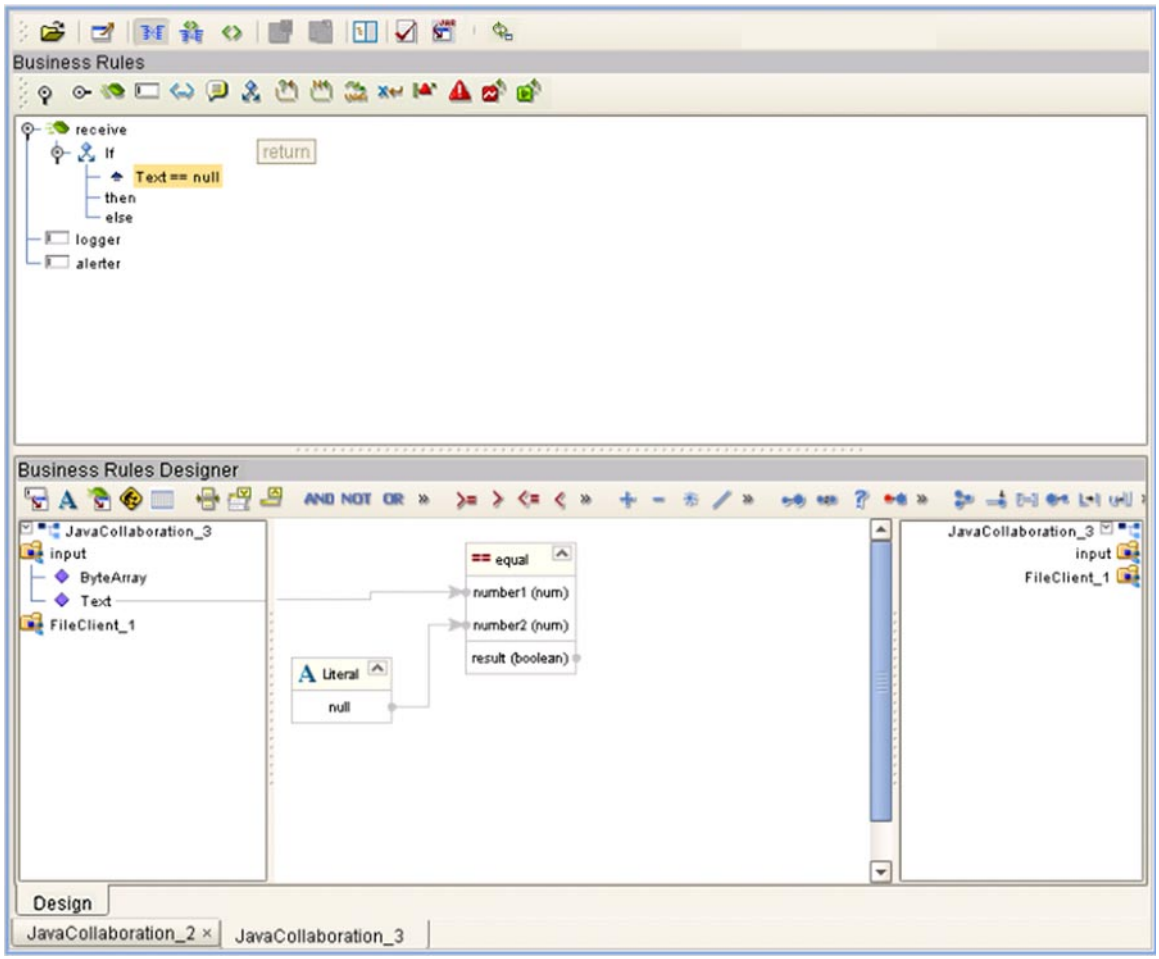
7.13 Creating Alerts

You can create alerts in the Collaboration Editor (Java) by means of the following procedure.

To create alerts using the JCE

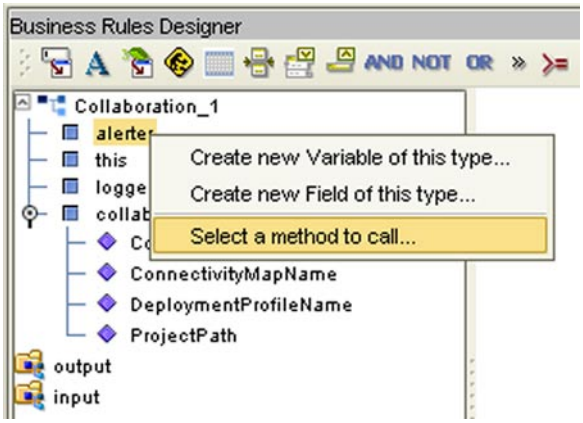
1. Create a new Java-based Collaboration.
2. Add your business rules. As an example, you might perform a test to determine whether or not a file is empty—if it is empty, then raise an alert (see Figure 209).

Figure 209 Empty File Test



3. Initiate an alert object.
4. Right-click the alerter node of the Collaboration in the left pane of the Business Rules Designer, which displays a menu (see Figure 210).

Figure 210 Alert Menu



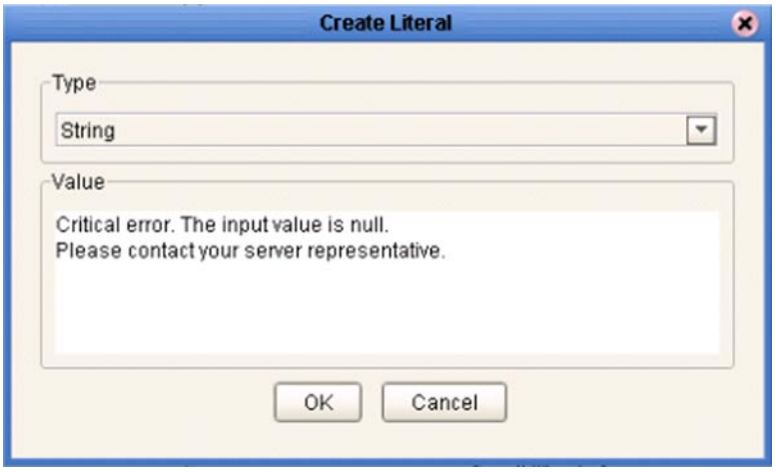
5 Click *Select a method to call ...*

6 Select the severity of the alert from the selection window (see Figure 211). As an example, we will select **critical**.

Figure 211 Alert Severity Selection Window

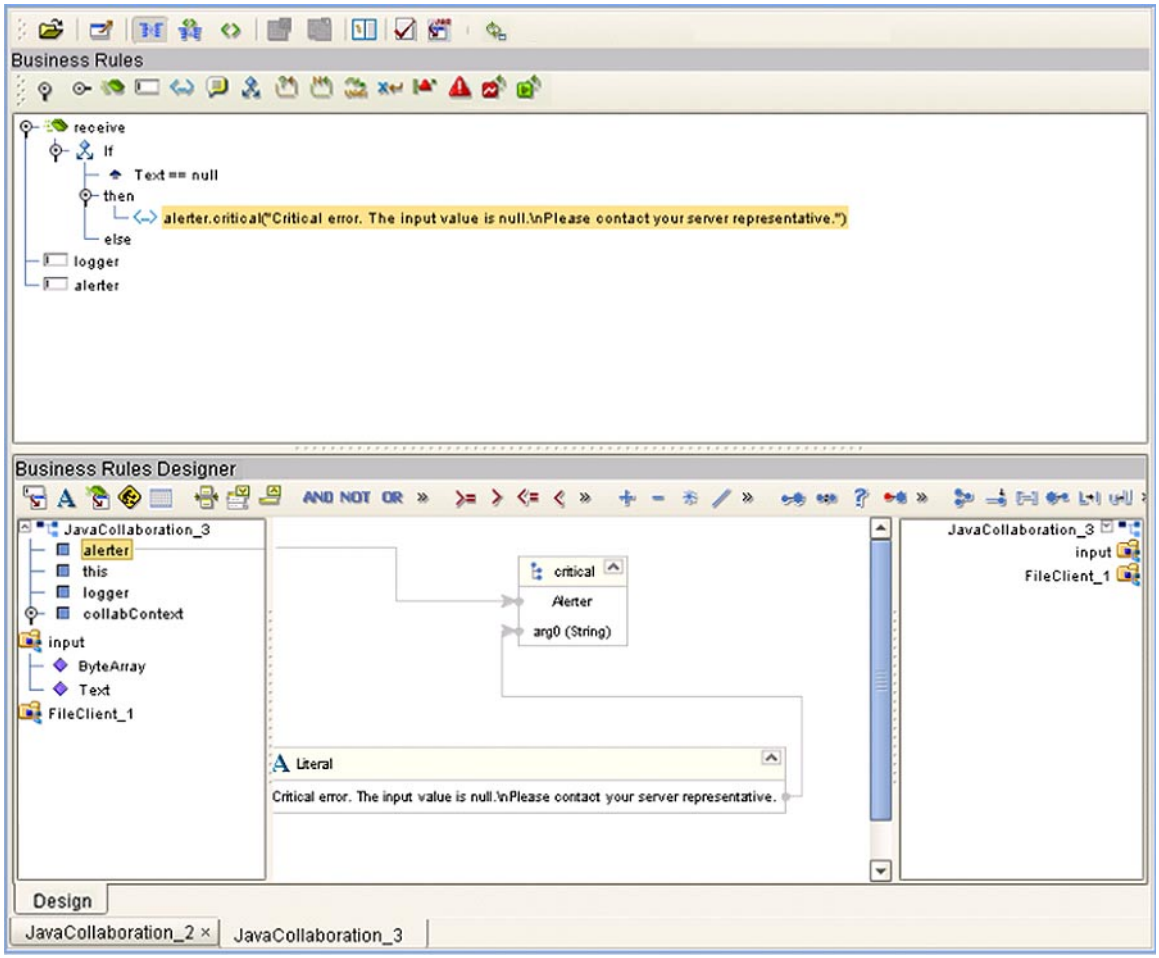
7 Create your alert message, which can be a literal, a constant, or an OTD field name. As an example, we will create a **literal** (see Figure 212).

Figure 212 Create Literal Dialog Box



8 Pass the alert message to the alert event by dragging the message to the argument of the alerter object (see Figure 213).

Figure 213 Pass Alert Message to Object Argument



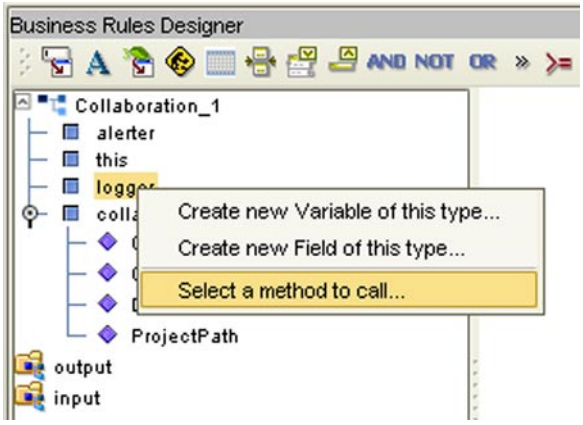
7.14 Creating Log Entries

You can create log file entries in the Collaboration Editor (Java) by means of the following procedure.

To create log entries using the JCE

1. Create a new Collaboration.
2. Add your Collaboration rules and initiate a logging event.
3. Right-click the **logger** node for the Collaboration in the left pane of the Business Rules Designer, which displays a context menu (see Figure 214).

Figure 214 Logging Menu

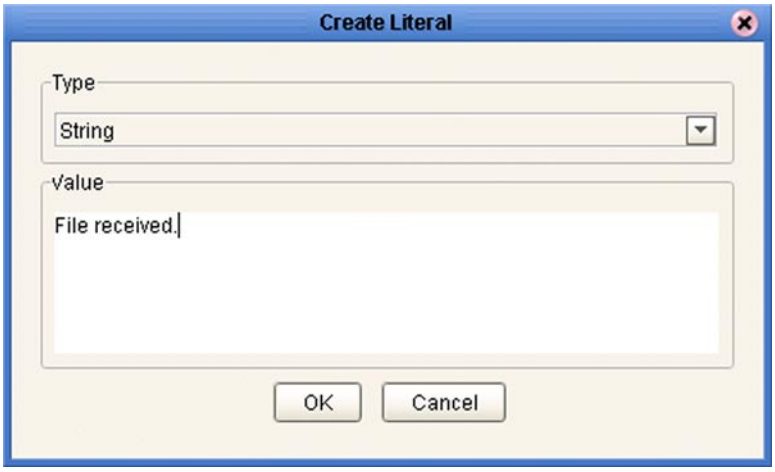


4. Click *Select a method to call ...*
5. Select the logging level with desired method from the pop-up list (see Figure 215). As an example, select **Debug**.

Figure 215 Logging Level/Method List

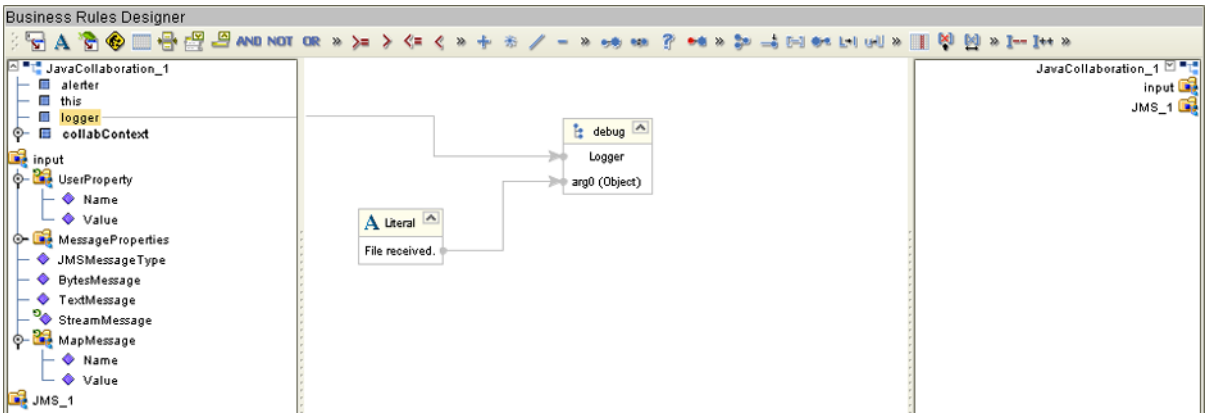
6 Create your log message, which can be a literal, a constant, or an OTD field name. As an example, we will create a **literal** (see Figure 216).

Figure 216 Create Literal Dialog Box



7 Pass the log message to the logging event by dragging the message to the argument of the logging object (see Figure 217).

Figure 217 Pass Log Message to Object Argument



Note: *Compiler errors are logged in the IDE log (\edesigner\usrdir\system\IDE.log), at the WARNING level. See the eGate Integrator System Administration Guide for information regarding logging level settings.*

7.15 Using Byte-Array Converters

Note: *Byte-array converters are currently available only when using either Japanese or Korean localized versions of eGate Integrator.*

Several data-encoding converters are available for byte arrays (byte[]s) within eGate Integrator. To access these converters, you must enable the extended language options as described in **Options Setup** on page 62.

Table 61 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

Table 62 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

7.15.1 Defining Encoding Converter Methods

Encoding converter methods are defined using a dialog box, shown in Figure 218, that is invoked by clicking the **Encoding Converter** icon in the Business Rules Designer toolbar (see **Business Rules Toolbar Icons** on page 182). The options in the dialog box are described in Table 63.

Figure 218 Encoding Converter Methods Dialog Box

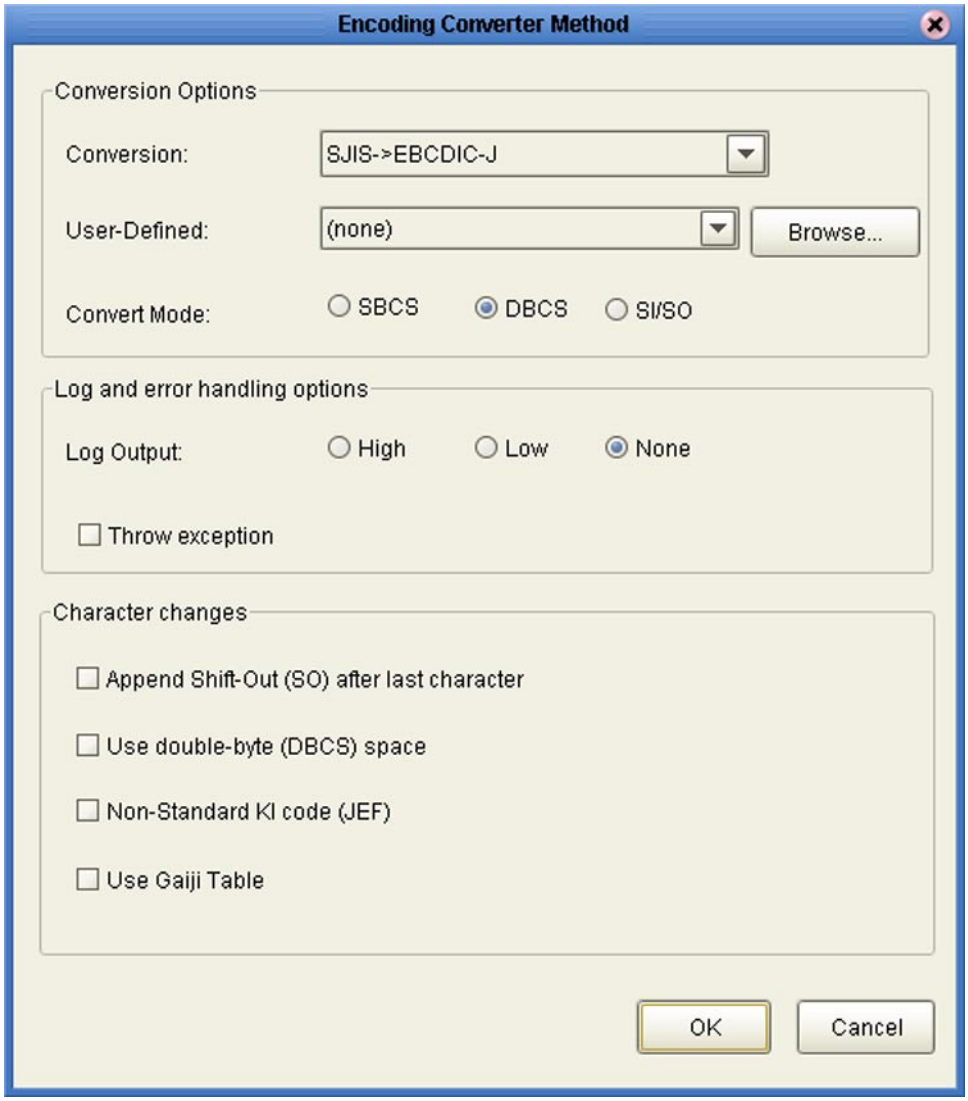


Table 63 Encoding Converter Methods Dialog Box Options

Option	Description
Conversion	Specifies the type of byte-array encoding converter to use (see Table 61 and Table 62 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 the DBCS option is selected for Convert Mode.

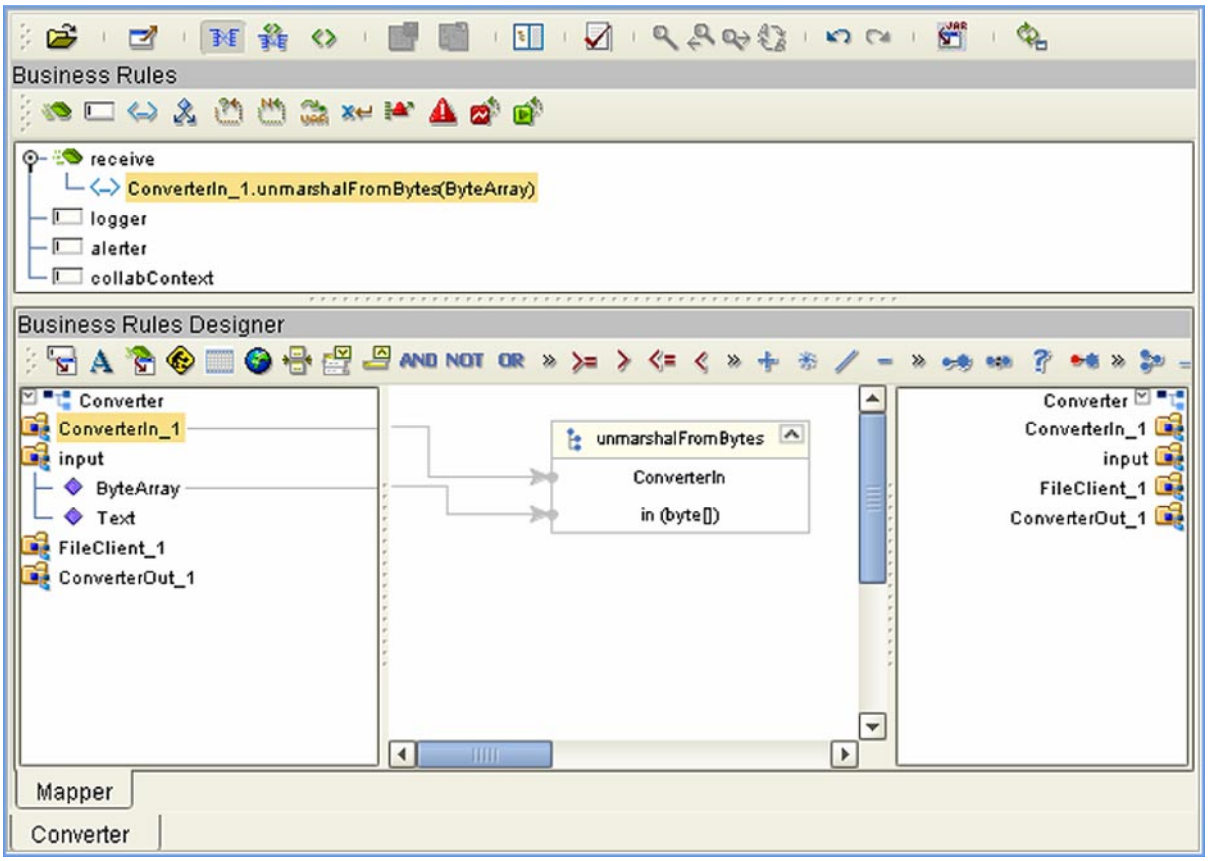
Table 63 Encoding Converter Methods Dialog Box Options

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 Define you business rules to receive the input data as a byte array (see Figure 219).

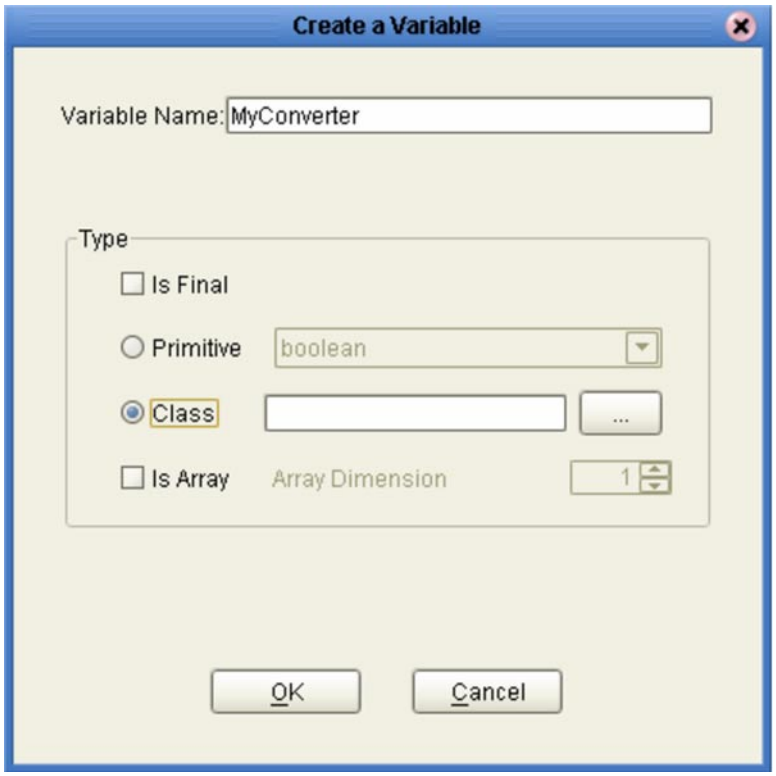
Figure 219 Using *unmarshalFromBytes* Method



2 Create a local variable of the Encoding Converter class.

A Click the **Local Variable** icon to display the *Create a Variable* dialog box (see Figure 220).

Figure 220 Create a Variable Dialog Box

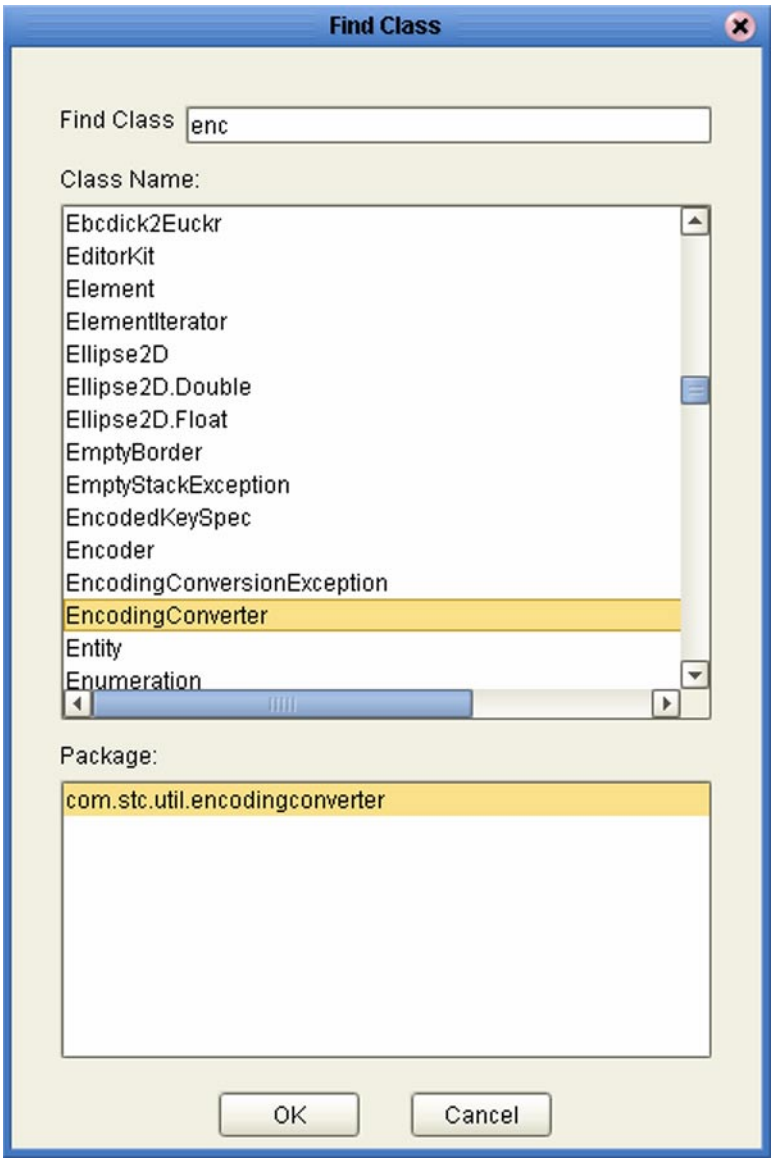


B Enter a name for the variable, for example: **MyConverter**.

C Click the **Class** option button to enable tne entry field.

D Click the ellipsis (**...**) button to display the *Find Class* dialog box (see Figure 221).

Figure 221 Find Class Dialog Box

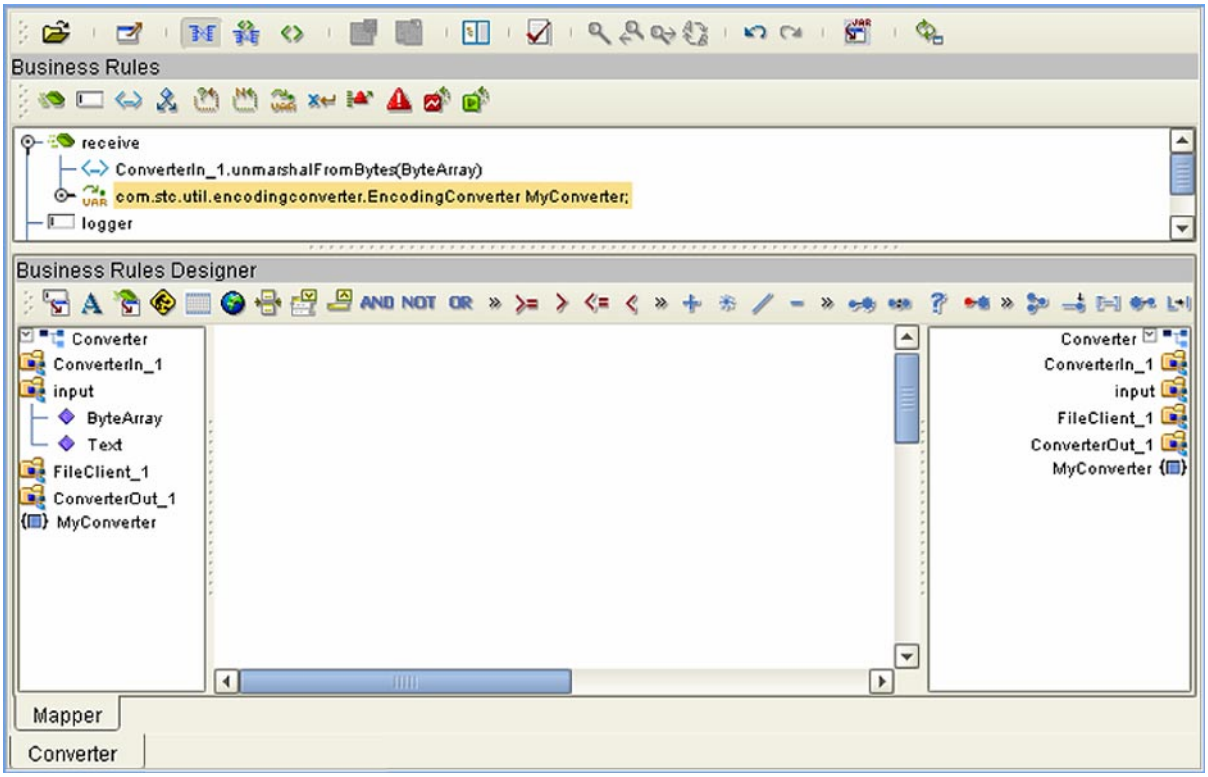


E Scroll to **Encoding Converter** and select it.

F Click **OK** to return to the **Create a Variable** dialog box.

G Click **OK** again to return to the main Collaboration Editor interface, where the local variable is now displayed (see Figure 222).

Figure 222 Local Variable *MyConverter* Added



3 Click the **Encoding Converter** icon (see Figure 223) to display the *Encoding Converter Method* dialog box shown in Figure 224.

Figure 223 Encoding Converter Icon

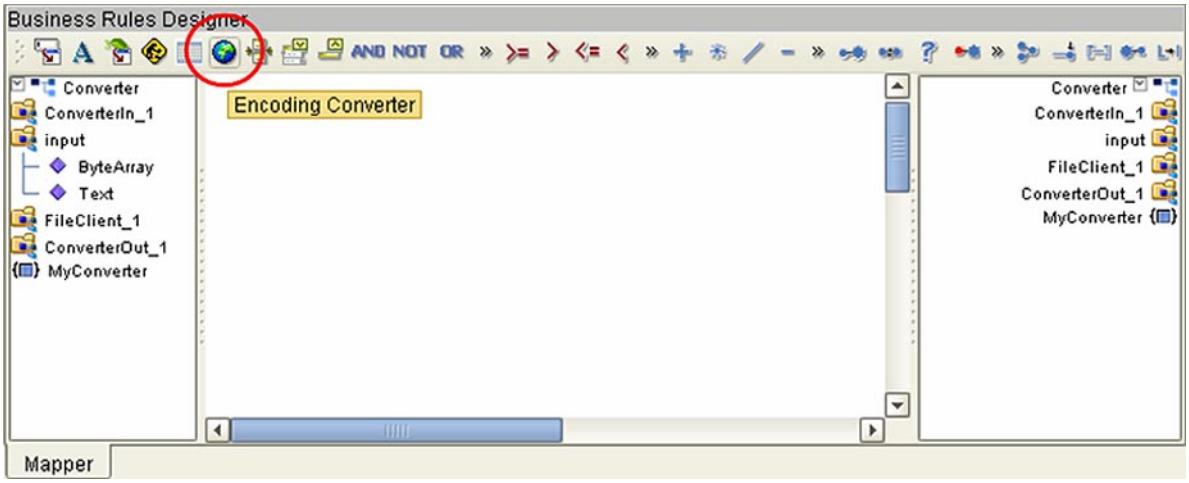
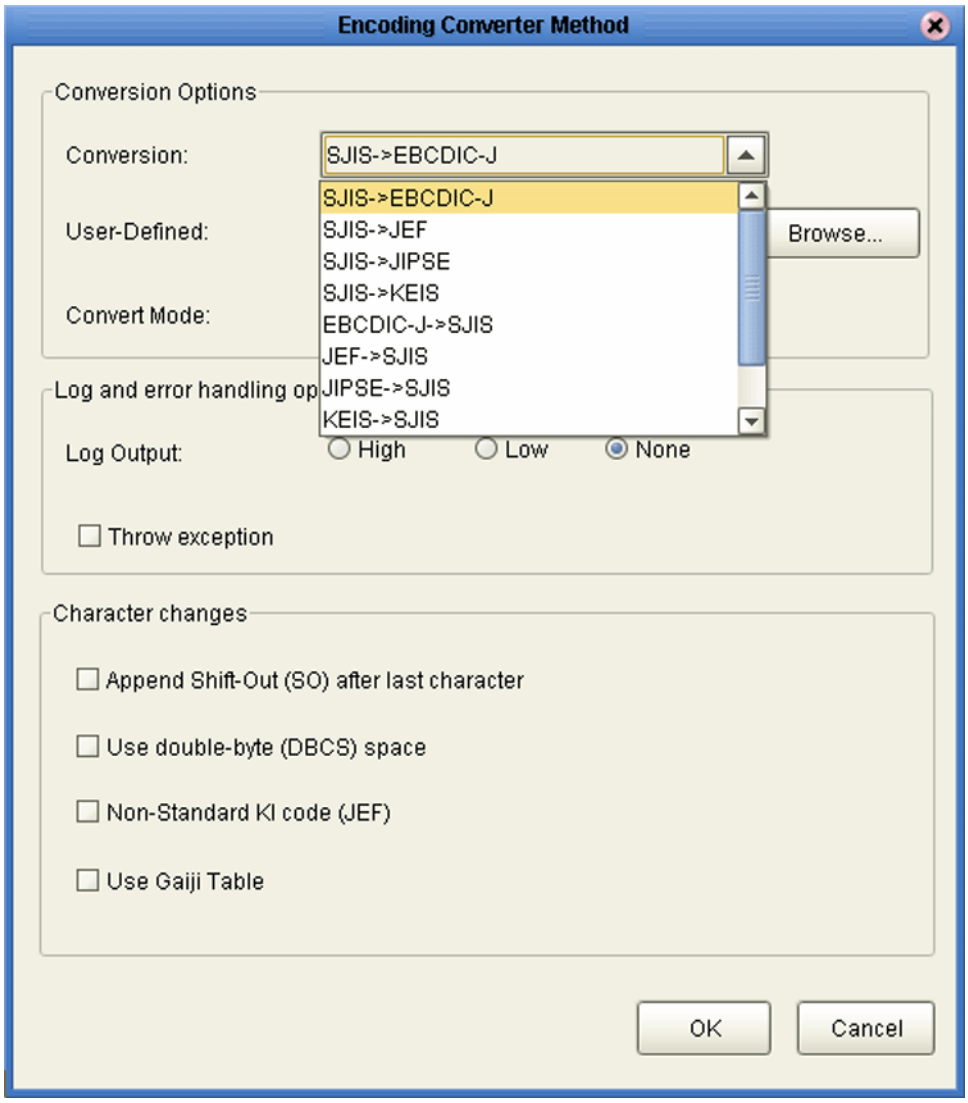


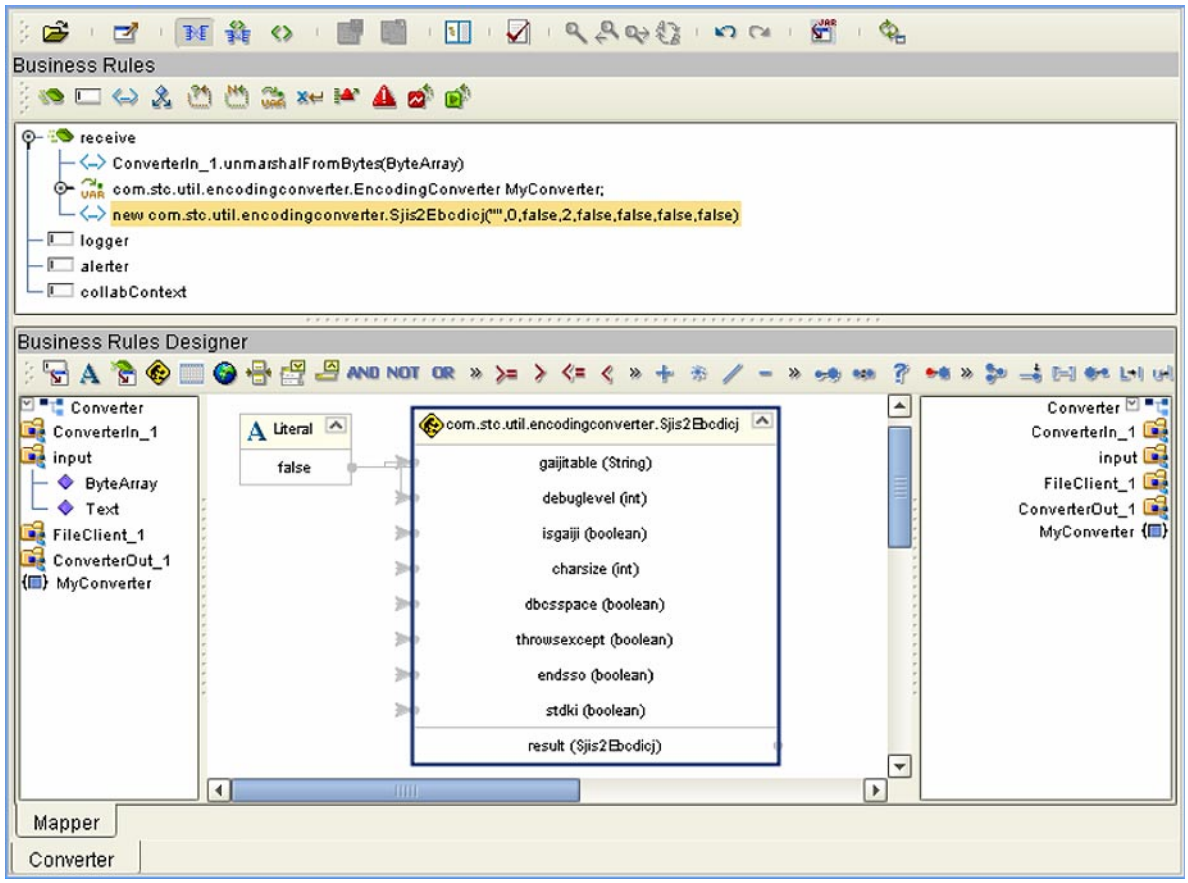
Figure 224 Encoding Converter Method Dialog Box



4 From the Conversion Options group box, select **SJIS->EBCDIC-J**, as shown in Figure 224, which creates the encoding converter rule shown in Figure 225.

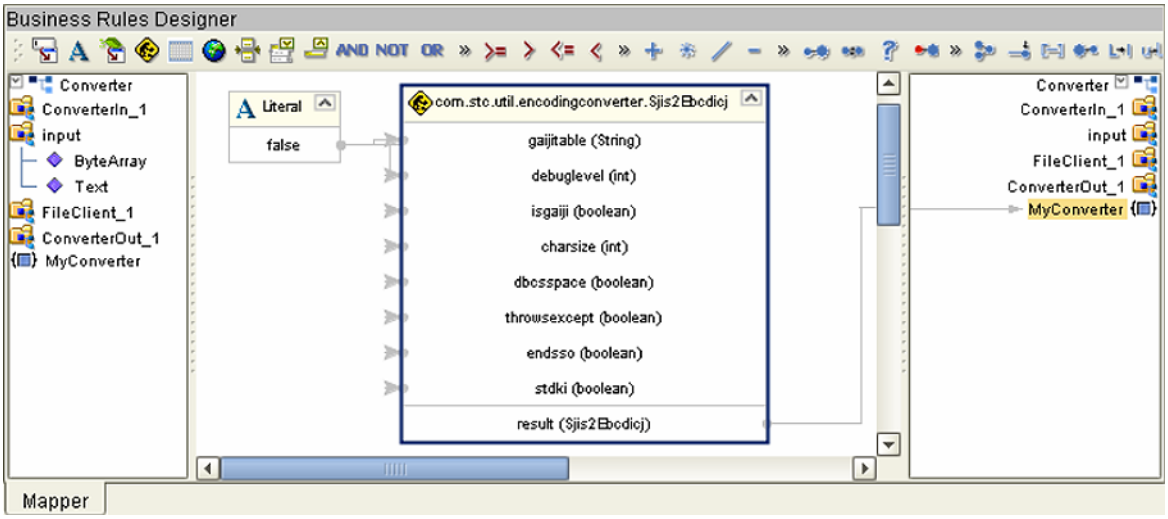
5 Select any desired options from the other group boxes and click **OK**.

Figure 225 Encoding Converter Rule Created



6 Map the result to the local variable created in step 2 (**MyConverter**) in the right pane as shown in Figure 226.

Figure 226 Link Rule Result to Variable



7 Click on the variable **MyConverter** in the left pane to display its context menu and click **Select a method to call** (see Figure 227). This displays the list box shown in Figure 228.

Figure 227 Select a Method to Call

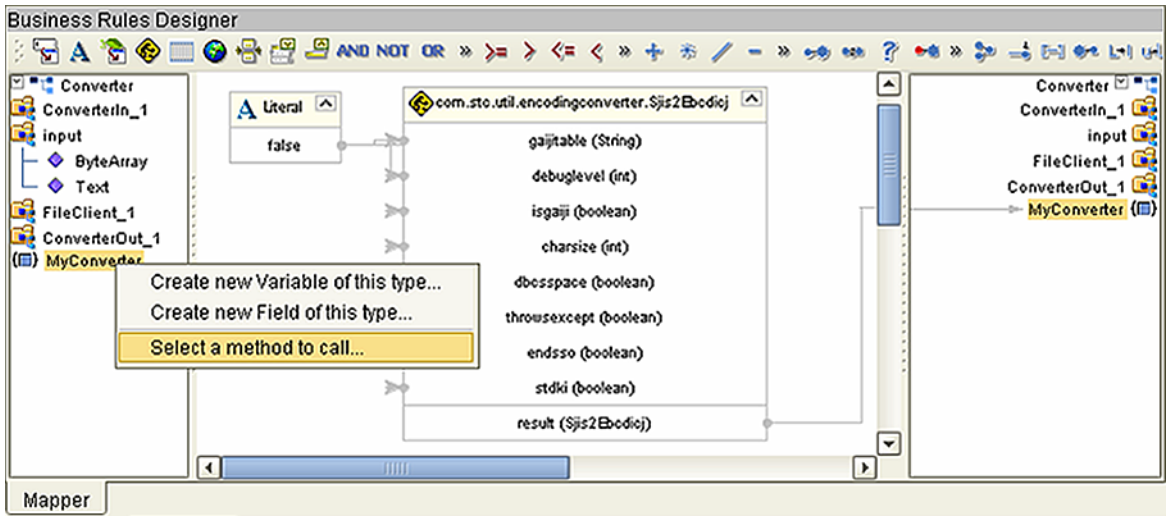
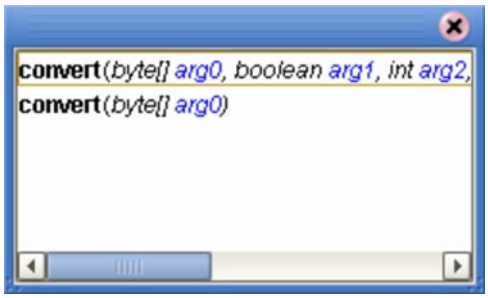
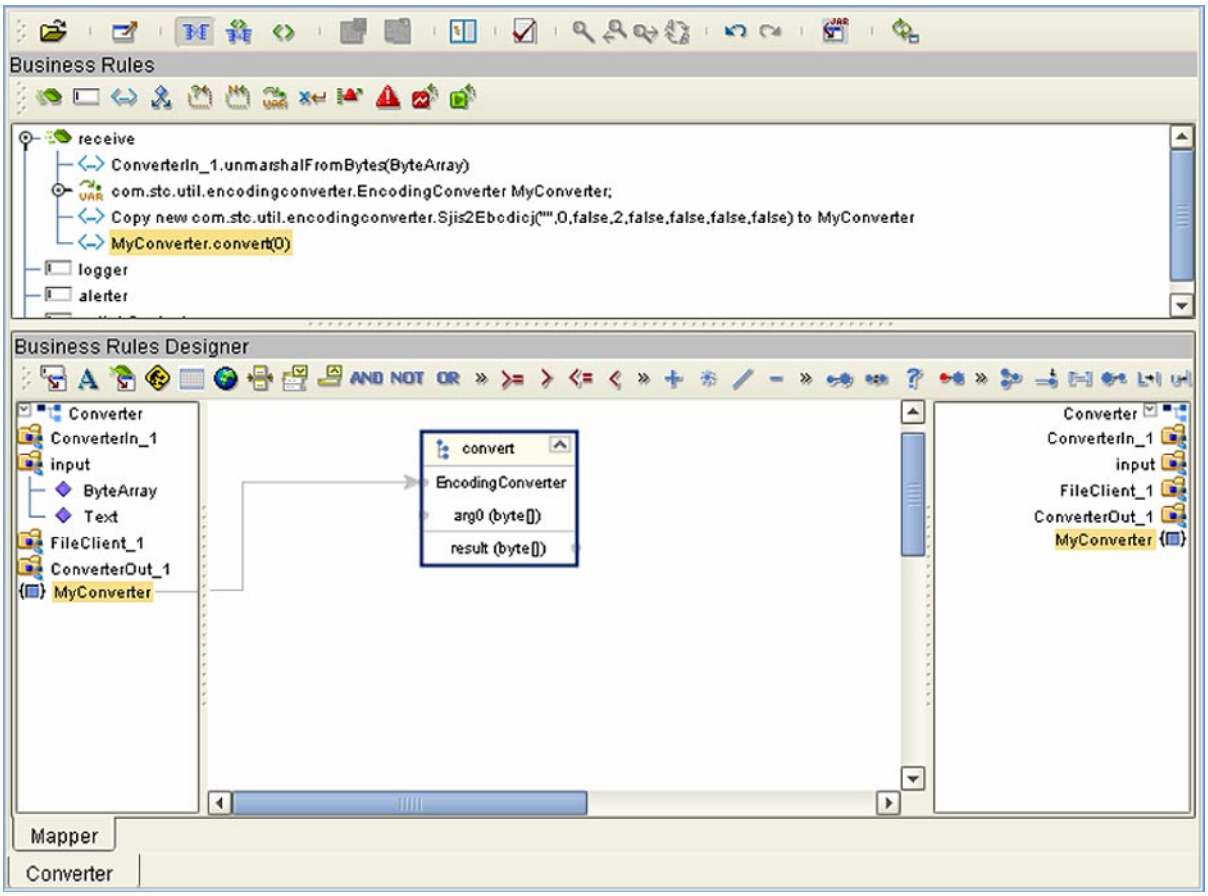


Figure 228 Select Method List



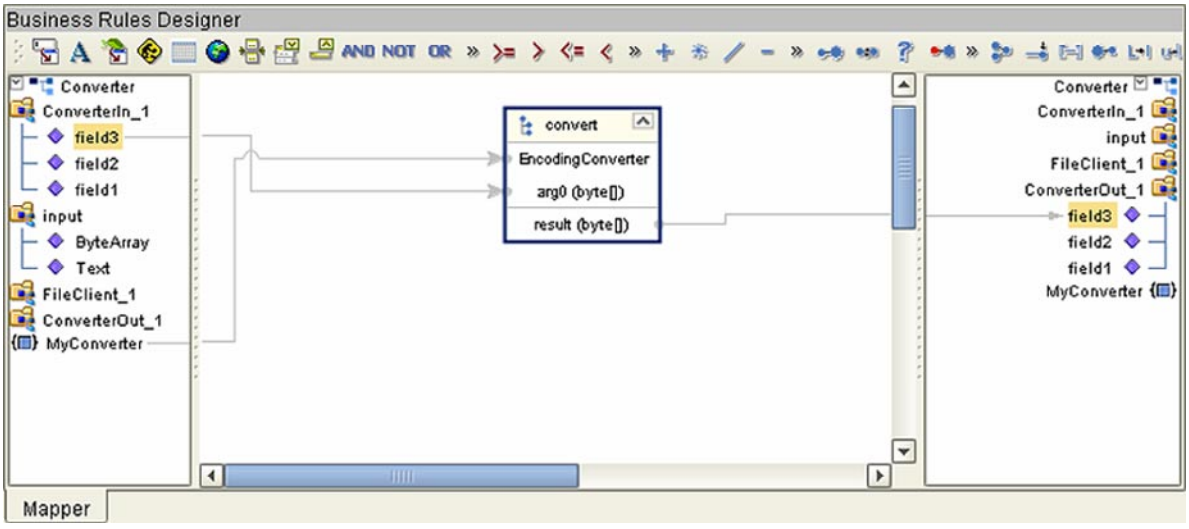
8 Select **convert**(*byte[] arg0)*, which displays the method box shown in Figure 229.

Figure 229 Convert Method Box



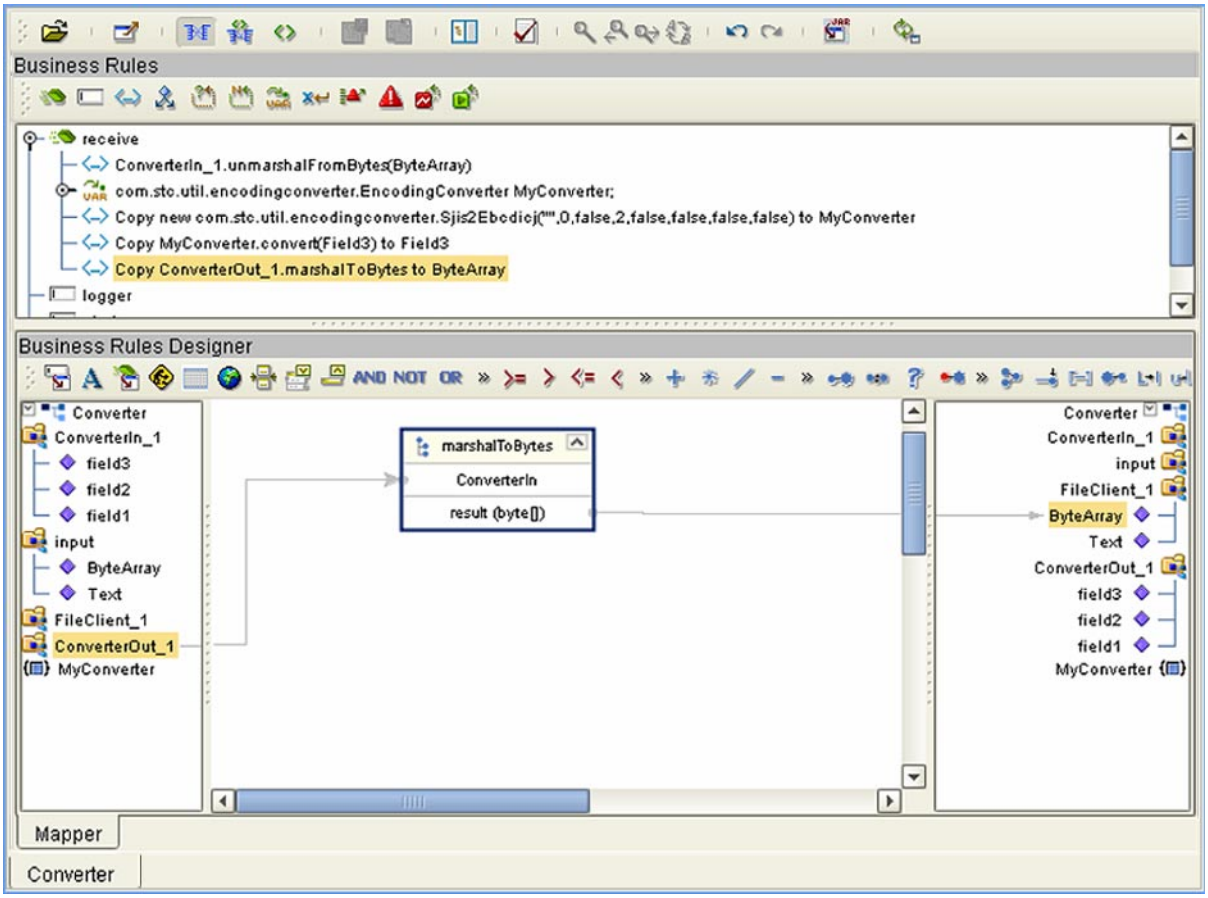
9 Map the node that contains data to be converted (**ConverterIn_1/field3**) to **arg0** (see Figure 230), and map the **result** to the node that gets the converted data (**ConverterOut_1/field3**).

Figure 230 Data Mapping



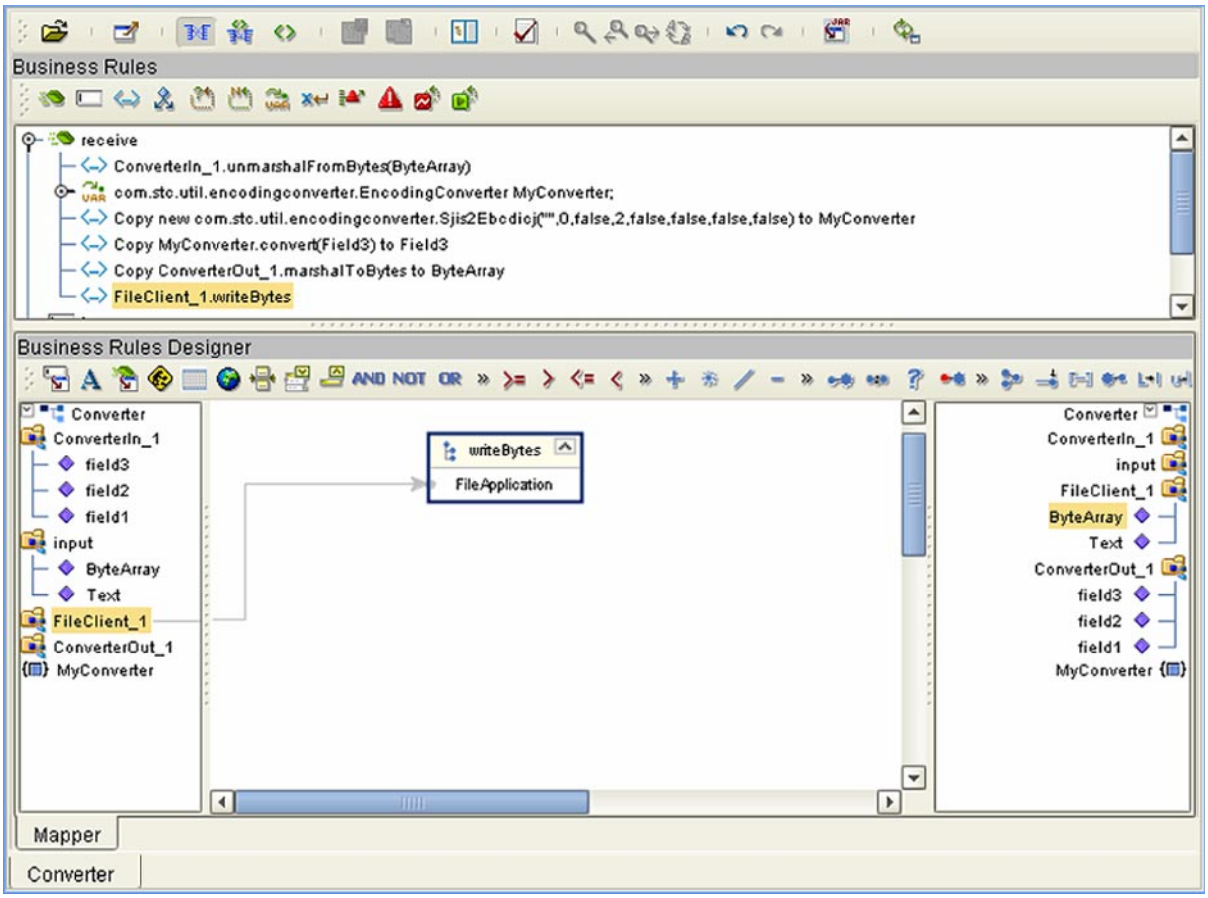
10 When finished with the Collaboration rules, marshal the output data as shown in Figure 231.

Figure 231 Marshal Output Data



11 Finally, output the data as shown in Figure 232.

Figure 232 Output Data



Chapter 8

Collaboration Definitions (XSLT)

This chapter describes the process for building XSLT-based Collaboration Definitions.

8.1 Overview

Collaborations use Collaboration Definitions to define how data should be routed between Project components. Collaborations also define how databases should be queried in response to requests and how APIs to one or more applications should be invoked. Collaborations are used when data translation is required.

The Enterprise Designer includes two tools, the Collaboration Definition Wizard (XSLT) and Collaboration Editor (XSLT), that are used to create and customize your XSLT-based Collaboration Definitions. You must have OTDs available to use as the foundation for creating an XSLT-based Collaboration Definition. See **Object Type Definitions** on page 114 for more details.

Important: *If you delete an OTD in the Project Explorer, any XSLT-based 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* **Impact Analyzer** *on page 81**).*

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.*

8.2 Using the Collaboration Definition Wizard (XSLT)

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). The user interface is highly self-explanatory, but details of the navigation buttons are listed in Table 64 for your reference.

Table 64 Wizard Navigation Buttons

Button	Function
< Back	Returns to the previous step in the wizard. This button is disabled on the first step.
Next >	Goes to the next step in the wizard. This button is disabled on the last step.
Finish	Saves all Collaboration Definition settings and closes the wizard. This button is only enabled on the last step.
Cancel	Closes the wizard without saving the Collaboration Definition.
Help	Displays the online help documentation for the Collaboration Definition Wizard dialog box.

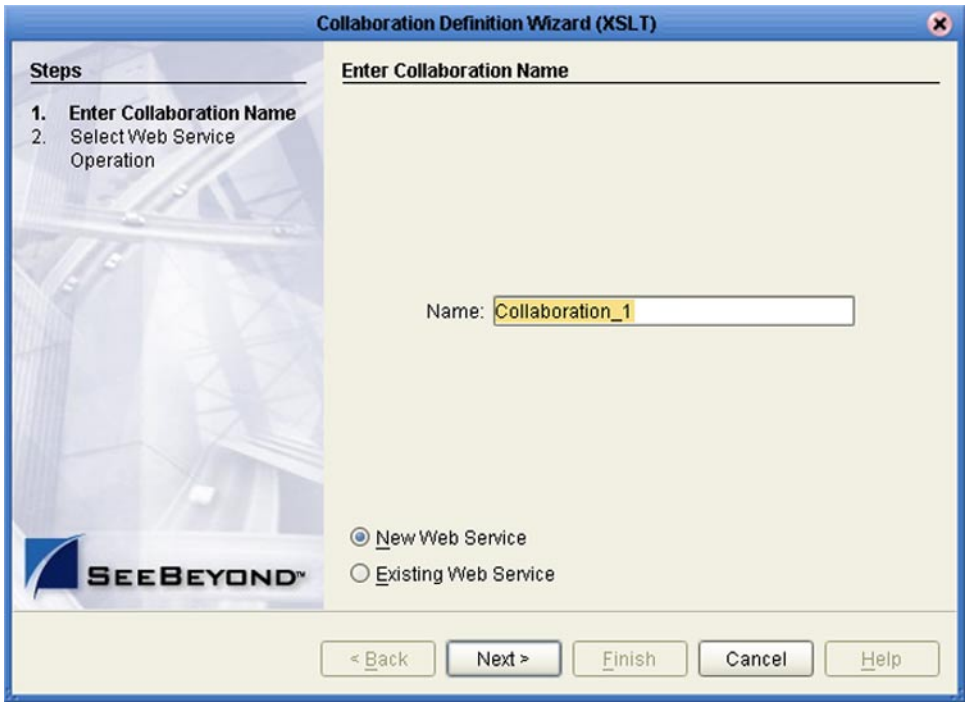
8.2.1 Creating a Collaboration Definition (XSLT)

1 Right-click on a Project in the Enterprise Explorer to display the Project context menu.

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

3 Enter a **Name** for your Collaboration, as shown in Figure 233.

Figure 233 Collaboration Wizard (XSLT) Dialog Box



4 Select a Web service, which can be either:

- A New Web Service.
- An Existing Web Service (for example, an eInsight process or a Java Web Service Operation).

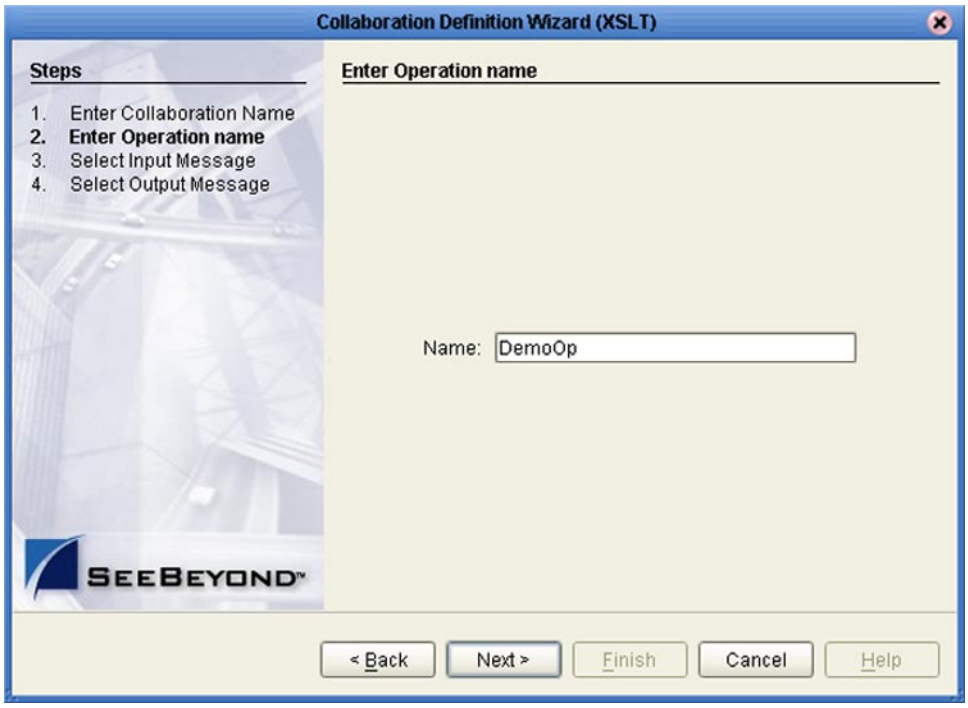
5 Click **Next** to proceed to the next Wizard dialog, which is dependent upon your Web Service selection.

New Web Service

If you selected a New Web Service, you will be presented with the following set of Wizard dialogs.

1 Enter an operation name, as shown in Figure 234. This will become the *method* that can be used to invoke the XSLT-based Collaboration as a Web service.

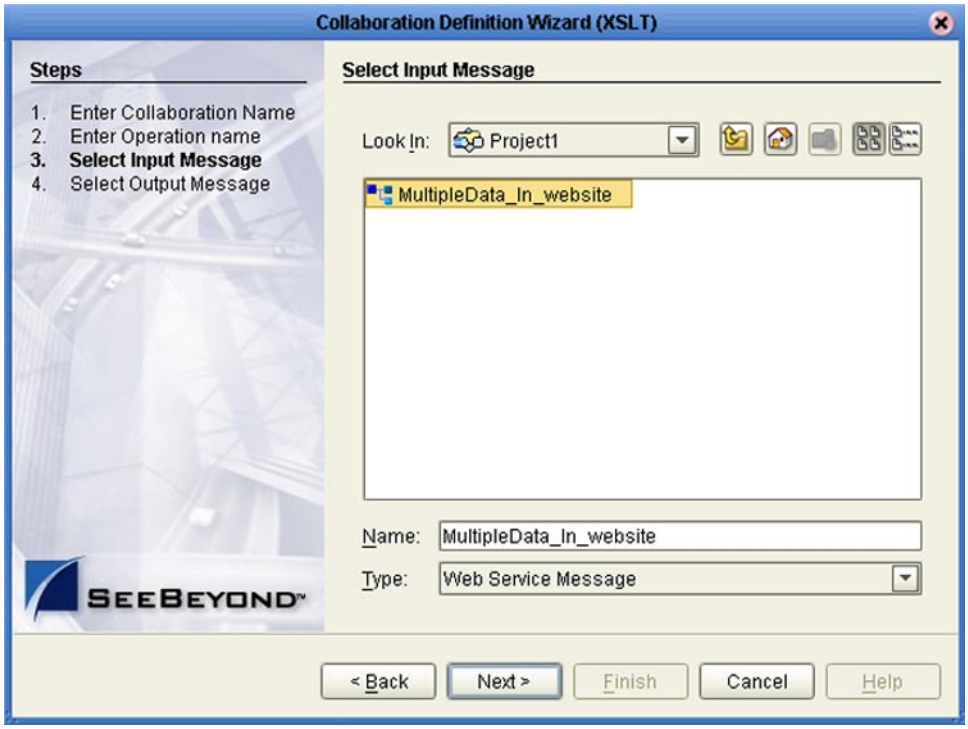
Figure 234 New Web Service: Operation Name



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

3 Select the input Web service message, as shown in Figure 235.

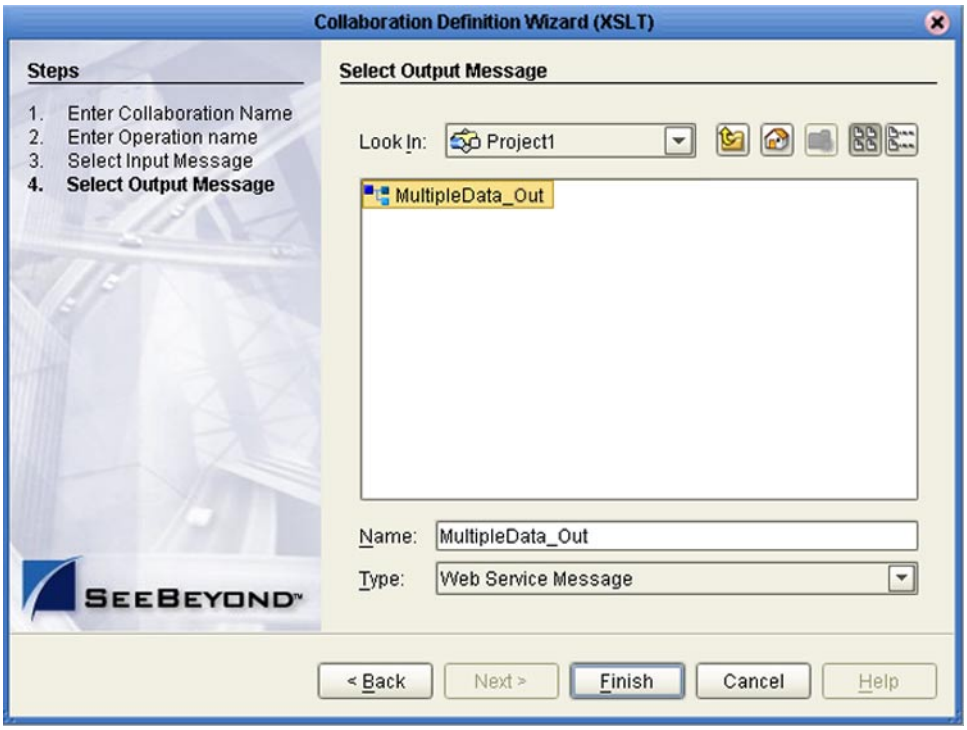
Figure 235 New Web Service: Input Message



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

5 Select the output Web service message, as shown in Figure 236.

Figure 236 New Web Service: Output Message



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

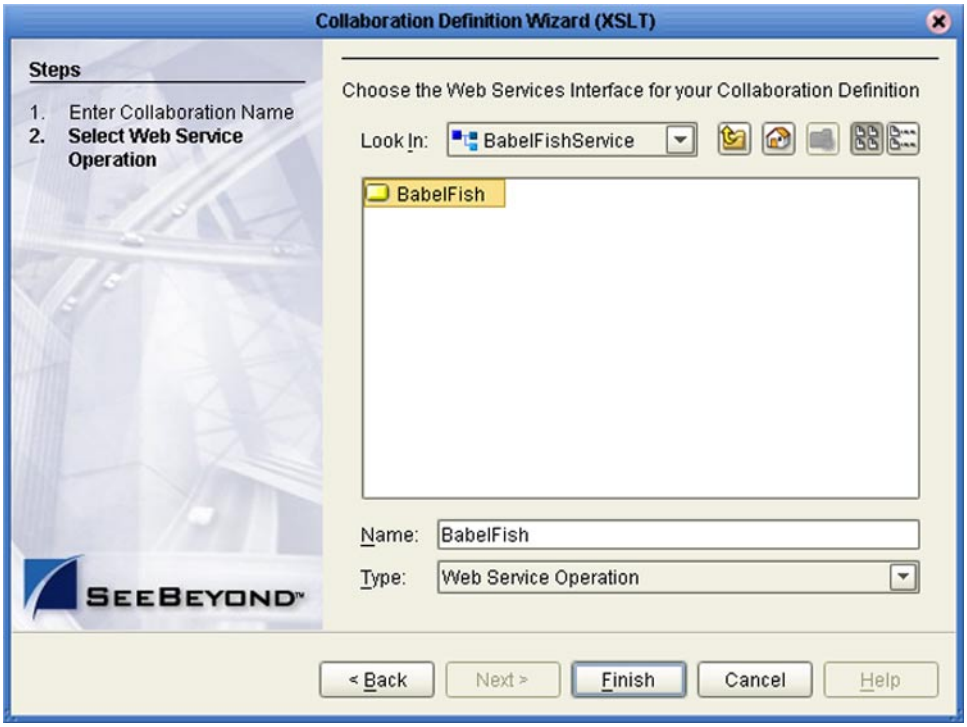
Existing Web Service

If you selected an Existing Web Service, you will be presented with the Wizard dialog shown in Figure 237.

1 Select a Web service operation, which can be either:

- An installed ICAN Web Service.
- A custom Web Service (for example, something that has been created in an eGate Project).

Figure 237 Existing Web Service: Select Operation

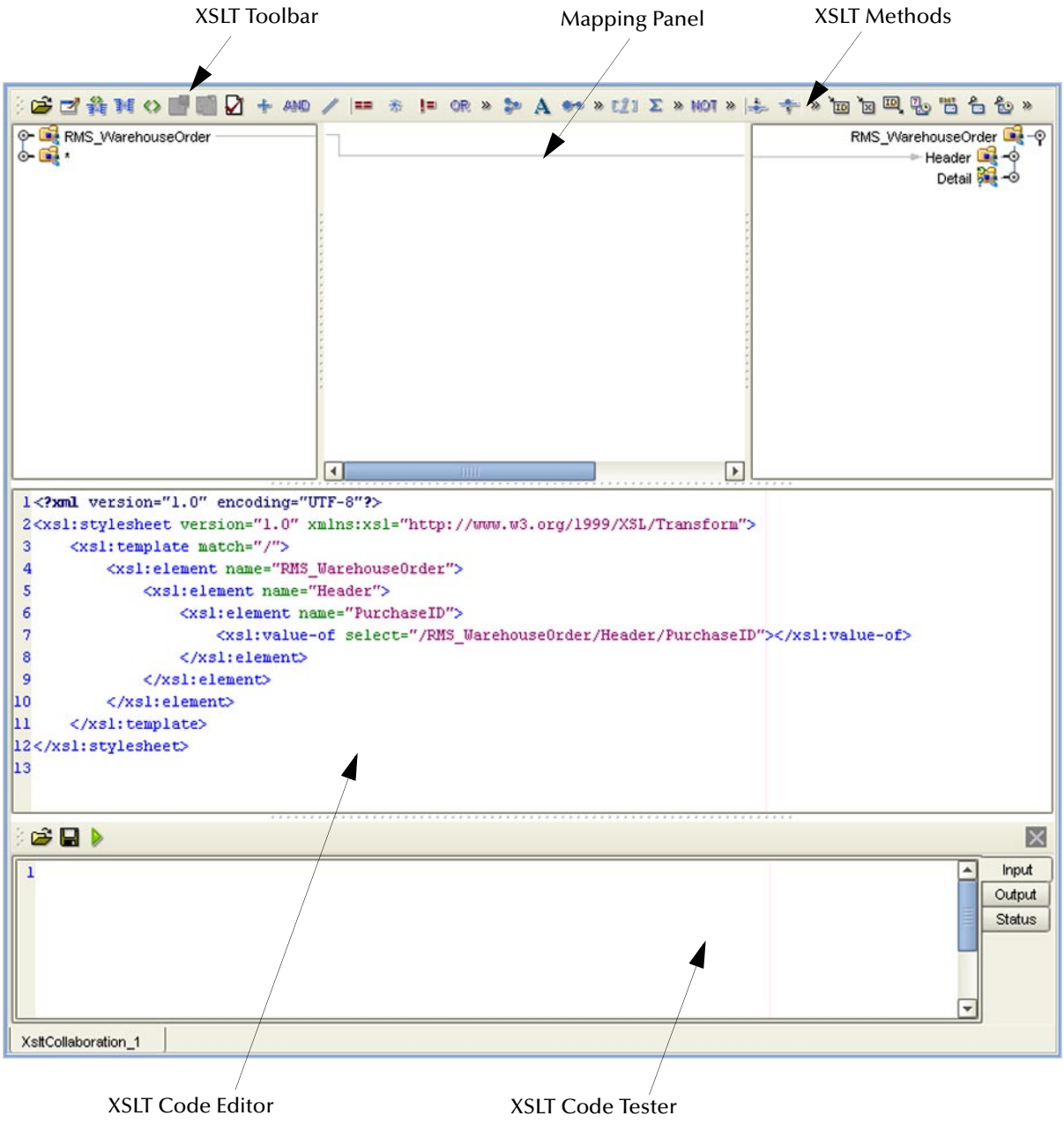


2 Click **Finish** to proceed to the Collaboration Editor (XSLT).

8.3 Using the Collaboration Editor (XSLT)

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 238.

Figure 238 Collaboration Editor (XSLT)



You can also invoke the Collaboration Editor (XSLT) by selecting **Open** in the context menu for an existing XSLT-based Collaboration Definition in the Enterprise Explorer.

The XSLT Mapping panel is used to map fields and add methods to the Collaboration Definition. At the top left of the Mapping panel is the toolbar, containing icons as described in Table 65. 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.

8.3.1 XSLT Toolbar Icons

Table 65 XSLT Toolbar Icons

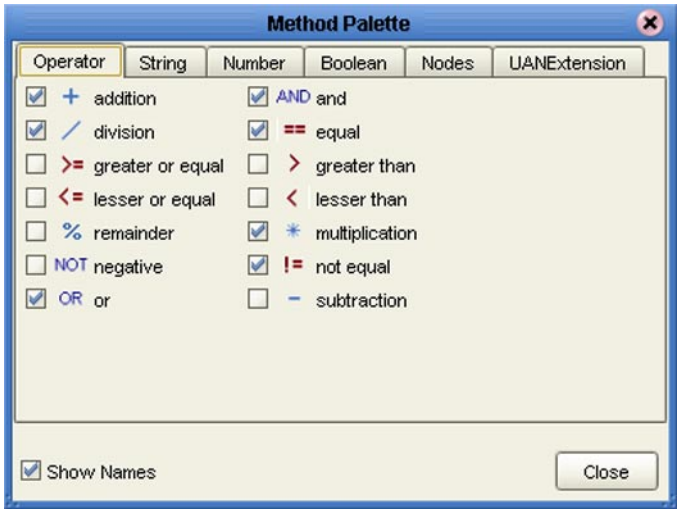
Icon	Command	Function
	Import XSLT from a Local File	Displays the **Open** dialog box, which you can use to locate and select an XSL file to import.
	Save XSLT to a Local File	Displays the **Save** dialog box, which you can use to save the selected XSLT-based Collaboration Definition to a file.
	Show 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.

Collaboration Method Palette (XSLT)

The Collaboration Method Palette includes a series of method icons that you can drag onto the Mapping area. Click the Chevron (>>) to the right of the method groups to display the dialog box shown in Figure 239.

Select a check box to add the method to the toolbar; clear a check box to remove the method from the toolbar. The methods are described in detail in **Collaboration Methods (XSLT)** on page 269. Note that **UANExtension** is an optional add-on.

Figure 239 XSLT Collaboration Method Palette Dialog Box



Collaboration Method Boxes (XSLT)

The method boxes are placed in the mapping area by dragging the corresponding icon from the method palette toolbar. As shown in **Figure 238 on page 266**, the method boxes typically have input and output nodes that you link to fields in the left and right panels, respectively. The method boxes are expanded by default (see Figure 240); you can collapse them (see Figure 241) 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 240 Expanded Method Box

Figure 241 Collapsed Method Box

8.4 Collaboration Methods (XSLT)

8.4.1 Operator Methods

Figure 242 Method Palette: Operator Methods

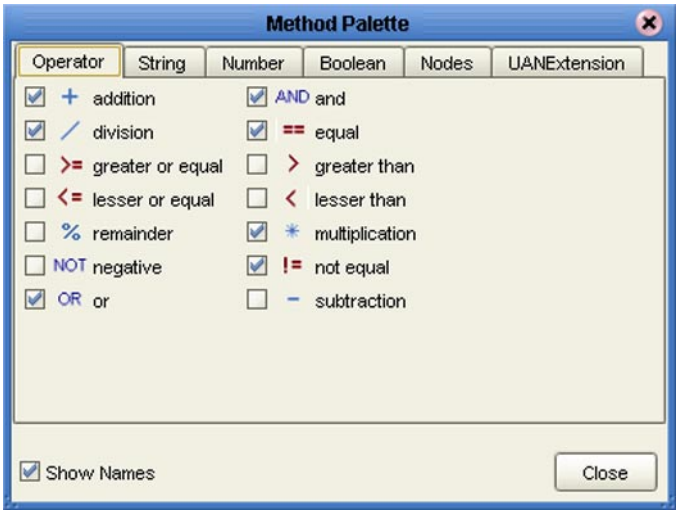


Table 66 Operator Collaboration Methods (XSLT)

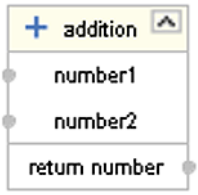
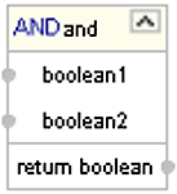
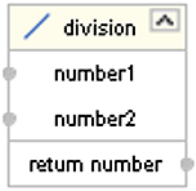
Method Box	Description/Usage
	The **addition** method adds the value of *number1* to the value of *number2*, returns the sum.
	The **and** method returns Boolean true if both *boolean1* and *boolean2* are true; otherwise, returns Boolean false.
	The **division** method divides the value of *number1*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

Table 66 Operator Collaboration Methods (XSLT)

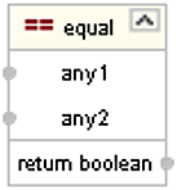
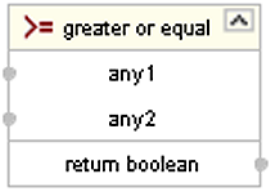
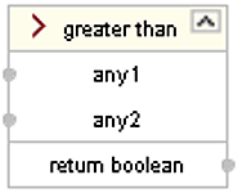
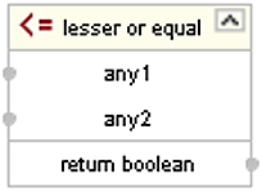
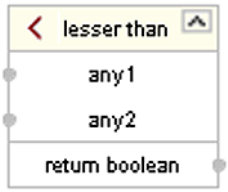
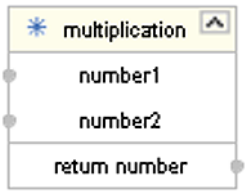
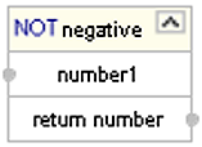
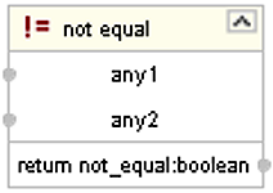
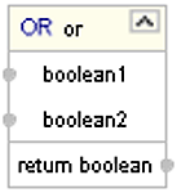
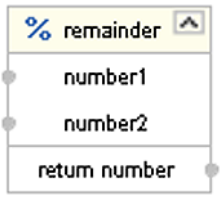
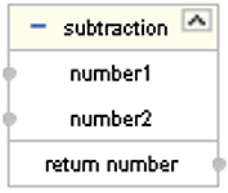
Method Box	Description/Usage
	The **equal** method returns Boolean true if *any1* is equal to *any2*; otherwise, returns Boolean false.
	The **greater_or_equal** method returns Boolean true if *any1* is greater than or equal to *any2*; otherwise, returns Boolean false.
	The **greater_than** method returns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

Table 66 Operator Collaboration Methods (XSLT)

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

8.4.2 String Methods

Figure 243 Method Palette: String Methods

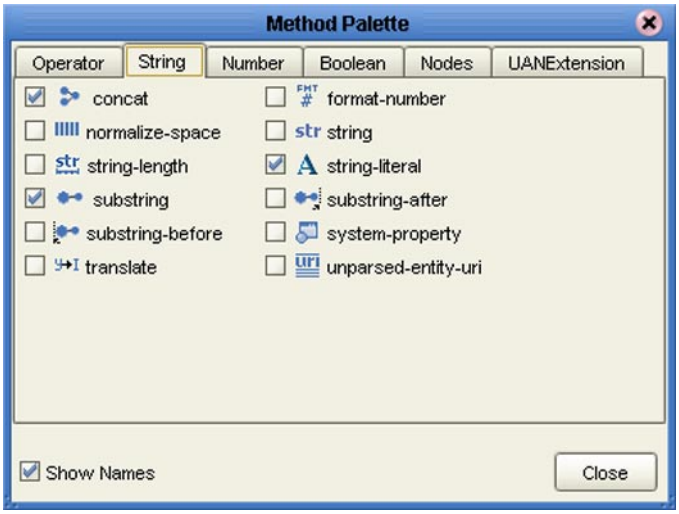


Table 67 String Collaboration Methods (XSLT)

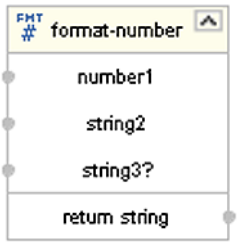
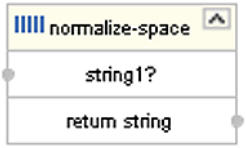
Method Box	Description/Usage
	The **concat** method returns the string created by concatenating *string2* to the end of *string1*, and *string3** (if present) to the end of *string2*.
	The **format-number** method converts its first argument (*number1*) to a string using the format pattern string specified 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.
	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 67 String Collaboration Methods (XSLT)

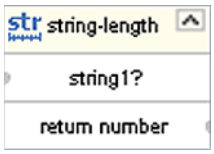
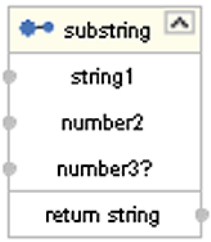
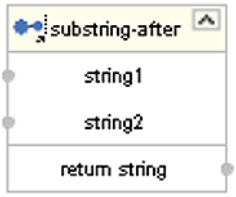
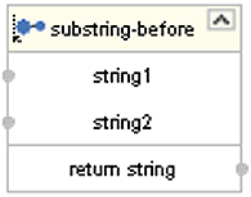
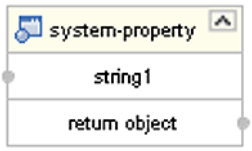
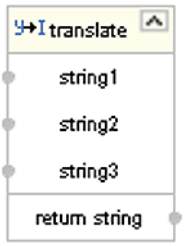
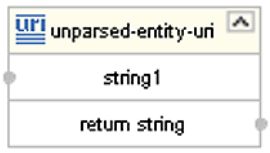
Method Box	Description/Usage
	The **string** method returns a string representation of the input object.
	The **string-length** method returns the number of characters in the string.
	Dragging the icon first displays the *String Literal* dialog box, where you enter the literal value; for example, "Data received":  The **string-literal** method then returns a string having the specified value.
	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.
	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.
	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 67 String Collaboration Methods (XSLT)

Method Box	Description/Usage
	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.
	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.
	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.

8.4.3 Number Methods

Figure 244 Method Palette: Number Methods

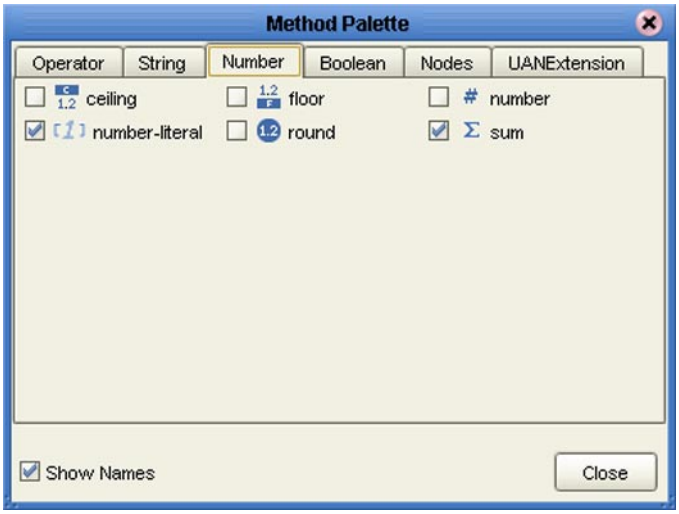


Table 68 Number Collaboration Methods (XSLT)

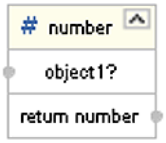
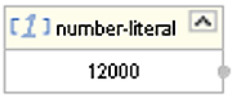
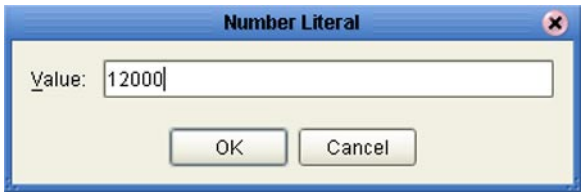
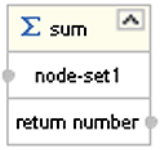
Method Box	Description/Usage
	The **ceiling** method returns the smallest (closest to negative infinity) number that is not less than the argument and that is an integer.
	The **floor** method returns the largest (closest to positive infinity) number that is not greater than the argument and that is an integer.
	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.

Table 68 Number Collaboration Methods (XSLT)

Method Box	Description/Usage
	Dragging the icon first displays the *Number Literal* dialog box, where you enter the literal value, such as "12000":  The **number-literal** method then returns a number having the specified value.
	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.
	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.

8.4.4 Boolean Methods

Figure 245 Method Palette: Boolean Methods

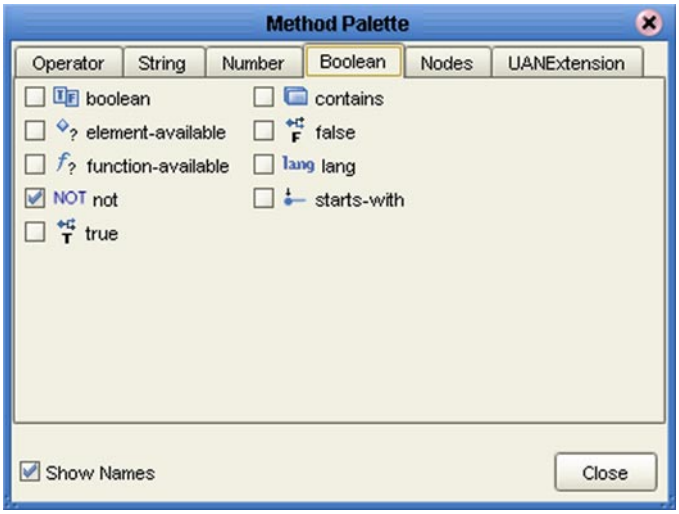


Table 69 Boolean Collaboration Methods (XSLT)

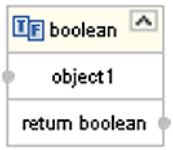
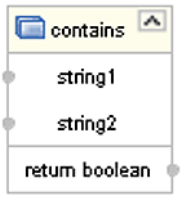
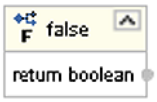
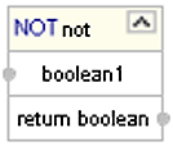
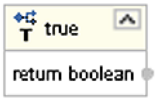
Method Box	Description/Usage
	The **boolean** method converts the argument *object1* to a Boolean true or false as follows: ▪ A number is true if and only if it is neither ±zero nor NaN (not a number). ▪ A node-set is true if and only if it is non-empty. ▪ A string is true if and only if it is non-zero.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

Table 69 Boolean Collaboration Methods (XSLT)

Method Box	Description/Usage
	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. 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.
	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.
	The **not** method returns the inverse of *boolean1*.
	The **starts-with** method returns Boolean **true** if the first argument (*string1*) starts with the second argument (*string2*); if not, returns Boolean **false**.
	The **true** method returns Boolean **true**.

8.4.5 Nodes Methods

Figure 246 Method Palette: Nodes Methods

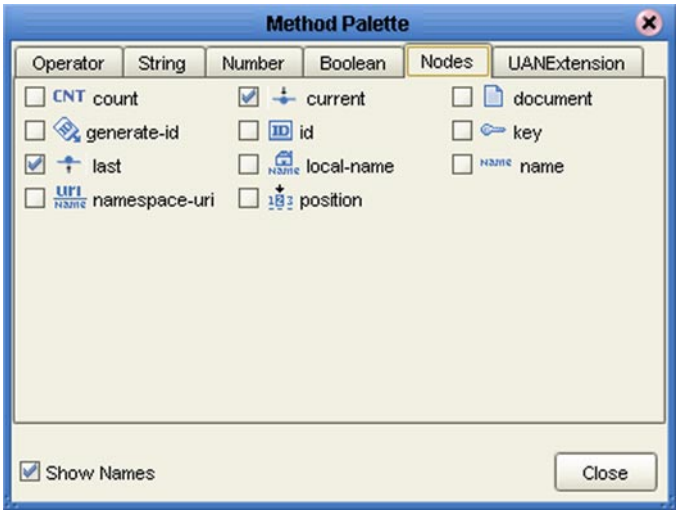


Table 70 Nodes Collaboration Methods (XSLT)

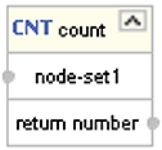
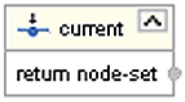
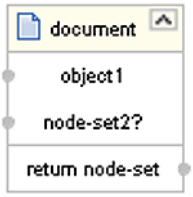
Method Box	Description/Usage
	The **count** method returns the number of nodes in the argument *node-set1*.
	The **current** method returns a node-set that has the current node as its only member.
	The **document** method allows access to XML documents other than the main source document. ▪ When the 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.

Table 70 Nodes Collaboration Methods (XSLT)

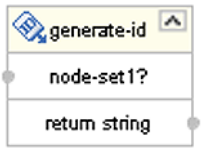
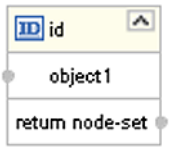
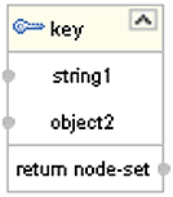
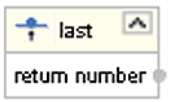
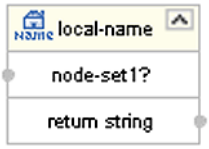
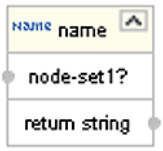
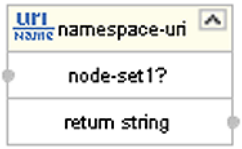
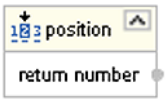
Method Box	Description/Usage
	The **generate-id** method returns a string that uniquely identifies the node in the argument *node-set1?* that is first in document order. The unique identifier must consist of ASCII alphanumeric characters and must start with an alphabetic character. Thus, the string is 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.
	The **last** method returns a number equal to the context size from the expression evaluation context.
	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.
	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.
	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.

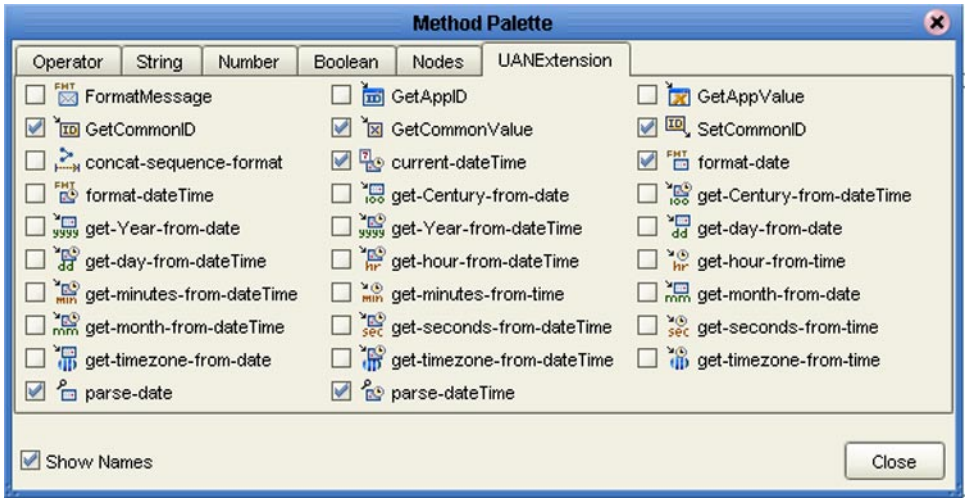
Table 70 Nodes Collaboration Methods (XSLT)

Method Box	Description/Usage
	The **position** method returns a number equal to the context position from the expression evaluation context.

8.4.6 UAN Extension Methods

These methods are used in conjunction with the Siebel UAN eWay for implementing UAN Extension functions during run time. Employing these methods causes the eWay to interact with the cross-reference (XRef) database to perform the specified request. Since these are run-time functions, their operation cannot be tested in the XSLT Tester. Definitions of the parameters used in these methods are listed in Table 72.

Figure 247 Method Palette: UAN Extension Methods



Note: *This Method Palette and the associated methods appear only if you have installed the SeeBeyond UANExtension add-on.*

Table 71 UAN Extension Collaboration Methods (XSLT)

Method Box	Description/Usage
	The **formatMessage** method function serves as a message packaging service in UAN. It is able to return a formatted message strings based on specifications given by the caller of the function. This function is able to retrieve pre-defined message text in repository, perform argument substitution and argument cross-referencing, and return the packaged message in the specified language. Only the **Code** and **Lang** parameters are required; all others are optional. ▪ **Code** is the code of the desired message; it is used to locate the desired message definition in the repository ▪ **Lang** is the desired language text of the message. ▪ **AppInst** is the name of the application instance that is to receive the message. Passing a null value indicates id cross-referencing will not be performed. ▪ **AppType** is the type of the application that is to receive this message. Passing a null value indicates value cross-referencing will not be performed. ▪ **ArgN** is the value of the Nth argument.

Table 71 UAN Extension Collaboration Methods (XSLT)

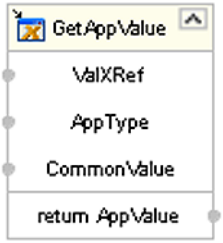
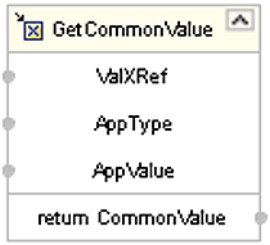
Method Box	Description/Usage
	The **getAppID** method is used to obtain a specific application ID by giving the ID cross reference name (**IDXRef**), application instance name (**AppInst**) and common ID (**commonID**). The return value of this method is the ID for the target application instance. If the application ID has not been registered, "" is returned.
	The **getAppValue** method returns the application value (**AppValue**) corresponding to the specified common value (**CommonValue**), given the application type (**AppType**) and type of object being cross-referenced (**ValXRef**).
	The **getCommonID** method is used to obtain the common ID corresponding to a specific application ID by giving the ID cross reference name (**IDXRef**), application instance name (**AppInst**) and application ID (**AppID**). The return value of this method is the common ID for the target application instance. If the application ID has not been registered, "" is returned.
	The **getCommonValue** method returns the common value (**CommonValue**) corresponding to the specified application value (**AppValue**), given the application type (**AppType**) and type of object being cross-referenced (**ValXRef**).

Table 71 UAN Extension Collaboration Methods (XSLT)

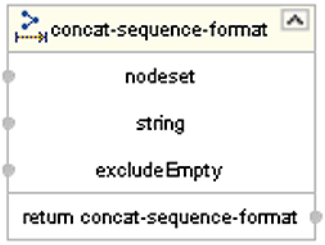
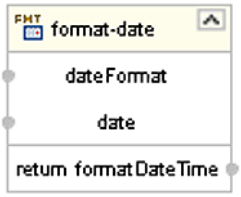
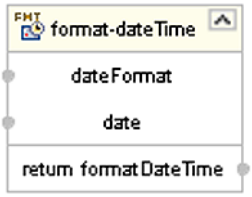
Method Box	Description/Usage
	The **setCommonID** method is used to create, query, and establish the relationship between application ID (**AppID**) and common ID (**CommonID**). ▪ **To Create:** If an application ID is being registered for the first time, the system will generate a new common ID to store the information. ▪ **To Query:** If an application ID is already registered, the common ID will be returned. ▪ **To Establish:** If the **CommonID** parameter is specified, the relationship between this common ID and the application ID will be established, linking the current application with an application ID. ▪ If the **CommonID** parameter is not specified, this method should generate an unique ID or return the existing common ID.
	The **concat-sequence-format** method concatenates the string value of each node in order, using a string separator between entries. If the Boolean parameter **excludeEmpty** is *true*, empty strings are not included. If **excludeEmpty** is omitted, *false* is assumed.
	The **currentDateTime** method returns the current time as a **dateTime** string.
	The **formatDate** method takes the given **date** string and converts it to an ISO 8601 **dateTime** string as specified by the **dateFormat** string.
	The **formatDateTime** method takes an ISO 8601 **dateTime** string and converts it to a **dateTime** string as specified by the **dateFormat** string.
	The **getCenturyFromDate** method extracts the century as a number from a string having an XML Schema **date** datatype, the encoding of which is based on the ISO 8601 reduced format (CCYY-MM-DD) with optional timezone.

Table 71 UAN Extension Collaboration Methods (XSLT)

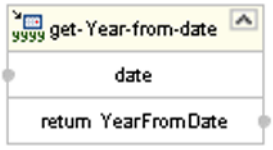
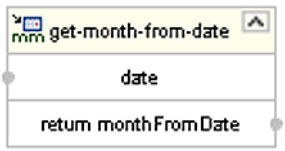
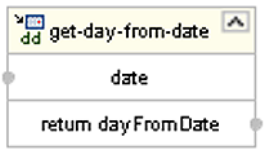
Method Box	Description/Usage
	The **getCenturyFromDateTime** method extracts the century as a number from a string having an XML Schema **dateTime** datatype, the encoding of which is based on the ISO 8601 extended format (CCYY-MM-DDThh:mm:ss) with optional timezone.
	The **getYearFromDate** method extracts the year as a number from a string having an XML Schema **date** datatype, the encoding of which is based on the ISO 8601 reduced format (CCYY-MM-DD) with optional timezone.
	The **getYearFromDateTime** method extracts the year as a number from a string having an XML Schema **dateTime** datatype, the encoding of which is based on the ISO 8601 extended format (CCYY-MM-DDThh:mm:ss) with optional timezone.
	The **getMonthFromDate** method extracts the month as a number from a string having an XML Schema **date** datatype, the encoding of which is based on the ISO 8601 reduced format (CCYY-MM-DD) with optional timezone.
	The **getMonthFromDateTime** method extracts the month as a number from a string having an XML Schema **dateTime** datatype, the encoding of which is based on the ISO 8601 extended format (CCYY-MM-DDThh:mm:ss) with optional timezone.
	The **getDayFromDate** method extracts the day as a number from a string having an XML Schema **date** datatype, the encoding of which is based on the ISO 8601 reduced format (CCYY-MM-DD) with optional timezone.
	The **getDayFromDateTime** method extracts the day as a number from a string having an XML Schema **dateTime** datatype, the encoding of which is based on the ISO 8601 extended format (CCYY-MM-DDThh:mm:ss) with optional timezone.
	The **getHourFromDateTime** method extracts the hour as a number from a string having an XML Schema **dateTime** datatype, the encoding of which is based on the ISO 8601 extended format (CCYY-MM-DDThh:mm:ss) with optional timezone. The hour value ranges from 0 to 23, inclusive.

Table 71 UAN Extension Collaboration Methods (XSLT)

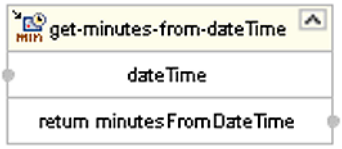
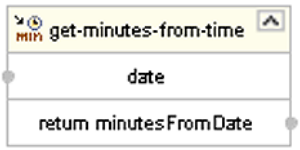
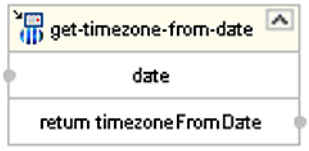
Method Box	Description/Usage
	The **getHourFromTime** method extracts the hour as a number from a string with XML Schema **time** datatype, which is the left-truncated lexical representation for **dateTime** (hh:mm:ss.ssss) with optional timezone. The hour value ranges from 0 to 23, inclusive.
	The **getMinutesFromDateTime** method extracts the minutes as a number from a string having an XML Schema **dateTime** datatype, the encoding of which is based on the ISO 8601 extended format (CCYY-MM-DDThh:mm:ss) with optional timezone. The minutes value ranges from 0 to 59, inclusive.
	The **getMinutesFromTime** method extracts the minutes as a number from a string with XML Schema **time** datatype, which is the left-truncated lexical representation for **dateTime** (hh:mm:ss.ssss) with optional timezone. The minutes value ranges from 0 to 59, inclusive.
	The **getSecondsFromDateTime** method extracts the seconds as a number from a string having an XML Schema **dateTime** datatype, the encoding of which is based on the ISO 8601 extended format (CCYY-MM-DDThh:mm:ss) with optional timezone. The seconds value ranges from 0 to 59, inclusive.
	The **getSecondsFromTime** method extracts the seconds as a number from a string with XML Schema **time** datatype, which is the left-truncated lexical representation for **dateTime** (hh:mm:ss.ssss) with optional timezone. The seconds value ranges from 0 to 59, inclusive.
	The **getTimezoneFromDate** method extracts the timezone as a number from a string having an XML Schema **date** datatype, the encoding of which is based on the ISO 8601 reduced format (CCYY-MM-DD) with optional timezone.
	The **getTimezoneFromDateTime** method extracts the timezone as a number from a string having an XML Schema **dateTime** datatype, the encoding of which is based on the ISO 8601 extended format (CCYY-MM-DDThh:mm:ss) with optional timezone.
	The **getTimezoneFromTime** method extracts the timezone as a number from a string with XML Schema **time** datatype, which is the left-truncated lexical representation for **dateTime** (hh:mm:ss.ssss) with optional timezone.

Table 71 UAN Extension Collaboration Methods (XSLT)

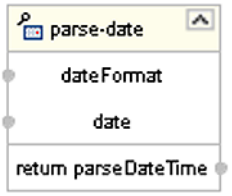
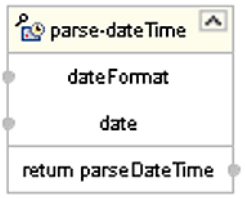
Method Box	Description/Usage
	The **parseDate** method parses the given **dateValue** string into an ISO 8601 **date** string as specified by the **dateFormat** string.
	The **parseDateTime** method parses the given **date** string into an ISO 8601 **dateTime** string as specified by the **dateFormat** string.

Table 72 UAN Extension Collaboration Method Parameter Definitions

Parameter	Description/Usage
AppID	The key, having more than 50 characters, that is used to identify the object in the particular application.
AppInst	A string of no more than 50 characters that uniquely identifies the application instance.
AppType	A string of no more than 50 characters that uniquely identifies the application type.
AppValue	The value, having no more than 50 characters, that is used by the application type.
CommonID	The key, having no more than 50 characters, that is to be used in the neutral, common format.
CommonValue	The value, having no more than 50 characters, that is to be used in the neutral, common format.
IDXRef	A string of no more than 50 characters that identifies the type of object within the system that is being cross-referenced.
ValueXRef	A string of no more than 50 characters that identifies the type of object within the application that is being cross-referenced.

8.5 Version Control

8.5.1 Creating a Modified Collaboration Definition (XSLT)

XSLT-based Collaboration Definitions can be saved to a Diff (.sdf) file. This feature allows two sites working with the same XSLT-based Collaboration Definition to seamlessly merge changes.

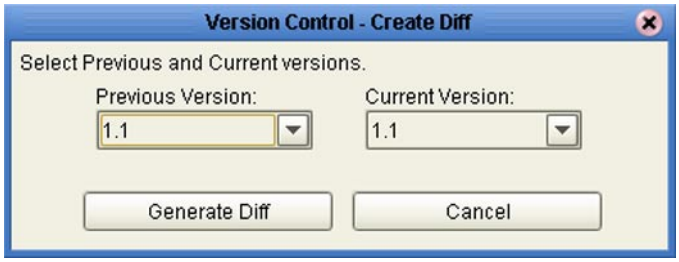
To create a *different* version of a XSLT-based Collaboration Definition file

1. From Project Explorer, select a **Collaboration Definition (XSLT)** icon.
2. Right-click to display the context menu shown in Figure 248.

Figure 248 Collaboration Definition (XSLT) Context Menu

3. Select **Create Diff** to display the dialog box shown in Figure 249.

Figure 249 Version Control - Create Diff Dialog Box



4. Click **Generate Diff** to display the *Specify Name ...* dialog box.
5. Enter a name for the Diff file in the **File Name** box (or use the default file name).
6. Click **Save** to save the Diff file.

8.5.2 Merging Two Versions of a Collaboration Definition (XSLT)

To merge two versions of an XSLT-based Collaboration Definition

1. From the Project Explorer tab, select a **Collaboration Definition (XSLT)** icon.
2. Right-click to display the Collaboration Definition (XSLT) context menu.

3 Select **Check Out**.

4 Right-click to again display the context menu.

5 Select **Merge Diff...** to display the dialog box shown in Figure 250.

Figure 250 Version Control - Merge Changes Dialog Box

6 Click the **Ellipsis** (**...**) button to display the *Specify Name ...* dialog box.

7 Locate and select the Diff file.

8 Click **Open** add the Diff file to the dialog box shown in Figure 250.

9 Click **Merge** to merge the Diff file with the corresponding XSLT-based Collaboration Definition in your Project.

10 Resolve any conflicts from the merge.

11 **Commit** the merged Collaboration.

12 **Save** the merged Collaboration.

Chapter 9

Environments

This chapter describes the process of defining eGate Environments, and the various components of an Environment.

9.1 Overview

Projects are run within *Logical Hosts*, which contain the logical resources required by the Project at run time. The Logical Hosts, in turn, are defined within *Environments*, which represent the physical resources required to implement the Project. The Environment also contains information about external systems with which the eGate Project interacts.

9.1.1 Environment Components

The components found in a typical run-time Environment are described in the following sections of this chapter:

- **Logical Hosts** on page 301
- **Integration Servers** on page 305
- **Message Servers** on page 321

9.2 Environment Explorer

The **Environment Explorer** displays the contents of the Repository that belong to the selected Environment (see Figure 251).

Figure 251 Enterprise Explorer: Environment Explorer View

The Environment Explorer is used in conjunction with the Environment Editor to create and configure the components of a run-time Environment. Each component in the Environment Explorer has an icon to identify the component type (see **Environment Explorer Icons** on page 292). Right-clicking on a component displays a context menu for that component (see **Context Menus** on page 293), from which you can select appropriate actions.

9.2.1 Environment Explorer Icons

The icons described in Table 73 appear in the Environment Explorer.

Table 73 Environment Icons

Icon	Function
	Represents the **Repository**, which is the central ICAN 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**, which contains the various logical components and files that are required at run time.
	Represents an **Environmental Constant**, which you can use to automate eWay and message destination configuration changes.
	Represents a **Scheduler** component of an Environment, which you can use to set data transfer to occur at set intervals.
	Represents a **SeeBeyond Integration Server** or third-party **application server**, which manages the Collaborations and other process interactions of an eGate Project. The integration server is deployed to a Logical Host.
	Represents a **SeeBeyond JMS IQ Manager** or third-party **message server**, which is used to store and forward eGate system messages. The message server is deployed to a Logical Host.

9.2.2 Context Menus

Right-clicking on a component in the Environment Explorer displays a context menu for that component. Included here are descriptions of options for the following component context menus:

- **Repository Menu** on page 293
- **Environment Menu** on page 294
- **Logical Host Menu** on page 295

Repository Menu

Figure 252 Repository Menu

Table 74 Repository Menu Options

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

Environment Menu

Figure 253 Environment Menu

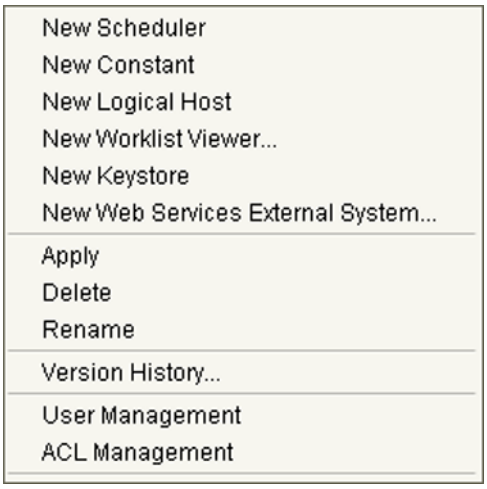


Table 75 Environment Menu Options

Option	Function
New Scheduler	Displays a dialog box with which you can add a new scheduling component to the selected Environment.
New Constant	Displays a dialog box with which you can add a constant to the selected Environment. See **Defining Environmental Constants** on page 299.
New Logical Host	Adds a new Logical Host to the selected Environment.
New Worklist Viewer	This option is present only when eInsight Business Process Manager is installed. See the *eInsight Business Process Manager User's Guide* for information.
New Keystore	Adds a new keystore to the selected Environment.
New Web Services External System	Adds a Web services external system to the selected Environment. See **Web Services External System** on page 353.
Apply	Applies the latest configuration in the Repository to the selected Environment.
Delete	If you have *delete* privileges for the Environment (see ACL Management, below), a dialog box is displayed 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.
Version History	Displays a dialog box with which you can track the version history for the selected Environment. See **Viewing a Component's Version History** on page 84 for more information.
User Management	Displays a dialog box with which an Administrator can manage message server access. See the *eGate Integrator System Administration Guide*.

Table 75 Environment Menu Options

Option	Function
ACL Management	Displays the **ACL Properties** dialog box, with which an Administrator can assign read/write/delete privileges to users for the selected Environment. See the *eGate Integrator System Administration Guide*.

Logical Host Menu

Figure 254 Logical Host Menu

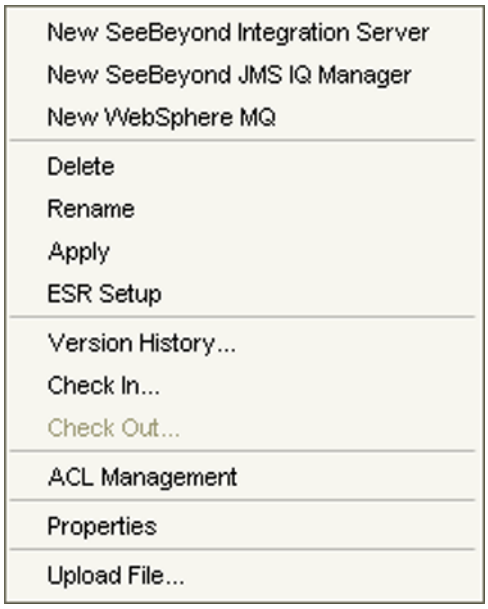


Table 76 Logical Host Menu Options

Option	Function
New SeeBeyond Integration Server	Adds a new SeeBeyond Integration Server to the selected Logical Host.
New SeeBeyond JMS IQ Manager	Adds a new SeeBeyond JMS IQ Manager to the selected Logical Host. See the *eGate Integrator JMS Reference Guide* for details.
New WebSphere MQ	Adds a new IBM WebSphere MQ message server to the selected Logical Host. See the *eGate Integrator JMS Reference Guide* for details.
Delete	Deletes the selected Logical Host, subject to the following conditions: ▪ You have *delete* privileges for the Logical Host (see ACL Management, below). ▪ The Logical Host is not checked out by anyone other than yourself. If these conditions are true, a dialog box is displayed in which you confirm that you want to delete the selected Logical Host. Clicking **Yes** then deletes the Logical Host.
Rename	Activates the field, allowing you to rename the selected Logical Host.

Table 76 Logical Host Menu Options

Option	Function
Apply	Applies the latest configuration in the Repository to the selected Logical Host.
ESR Setup	Displays a dialog box with which an Administrator can select emergency software releases (ESRs) to add to the selected Logical Host.
Version History	Displays a dialog box with which you can track the version history for the selected Logical Host. See **Viewing a Component's Version History** on page 84 for more information.
Check In	Displays a dialog box with which you can check in a new version of a Logical Host. Refer to **Checking a Component In** on page 85 for more details.
Check Out	Displays a dialog box with which you can check out the current version of a Logical Host. See **Checking a Component Out** on page 87 for more information.
ACL Management	Displays the **ACL Properties** dialog box, with which an administrator can assign read/write/delete privileges to users for the selected Logical Host. See the *eGate Integrator System Administration Guide*.
Properties	Displays a dialog box with which you can modify the configuration properties for the selected Logical Host.
Upload File	Allows you to upload third-party libraries (**.jar** files) to the Logical Host.

Note: *If you are using BEA WebLogic and/or IBM WebSphere, the Application Servers and JMS Message Servers for these products will also appear in the context menu (see Figure 255).*

Figure 255 Logical Host Menu with Third-Party Servers

Integration Server

The menu shown in Figure 256 is used for both the SeeBeyond Integration Server and third-party application servers. The configuration properties, however, are different.

Figure 256 Integration Server Menu

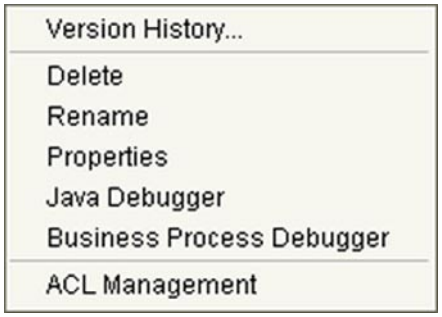


Table 77 Integration Server Menu Options

Option	Function
Version History	Displays a dialog box with which you can track the version history for the selected integration server. See **Viewing a Component's Version History** on page 84 for more information.
Delete	Deletes the selected integration server, subject to the following conditions: ▪ You have *delete* privileges for the integration server (see ACL Management, below). ▪ The integration server is not checked out by anyone other than yourself. If these conditions are true, a dialog box is displayed in which you confirm that you want to delete the selected integration server. Clicking **Yes** then deletes the integration server.
Rename	Activates the field, allowing you to rename the selected integration server.
Properties	Displays a dialog box with which you can modify the configuration properties for the selected integration server.
Java Debugger	Displays the Java Debugger (see **Invoking the Java Debugger** on page 229).
Business Process Debugger	Displays the Business Process Debugger. (This option appears in the menu only if eInsight Business Process Manager is installed in your ICAN Suite). See the *eInsight Business Process Manager User's Guide* for information.
ACL Management	Displays the **ACL Properties** dialog box, with which an administrator can assign read/write/delete privileges to users for the selected integration server. See the *eGate Integrator System Administration Guide*.

SeeBeyond JMS IQ Manager

The menu shown in Figure 257 is used for both the SeeBeyond JMS IQ Manager and third-party message servers. The configuration properties, however, are different.

Figure 257 JMS IQ Manager Menu

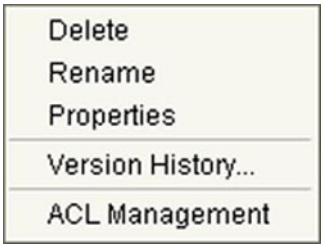


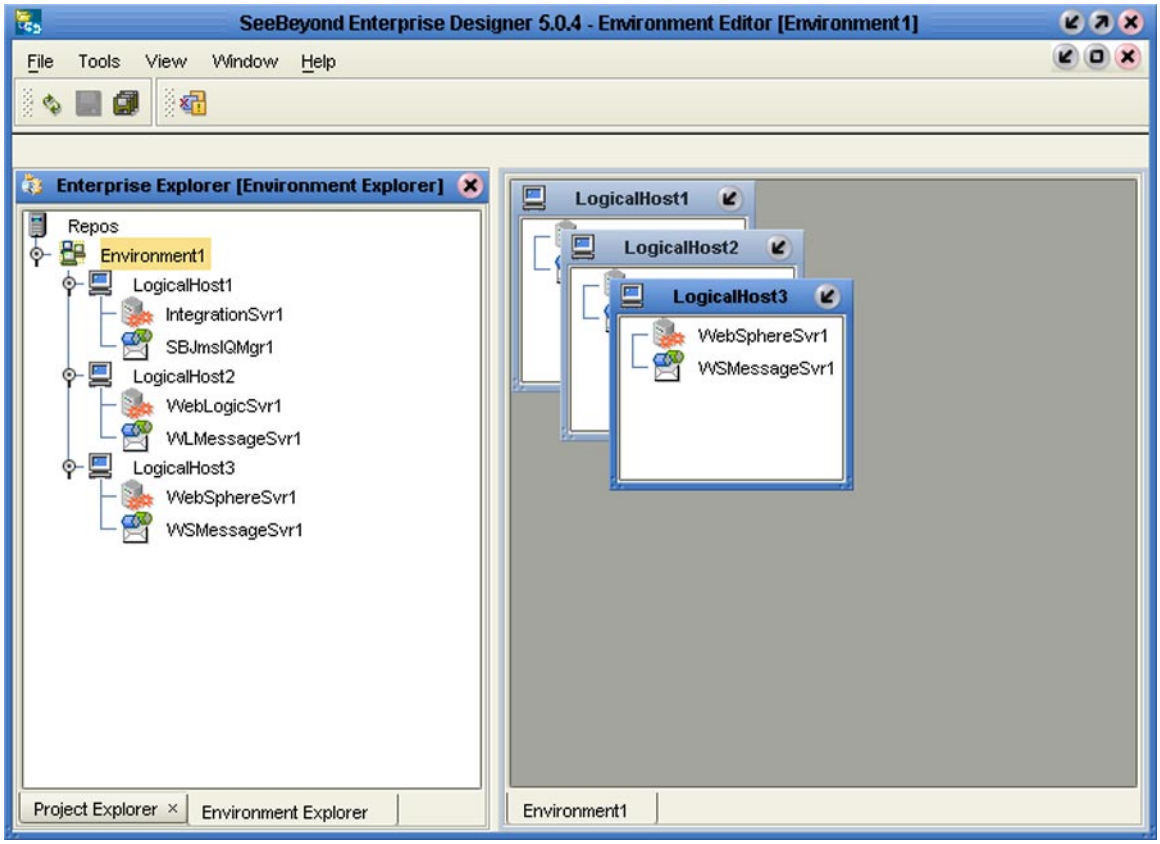
Table 78 Integration Server Menu Options

Option	Function
Delete	Deletes the selected message server, subject to the following conditions: ▪ You have *delete* privileges for the message server (see ACL Management, below). ▪ The message server is not checked out by anyone other than yourself. If these conditions are true, a dialog box is displayed in which you confirm that you want to delete the selected message server. Clicking **Yes** then deletes the message server.
Rename	Activates the field, allowing you to rename the selected message server.
Properties	Displays a dialog box with which you can modify the configuration properties for the selected message server.
Version History	Displays a dialog box with which you can track the version history for the selected message server. See **Viewing a Component's Version History** on page 84 for more information.
ACL Management	Displays the **ACL Properties** dialog box, with which an administrator can assign read/write/delete privileges to users for the selected message server. See the *eGate Integrator System Administration Guide*.

9.3 Environment Editor

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 258).

Figure 258 Environment Editor

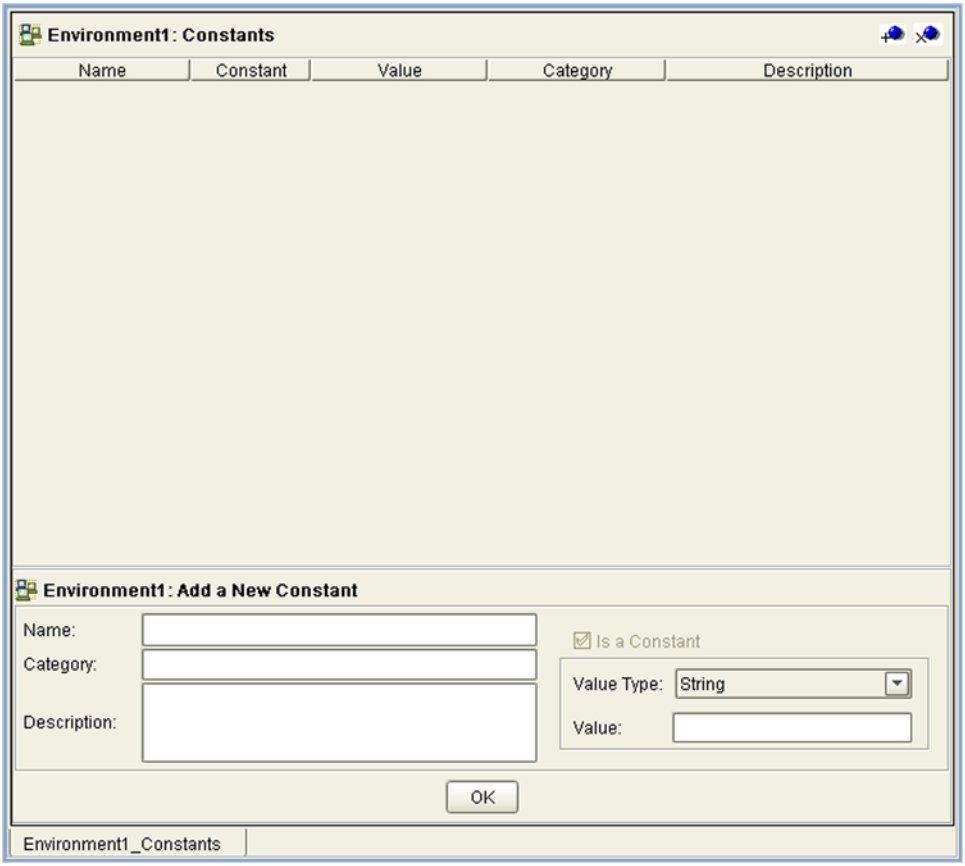


Here you can see the various components (Logical Hosts, servers, and external systems) included in the selected Environment. New Environments are added through the use of the Repository context menu (see **Repository Menu** on page 293). Components are added to the Environment by selecting options in the Environment and Logical Host context menus (see **Environment Menu** on page 294 and **Logical Host Menu** on page 295, respectively).

9.3.1 Defining Environmental Constants

Environmental constants are name/value pairs that are visible across the Environment. Selecting the **New Constant** option from the Environment context menu displays the Constants panel in the Environment Editor (see Figure 259).

Figure 259 Environmental Constants Panel



All constants defined for the specific Environment are listed in the *Constants* section of the panel, along with their various properties. New constants are added using the *Add a New Constant* section of the panel.

Note: *When you create an Environmental constant, you assign a permanent value to it—which cannot be overridden.*

Table 79 Environmental Constants Panel Icons

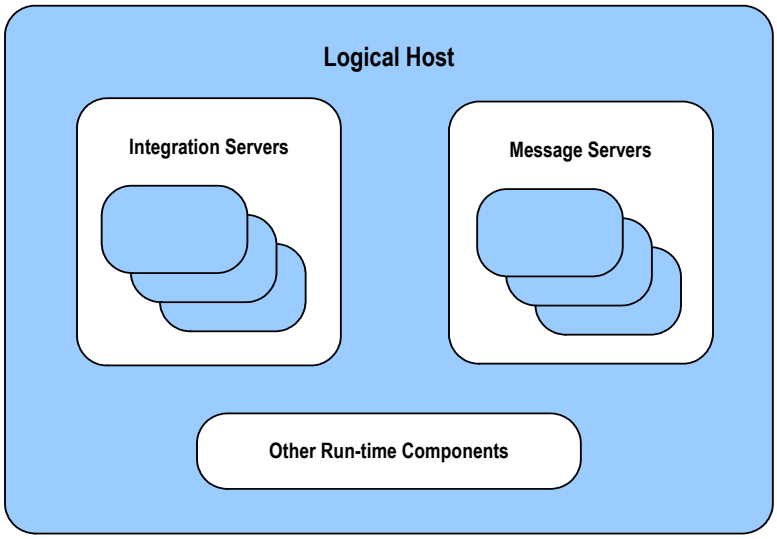
Icon	Name	Function
	Add a New Constant	Adds a new constant to the list.
	Delete a Highlighted Constant	Deletes the selected constant from the list.

9.4 Logical Hosts

9.4.1 Overview

A Logical Host contains the eGate run-time components that are installed on a host hardware platform. A Logical Host can be a member of only one Environment, but each Environment can contain multiple Logical Hosts. Both integration servers and message servers are deployed to the Logical Host, as illustrated in Figure 260.

Figure 260 Logical Hosts



The master service of the Logical host is the Management Agent. This service starts the other services on the Logical Host as part of the bootstrap process. The Management Agent also communicates with the Enterprise Manager via JMX (Java Management Extensions) to report the status of the message servers and integration servers. You can view the status of these components by means of the Enterprise Monitor.

The Enterprise Monitor also allows you to control the components within the Logical Host. Stopping the Logical Host through the Enterprise Monitor stops all components except the Management Agent, which remains running so that the components can be restarted from the Enterprise Monitor. This behavior is necessary to accommodate situations where the Logical Host and Repository are located on separate computers. See the *eGate Integrator System Administration Guide* for additional information.

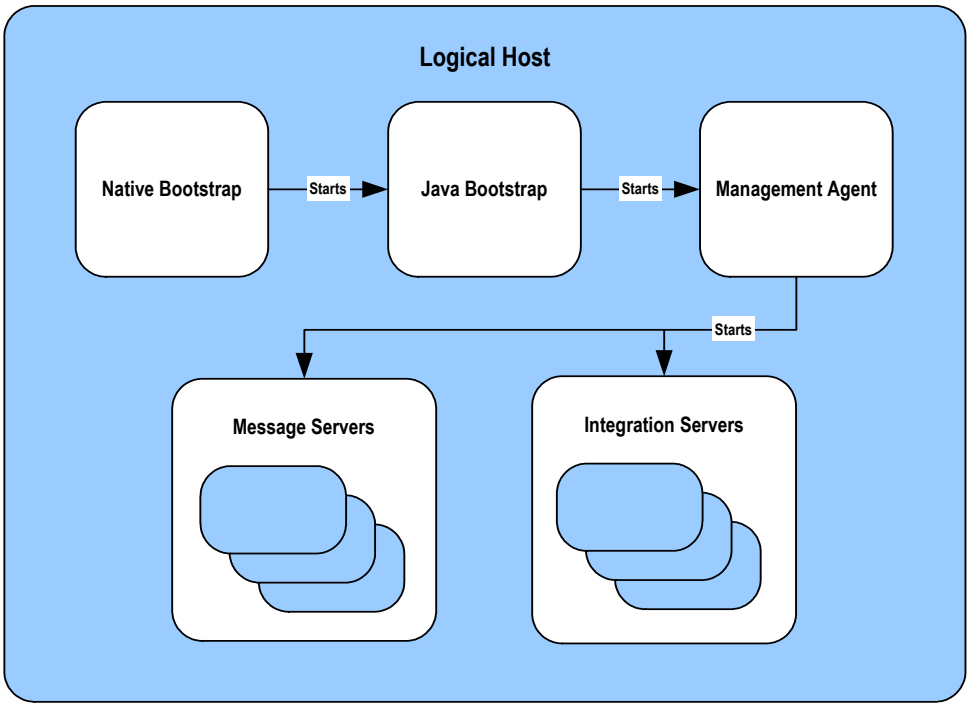
9.4.2 Bootstrapping

At run time, a platform-specific bootstrap script starts a Java bootstrap program. The first time the bootstrap runs, it downloads the Management Agent, message server, and integration server from the Repository. The Management Agent is then started, which in turn starts the message server(s) and integration server(s). Figure 261 illustrates this sequence.

Subsequent bootstraps simply start the existing Management Agent without downloading component configurations from the Repository, unless you force a download by:

- Selecting **Apply** from the Logical Host context menu (see **Logical Host Menu** on page 295).
- Using the command-line option (**-f**). See the *eGate Integrator System Administration Guide* for information.

Figure 261 Startup Sequence



Each Logical Host has a separate bootstrap process. The process is started from a script (***ICAN-root***\logicalhost\bootstrap\bin\bootstrap.bat) or (***ICAN-root***/logicalhost/bootstrap/bin/bootstrap.sh). The Repository location is furnished either by command-line parameters or the configuration properties file (***ICAN-root***\logicalhost\bootstrap\config\logical-host.properties). See the *eGate Integrator System Administration Guide* for additional information.

9.4.3 Configuring a Logical Host

To access the configuration properties for a Logical Host

1. Right-click a Logical Host in the Environment Explorer tree to display the context menu for that Logical Host instance.
2. Select **Properties** from the context menu to display the **Properties** dialog box.
3. Select the **Logical Host Configuration** node in the properties tree to display the Logical Host Configuration Section, which contains the top-level configuration properties for the Logical Host (see Figure 262).

Figure 262 Logical Host Configuration Properties

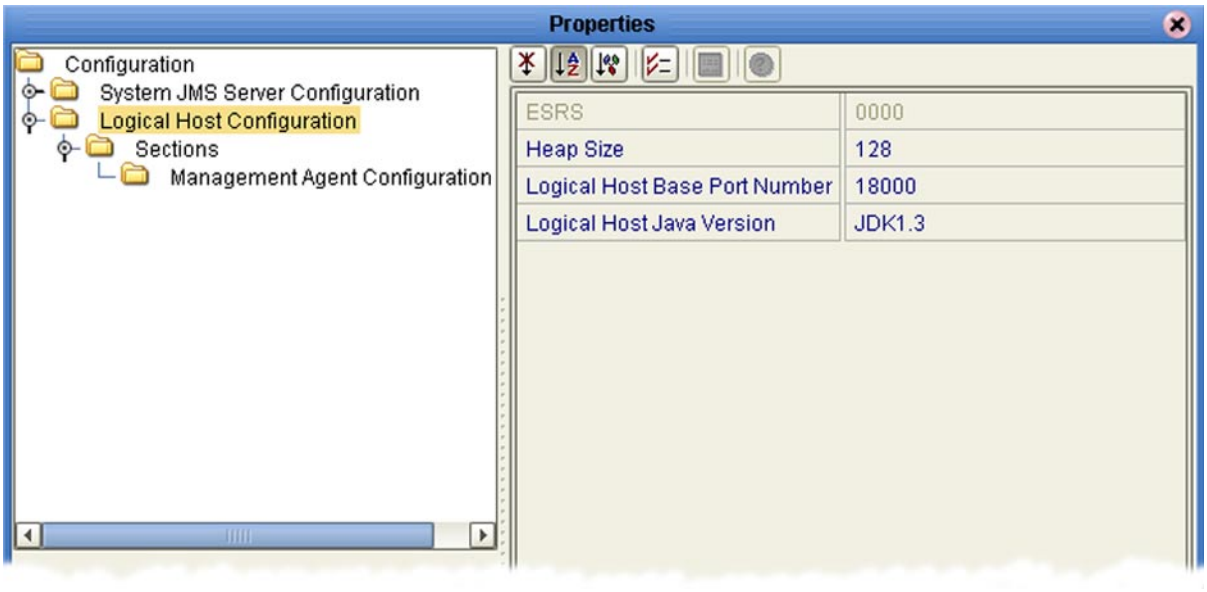


Table 80 Logical Host Configuration Properties List

Property	Description
ESRS	Shows a list of all installed Emergency Software Releases (ESRs).
Heap Size	Specifies the Heap size in Megabytes; the minimum size is **128** Mb, which is the default value. Note that this property is only for the bootstrap and management processes, and does not affect the integration server or any runtime components that are processing data.
Logical Host Base Port Number	Specifies the base port number for the Logical Host. The default value is **18000**. When multiple Logical Hosts reside on a single hardware platform, you must configure the base port numbers; see the following section.
Logical Host Java Version	Specifies the Java version being used to the eWay RAR file generation program, so that any generated file will be properly compatible. The default value is **JDK1.3**.

Configuring the Base Port Number

If multiple Logical Hosts concurrently run on the same computer, you must ensure that each Logical Host has a different base port number to avoid conflicts. This base port number is propagated throughout the Logical Host, so that the various components are automatically given successive port numbers following that assigned to the Logical Host itself.

The number of port numbers used in a Logical Host varies according to the specific implementation, so when assigning new base port numbers you need to skip successive

numbers by an adequate amount. The default base port number is 18000, so base port numbers of 19000, 20000, and so on are recommended.

If you need to assign a specific port number to a particular Logical Host component, the automatic numbering process will skip the component port number you have assigned manually (*be sure this port number is not used elsewhere*).

Note: *While Windows accepts port numbers below 12000, UNIX does not.*

Figure 263 Management Agent Configuration Properties

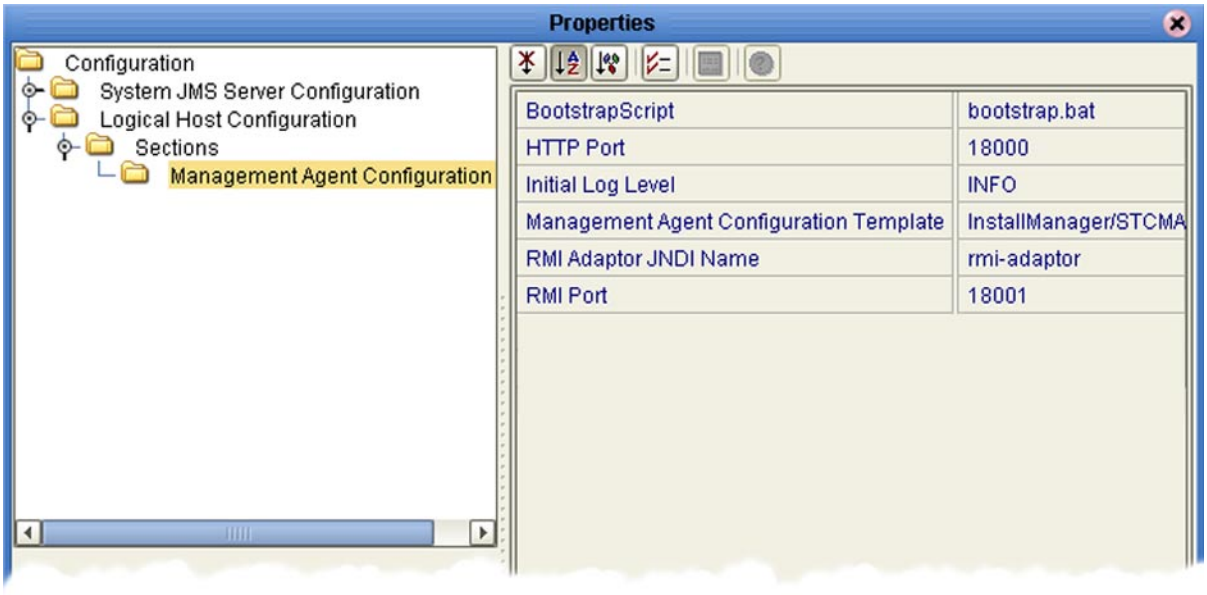


Table 81 Management Agent Configuration Properties List

Property	Description
BootstrapScript	Name of the bootstrap script; the default is **bootstrap.bat**.
HTTP Port	The HTTP port; the default value is **18008**.
Initial Log Level	The initial log level (OFF, ALL, DEBUG, INFO, WARN, ERROR, FATAL); the default value is **INFO**. See the *eGate Integrator System Administration Guide* for log level details.
Management Agent Configuration Template	The name and path of the configuration template; the default value is **InstallManager/STCMA/common/config/templates/ManagementAgent-config.vm**.
RMI Adaptor JNDI Name	The name of the RMI adapter; the default value is **rmi-adaptor**.
RMI Port	The RMI port; the default value is **18001**.

9.5 Integration Servers

The Logical Host contains one or more instances of a J2EE integration server, which is the engine that runs eGate Collaborations for processing business logic, and eWays that communicate with external applications. The integration server provides services for security, transactions, business rules execution, and connectivity management. eGate Integrator contains the SeeBeyond Integration Server, and also supports the use of third-party application servers such as BEA WebLogic and IBM WebSphere (see **Deploying Projects to Third-Party Servers** on page 341).

9.5.1 Configuring an Integration Server

To access the configuration properties for an integration server

1. Right-click an integration server in the Environment Explorer tree to display the context menu for that instance.
2. Select **Properties** from the context menu to display the **Properties** dialog box.
3. Select the **IS Configuration** node in the properties tree to display the top-level IS configuration properties (see Figure 264).

Figure 264 Top-level IS Configuration Properties

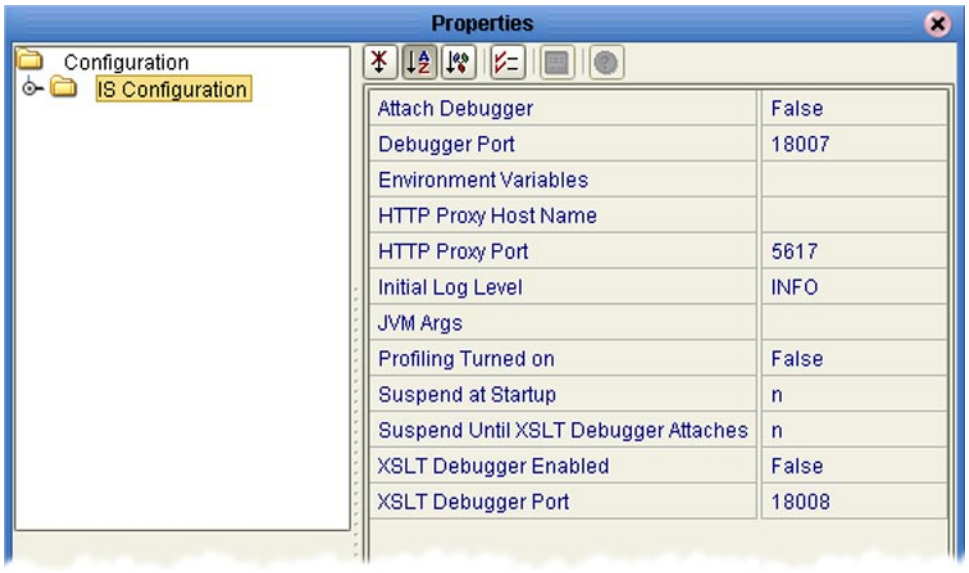


Table 82 Top-level IS Configuration Properties List

Property	Description
Attach Debugger	Enables/disables debugging for the IS. The default is **False** (disabled).
Debugger Port	This property is used only when the Debugger is enabled. The default value depends upon the number of the Logical Host base port.

Table 82 Top-level IS Configuration Properties List

Property	Description
Environment Variables	Specifies user-defined Environment Variables. Each element has the format name=value. When present, these values override the system settings, so that *all* required variables must be set. There is no default.
HTTP Proxy Host Name	
HTTP Proxy Port	
Initial Log Level	The initial log level (OFF, ALL, DEBUG, INFO, WARN, ERROR, FATAL); the default value is **INFO**. See the *eGate Integrator System Administration Guide* for log level details.
JVM Args	Java Virtual Machine (JVM) arguments. Each element in the collection should specify one, and only one, argument.
Profiling Turned On	Enables/disables performance monitoring for the IS. The default is **False** (disabled). To enable, change to **True** (as shown) and configure the properties described in **Performance Monitoring (Profiling)** on page 310.
Suspend at Startup	Allows the VM to begin executing before the debugger application attaches. The default is **n** (do not suspend).
Suspend Until XSLT Debugger Attaches	Allows the VM to begin executing before the XSLT debugger application attaches. The default is **n** (do not suspend).
XSLT Debugger Enabled	Enables/disables XSLT debugging for the IS. The default is **False** (disabled).
XSLT Debugger Port	This property is used only when the XSLT Debugger is enabled. The default depends upon the value of the Logical Host base port.

The IS Configuration node contains several sections, each containing detailed configuration properties for a particular IS component (including the integration server itself). You can also access these properties by selecting **Properties** from the context menus for the appropriate nodes.

Web Container

Properties included here are used for setting up Web services.

Figure 265 Web Container Configuration Properties

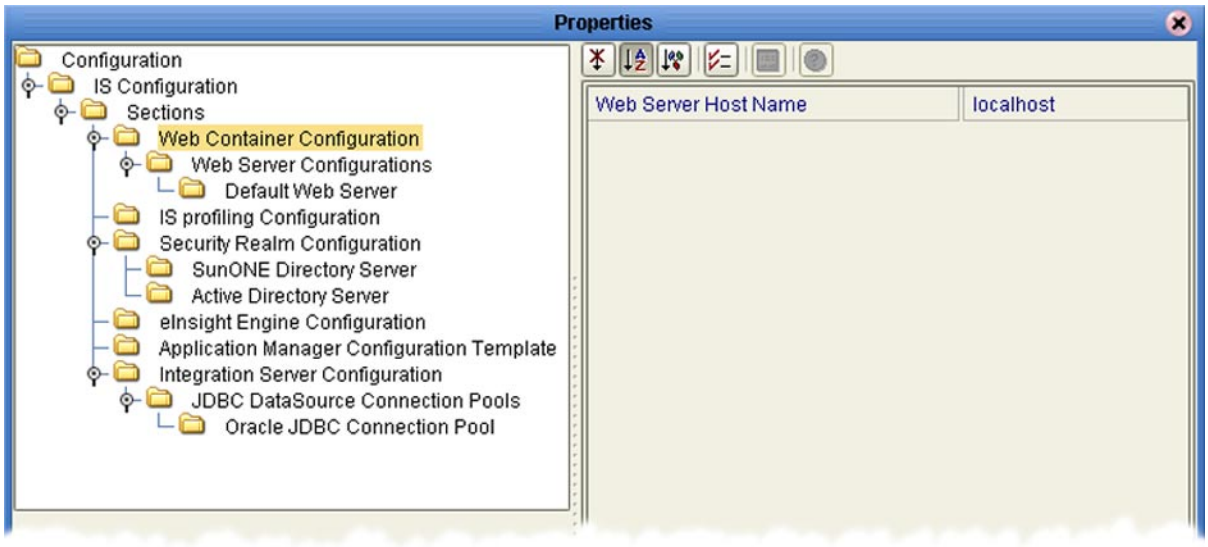


Table 83 Web Container Configuration Properties List

Property	Description
Web Server Host Name	Specifies the host name; the default is **localhost**.

Web Server Configurations

This directory serves as a container for the configuration properties for all Web servers associated with the selected integration server. You should create new sets of configuration properties for each server, and preserve the default set as a template.

To create a new set of Web server configuration properties

1. Right-click Web Server Configurations to display the context menu, and click **Create New Section**.
2. Give the newly created section an appropriate name.
3. In the properties for the newly created section, make any necessary changes to the default properties.
4. Click **OK**.

Default Web Server

The Default Web Server properties serve as a template for the individual Web servers.

Figure 266 Default Web Server Properties

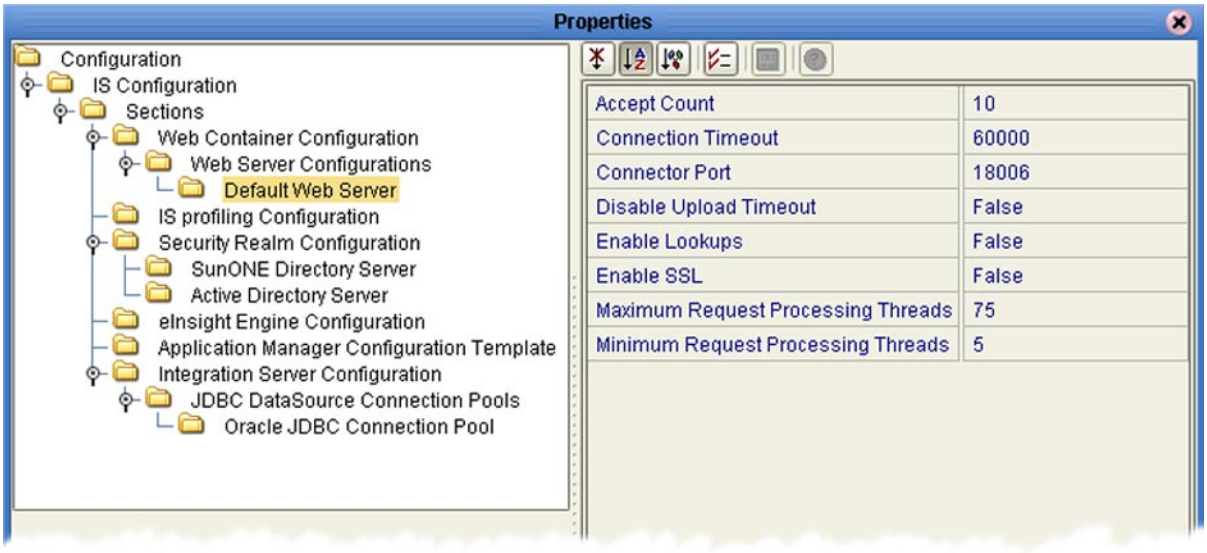


Table 84 Default Web Server Properties List

Property	Description
Accept Count	Specifies the maximum acceptable number of incoming connection requests when all possible request processing threads are in use. Any requests received beyond this number when the queue is full are refused. The default value is **10**.
Connection Timeout	Specifies the time period in milliseconds that this connector will wait for the request URI line to be presented, after accepting the connection. The default value is **60,000** ms (1 min).
Connector Port	Specifies the connection port for the Web server. The default value is **18006**.
Disable Upload Timeout	Allows the servlet container to use a different, and longer, connection timeout while a servlet is executing. This gives the servlet a longer time to complete execution, and/or provides a longer timeout during data upload. The default value is **False**.
Enable Lookups	If set to **True**, calls are made requesting getRemoteHost() to perform DNS lookups in order to return the actual host name of the remote client. If set to **False**, the DNS lookup is bypassed and the IP address is returned in string form, thereby improving performance. The default value is **False**.
Enable SSL	Specifies whether or not to enable the Secure Sockets Layer (SSL) protocol. The default value is **False**.

Table 84 Default Web Server Properties List

Property	Description
SSL Client Authentication Required	Displayed only if **Enable SSL** is set to **True**. Set this property to **True** if you want to require SSL Client Authentication. The default value is **False**.
SSL Keystore Password	Displayed only if **Enable SSL** is set to **True**. Enter your desired Keystore password (there is no default value).
Maximum Request Processing Threads	Specifies the maximum number of request processing threads to be created by this connector, thereby determining the maximum number of simultaneous requests that can be handled. The default value is **75**.
Minimum Request Processing Threads	Specifies the number of request processing threads to be created by this connector when it is first started. This value must be less than the value set for the Maximum Request Processing Threads property. The default value is **5**.

Performance Monitoring (Profiling)

You can monitor the performance of the integration server by using the built-in *Heap Analysis* tool, which is enabled and configured using the Profiling Configuration dialog box (see Figure 267).

Figure 267 Profiling Configuration Properties

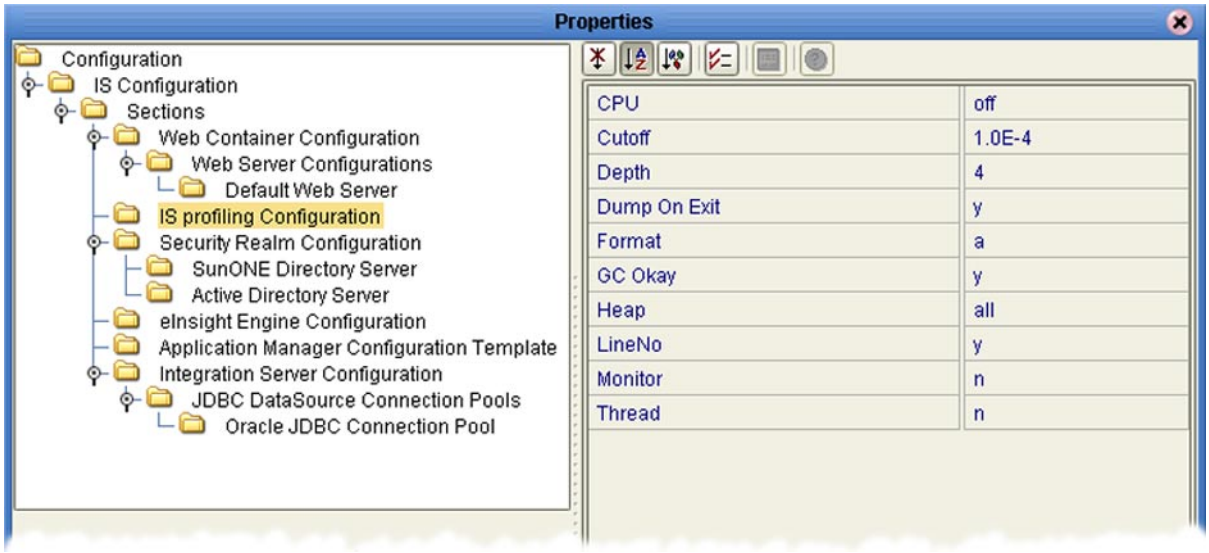


Table 85 Profiling Configuration Properties List

Property	Description
CPU	Specifies whether or not CPU usage is included in the trace. The default value is **off**.
Cutoff	Specifies the output cutoff point. The default value is **1.0E-4**.
Depth	Specifies the stack trace depth. The default value is **4**.
Dump on Exit	Specifies whether or not to dump on exit. The default value is **y** (yes).
Format	Specifies ASCII (**a**) or binary (**b**) output. The default value is **a** (ASCII).
GC Okay	Specified whether or not to allow garbage collection (GC) during sampling. The default value is **y** (yes).
Heap	Specifies the blocks of memory to include in traces. The default value is **all**.
LineNo	Specifies whether or not to include line numbers in traces. The default value is **y** (yes).
Monitor	Specifies whether or not to include monitor contention. The default value is **n** (no).
Thread	Specifies whether or not to include the thread in traces. The default value is **n** (no).

Security Realm

These properties pertain to the Lightweight Directory Access Protocol (LDAP), if used. Subdirectories contain properties for SunONE Directory Server and Microsoft Active Directory Server. See the *eGate Integrator System Administration Guide* for information regarding Security Realm configuration.

Figure 268 Security Configuration Properties

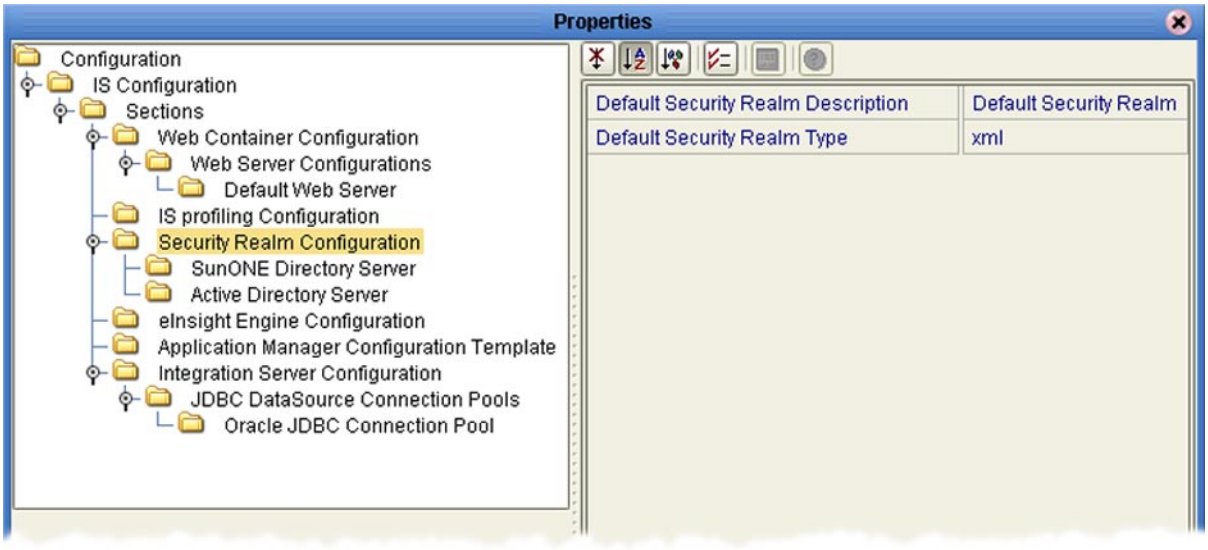


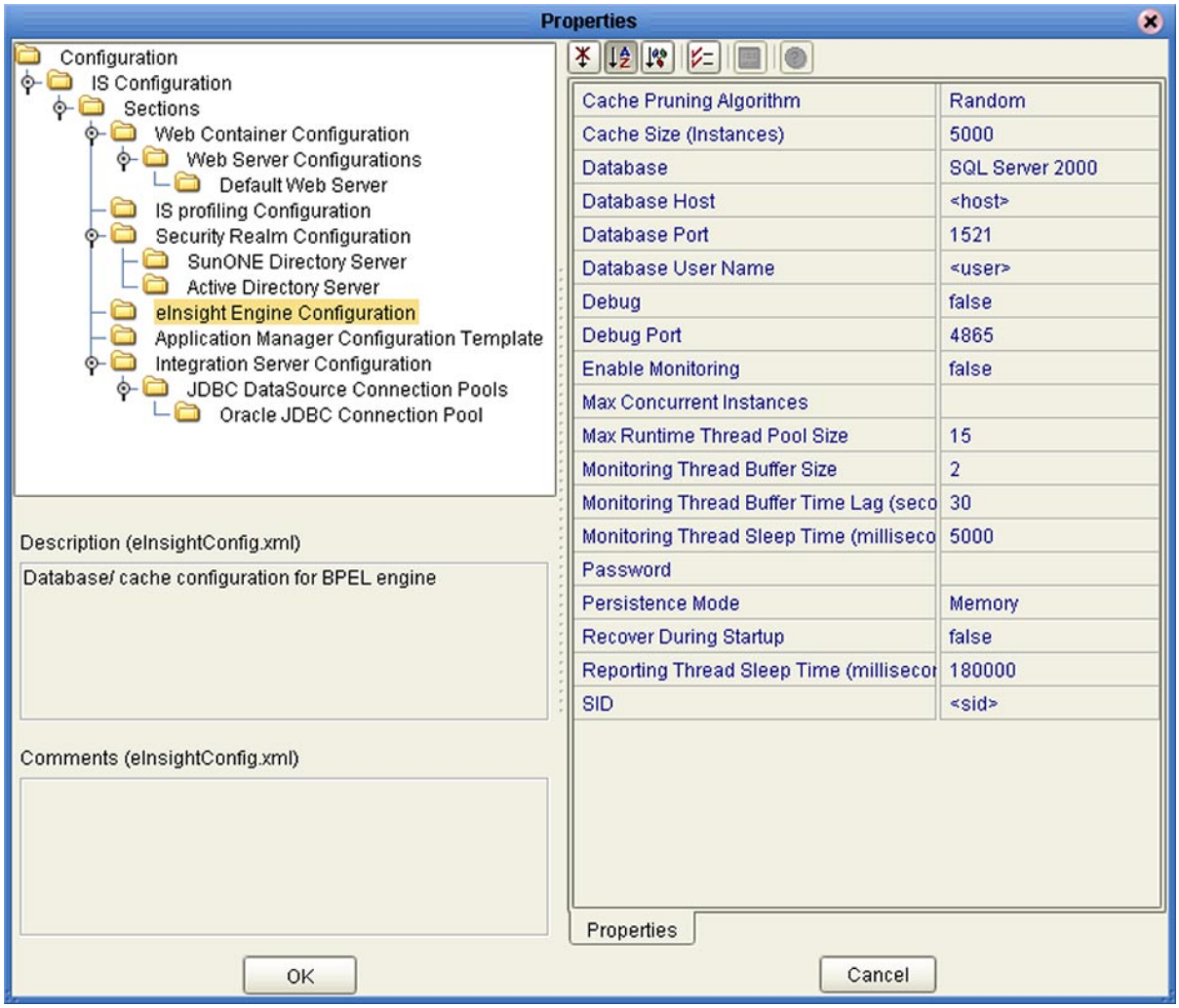
Table 86 Security Realm Configuration Properties List

Property	Description
Default Security Realm Description	Specifies the name for the default LDAP Security Realm. The default value is **Default Security Realm**.
Default Security Realm Type	Specifies the default LDAP Security Realm type. The default value is **xml**.

eInsight Engine

This configuration node is displayed 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 *eInsight Business Process Manager User's Guide* for information regarding these properties (see Figure 269).

Figure 269 eInsight Engine Configuration Properties



Application Manager

You can set integration server thread pool variables using the Application Manager Configuration Properties dialog box (see Figure 270).

Figure 270 Application Manager Configuration Properties

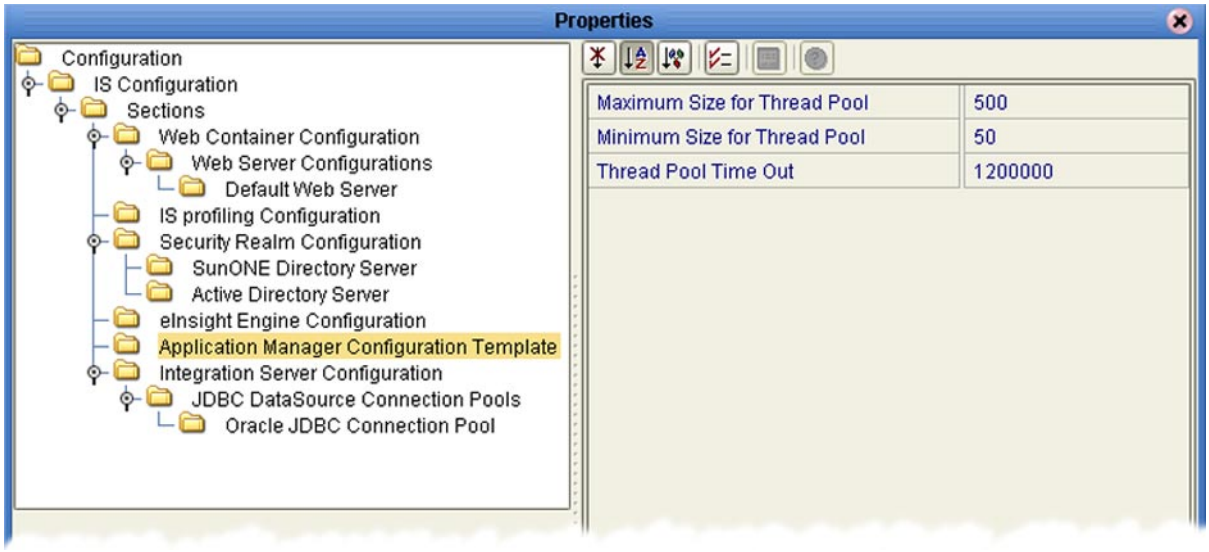


Table 87 Application Manager Configuration Properties List

Property	Description
Maximum Size for Thread Pool	Specifies the maximum number of threads that can exist in the thread pool. The default value is **500**.
Minimum Size for Thread Pool	Specifies the minimum number of threads for the thread pool. The default value is **50**.
Thread Pool Time Out	Specifies the timeout interval for the thread pool, measured in milliseconds. The default value is **1,200,000** ms (20 min).

Integration Server

Detailed, low-level configuration of the integration server is performed using the Integration Server Configuration Properties dialog box (see Figure 271).

Figure 271 Integration Server Configuration Properties

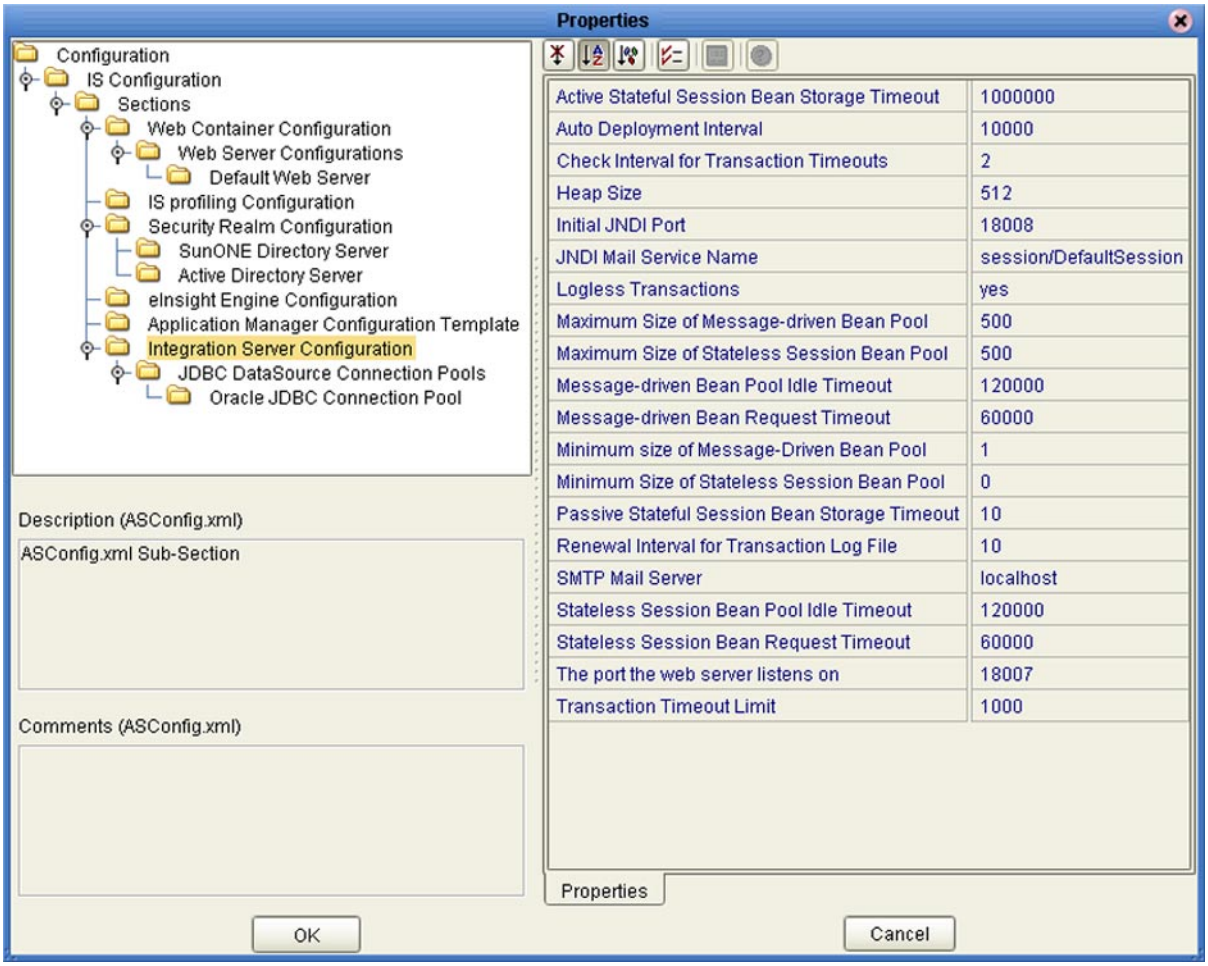


Table 88 Integration Server Configuration Properties List

Property	Description
Active Stateful Session Bean Storage Timeout	Specifies the interval after which an Active Stateful Session Bean is removed from storage, measured in minutes. The default value is **1,000,000** min, which ensures that it will not be removed unintentionally.
Auto Deployment Interval	Specifies the interval at which the auto-deployer checks the deployment directory for files, measured in milliseconds. The default value is **10,000** ms.
Check Interval for Transaction Timeouts	Specifies the interval between checks for transaction timeouts, measured in minutes. The default value is **2** min.

Table 88 Integration Server Configuration Properties List

Property	Description
Heap Size	Specifies the Heap size in Megabytes; the minimum size is **512** Mb, which is the default value. Increasing this value increases the JVM size.
Initial JNDI Port	Specifies the initial port required by the Naming Service class for startup. The default value depends upon the value of the Logical Host base port.
JNDI Mail Service Name	Specifies the name of the JNDI mail service. The default value is **session/DefaultSession**.
Logless Transactions	Specifies whether or not logless transactions are allowed. The default value is **yes**.
Maximum Size of Message-driven Bean Pool	Specifies the maximum number of Message-driven Beans allowed in the Message-driven Bean pool at one time. The default value is **500**.
Maximum Size of Stateless Session Bean Pool	Specifies the maximum number of Stateless Session Beans allowed in the Stateless Session Bean pool at one time. The default value is **500**.
Message-driven Bean Pool Idle Timeout	Specifies the timeout interval for the Message-driven Bean pool, measured in milliseconds. The default value is **120,000** ms (2 min).
Message-driven Bean Request Timeout	Specifies the interval after which a Message-driven Bean request times out, measured in milliseconds. The default value is **60,000** ms (1 min).
Minimum Size of Message-driven Bean Pool	Specifies the minimum number of Message-driven Beans allowed in the Message-driven Bean pool at one time. The default value is **1**.
Minimum Size of Stateless Session Bean Pool	Specifies the maximum number of Stateless Session Beans allowed in the Stateless Session Bean pool at one time. The default value is **1**.
Passive Stateful Session Bean Storage Timeout	Specifies the interval after which a Passive Stateful Session Bean is removed from storage, measured in minutes. The default value is **10** min.
Renewal Interval for Transaction Log File	Specifies the interval for renewing the Transaction Service log file, measured in hours. The default value is **10** hr.
SMTP Mail Server	Specifies the name of the SMTP mail host server. The default value is **localhost**.
Stateless Session Bean Pool Idle Timeout	Specifies the timeout interval for the Stateless Bean pool, measured in milliseconds. The default value is **120,000** ms (2 min).
Stateless Session Bean Pool Request Timeout	Specifies the interval after which a Stateless Bean request times out, measured in milliseconds. The default value is **60,000** ms (1 min).
The Port the Web Server Listens On	Specifies the port the Web server listens on. The default value depends upon the value of the Logical Host base port.

Table 88 Integration Server Configuration Properties List

Property	Description
Transaction Timeout Limit	Specifies the time limit for transactions to time out, measured in seconds. The default value is **1,000** sec.

JDBC DataSource Connection Pools

This directory serves as a container for the configuration properties for all JDBC DataSource Connection Pools associated with the selected integration server. The only set of default configuration properties currently furnished is for Oracle JDBC.

Oracle JDBC Connection Pool

Connection Pool properties for an Oracle database associated with the integration server are specified in the Oracle JDBC Connection Pool dialog box (see Figure 272).

Figure 272 Oracle JDBC Connection Pool Properties

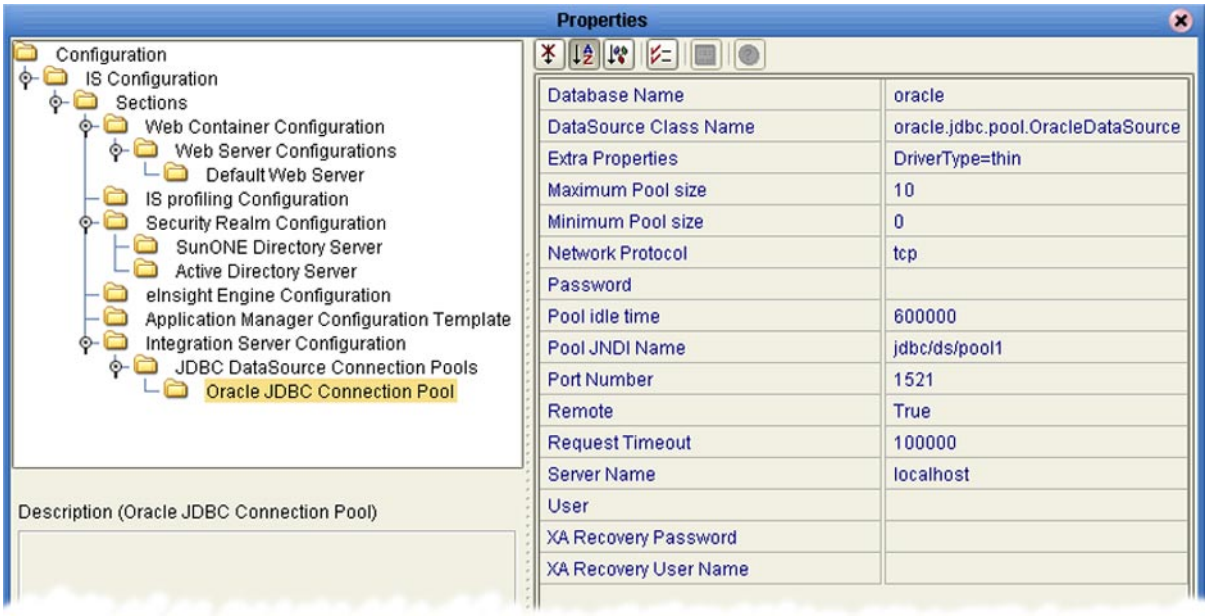


Table 89 Oracle JDBC Connection Pool Properties List

Property	Description
Database Name	Specifies the name of the database for which the pool is created. The default value is **oracle**.
DataSource Class Name	Specifies the name of the DataSource class. The default value is **oracle jdbc pool OracleDataSource**.
Extra Properties	Specifies custom properties for the DataSource, using semicolon-separated key-value pairs. The default value is **DriverType=thin**.

Table 89 Oracle JDBC Connection Pool Properties List

Property	Description
Maximum Pool Size	Specifies the maximum number of connections in the pool. The default value is **10**.
Minimum Pool Size	Specifies the minimum number of connections in the pool. The default value is **0**.
Network Protocol	Specifies the network protocol. The default value is **tcp**.
Password	Specifies the password for the connection. There is no default value.
Pool Idle Time	Specifies the maximum time period in milliseconds that a connection may remain unused before it is removed from the pool in order to reduce the pool size. The default value is **600,000** ms (10 min).
Pool JINI Name	Specifies the unique JINI name of the DataSource pool. The pool is bound in the java/namespace for local access or into the global namespace for remote access. The default value is **jdbc/ds/pool1**.
Port Number	Specifies the port number on which the server receives data. The default value is **1521**.
Remote	Specifies whether or not the DataSource should be bound into the global remote JINI namespace for access by remote clients. The default value is **true**.
Request Timeout	Specifies the maximum time period in milliseconds that a request for connection from the pool may block all other connections currently in use. The default value is **100,000** ms.
Server Name	Specifies the host name of the database server or IP address where the database server is running. The default value is **localhost**.
User	Specifies the user name authorized for creating connections. There is no default value.
XA Recovery Password	For XA DataSources only, specifies the password to use for XA transaction recovery. There is no default value.
XA Recovery User Name	For XA DataSources only, specifies the user name to use for XA transaction recovery. There is no default value.

9.5.2 Deploying User-Defined Stateless Session Beans

User-defined stateless session beans can be deployed to the eGate Integration Server following the procedure outlined in this section.

Note: *The deployment of stateful session beans, entity beans, and message-driven beans is not currently supported.*

To deploy a stand-alone SLSB to the eGate Integration Server

1. Create and compile the EJB.
2. Write the **ejb-jar.xml** and **seebeyond-ejb.xml** deployment descriptors for your EJB.
3. Create a **.jar** file with the deployment descriptors in the **\META-INF** directory and the code in the root.
4. Move the **.jar** file into the **\logicalhost\stcis\deploy\new***integration_server_name* directory for deployment. The Integration Server will automatically pick up the **.jar** file from this location and deploy the EJB.

Examples of the EJBs and associated **.xml** files are as follows.

Example Remote Interface

```
package ejb.CustomApp;
import java.rmi.RemoteException;
import java.rmi.Remote;
import javax.ejb.*;

public interface CustomApp extends EJBObject, Remote
{
  public String getId() throws RemoteException;
}
```

Example Home Interface

```
package ejb.CustomApp;

import javax.ejb.*;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.*;

public interface CustomAppHome extends EJBHome
{
   public CustomApp create() throws CreateException, RemoteException;
}
```

Example Stateless Session Bean (SLSB)

```
package ejb.CustomApp;

import javax.ejb.*;
import java.io.Serializable;
import java.util.*;
import java.rmi.*;
import javax.naming.Context;
import javax.naming.InitialContext;

// import addtional classes as needed "CustomController"
```

```
public class CustomAppBean implements SessionBean
{
  private SessionContext ctx;
  private CustomController mCustom;

  public void setSessionContext( SessionContext context )
  {
     this.ctx = context;
  }

  public void ejbCreate()
  {
     try {
       javax.naming.Context context = new InitialContext();
       // lookup Custom application
       Object ref = context.lookup("ejb/CustomController");
       CustomControllerHome CustomHome =
(CustomControllerHome)javax.rmi.PortableRemoteObject.narrow(ref,
CustomControllerHome.class);
       mCustom = CustomHome.create();
     } catch (Exception e) {
     System.out.println( e.getMessage() );
     }
  }

  public String getId()
  {
     SystemObjectPK key = new SystemObjectPK( "SBYN", "0000000001" );
     String EUID= "Not Found";
     try {
    EUID = mCustom.getEUID( key );
     }
     catch (Exception e) {
     System.out.println("===> Exception: " );
     System.out.println( e.getMessage() );
     }

     return( EUID );
  }

  // add addtional EJB methods

}
```

Example ejb-jar.xml file for the above SLSB

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ejb-jar PUBLIC '-//Sun Microsystems, Inc.//DTD Enterprise
JavaBeans 2.0//EN' 'http://java.sun.com/dtd/ejb-jar_2_0.dtd'>
<!-- Generated XML! -->
<ejb-jar>
    <display-name>ServiceBeans</display-name>
    <enterprise-beans>
        <session>
            <description><![CDATA[Custom App Session Bean]]></
description>
            <display-name>Custom App</display-name>
            <ejb-name>CustomApp</ejb-name>
            <home>ejb.CustomApp.CustomAppHome</home>
            <remote>ejb.CustomApp.CustomApp</remote>
            <ejb-class>ejb.CustomApp.CustomAppBean</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Bean</transaction-type>
            <ejb-ref>
                <ejb-ref-name>ejb/CustomController</ejb-ref-name>
```

```
                    <ejb-ref-type>Session</ejb-ref-type>

<home>com.stc.eindex.ejb.Custom.CustomControllerHome</home>
                    <remote>com.stc.eindex.ejb.Custom.CustomController</
remote>
                    <ejb-link>CustomController</ejb-link>
                </ejb-ref>
            </session>
        </enterprise-beans>
</ejb-jar>
```

Example seebeyond-ejb.xml file for the above SLSB

```
<sbyn-ejb-deployment-descriptor>
    <enterprise-beans>
        <session>
            <ejb-name>CustomApp</ejb-name>
            <jndi-name>ejb/CustomApp</jndi-name>
            <security>
                <authorize>no</authorize>
                <authenticate>no</authenticate>
                <security-audit>no</security-audit>
            </security>
            <pool-min>1</pool-min>
        </session>
    </enterprise-beans>
</sbyn-ejb-deployment-descriptor>
```

SLSB Deployment verification

Examine the log file **\logicalhost\logs\stc_is_*integration_server_name*.log**. You should find text such as "**CustomApp (EJB) was successfully deployed**" confirming deployment.

To remove a stand-alone SLSB from the eGate Integration Server

1. Shut down the Logical Host containing the Integration Server where the SLSB is deployed.
2. Remove the **.jar** file created in the preceding deployment procedure from the **\logicalhost\stcis\repository\applications*integration_server_name*\EJB** directory.
3. Restart the Logical Host.

9.6 Message Servers

The Logical Host contains one or more Message Servers, which manage JMS topics (publish-and-subscribe messaging) and queues (point-to-point messaging). eGate Integrator includes the SeeBeyond JMS IQ Manager as its Java Messaging Service (JMS) implementation. The JMS IQ Manager conforms to the Java Message specification 1.0.2b, and supports both topic (publish-and-subscribe) and queue (point-to-point) messaging styles.

eGate also includes support for HP NonStop JMS for eGate implementations on HP NonStop Server platforms. Third-party application servers such as BEA WebLogic and IBM WebSphere incorporate their own message servers.

This section presents an overview of the configuration properties for the SeeBeyond JMS IQ Manager. For more information on the JMS IQ Manager, and deploying Project components to third-party message servers, see the *eGate Integrator JMS Reference Guide*.

9.6.1 SeeBeyond JMS IQ Manager Configuration

General Configuration

These properties cover the basic configuration of the JMS IQ Manager.

Figure 273 JMS IQ Manager - General Configuration Properties

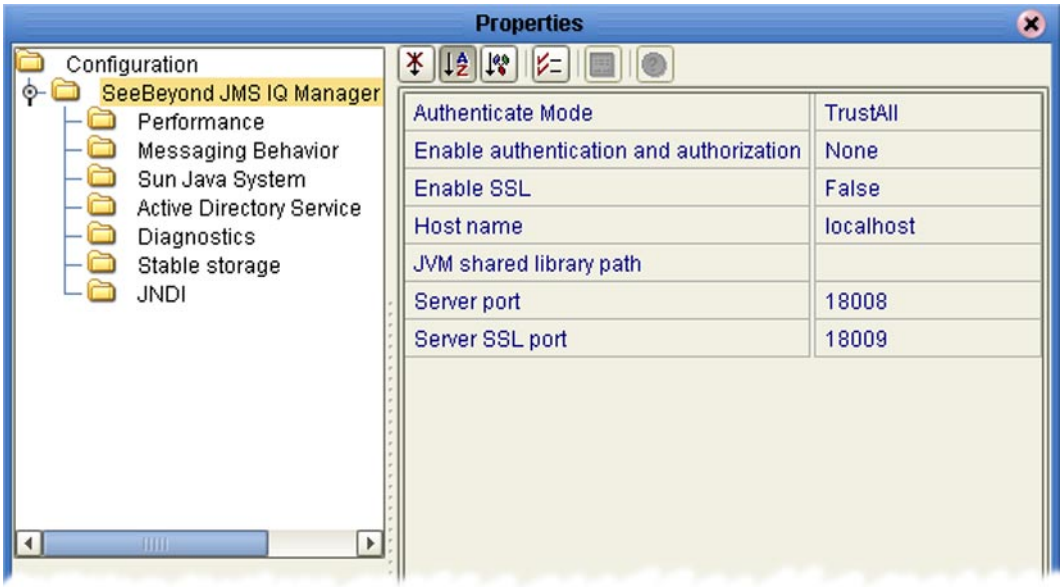


Table 90 JMS IQ Manager - General Configuration Properties List

Property	Description
Authenticate Mode	Specifies whether or not the JMS IQ Manager authenticates components attempting to connect to it. The options are **Authenticate** and **TrustAll**; the default value is **TrustAll**.
Enable authentication and authorization	Specifies whether or not security is enabled for the JMS IQ Manager, requiring a user name and password. The options are: ▪ **None** (no security—default value) ▪ **File** (use File realm) ▪ **SunONE** (use SunONE LDAP server) ▪ **AD** (use Active Directory LDAP server) If you specify either **SunONE** or **AD**, you must also configure your LDAP server. See **Sun Java System** on page 326 and **Active Directory Service** on page 327.
Enable SSL	Specifies whether or not to enable the Secure Sockets Layer (SSL) protocol for the JMS IQ Manager's TCP/IP connections. The options are **True** and **False**; the default value is **False**.
Host name	Specifies the name of the host system for the JMS IQ Manager. The default value is **localhost**.
JVM shared library path	Specifies the path to the Java Virtual Machine shared library. The value depends on the operating system under which the Logical Host is running—use the ellipsis (...) button to select the correct value. The default value is **../../jre/bin/client/jvm.dll**, for Windows platforms.
Server port	Specifies the TCP/IP port number; the default value depends upon the number of the Logical Host base port. Each JMS IQ Manager must have a unique port number per system.
Server SSL port	Specifies the port on which the JMS IQ Manager listens for SSL connections; the default value depends upon the number of the Logical Host base port.

Performance

The Performance properties allow you to specify memory usage for optimum system performance. See the *eGate Integrator JMS Reference Guide* for additional information.

Figure 274 Performance Configuration Properties

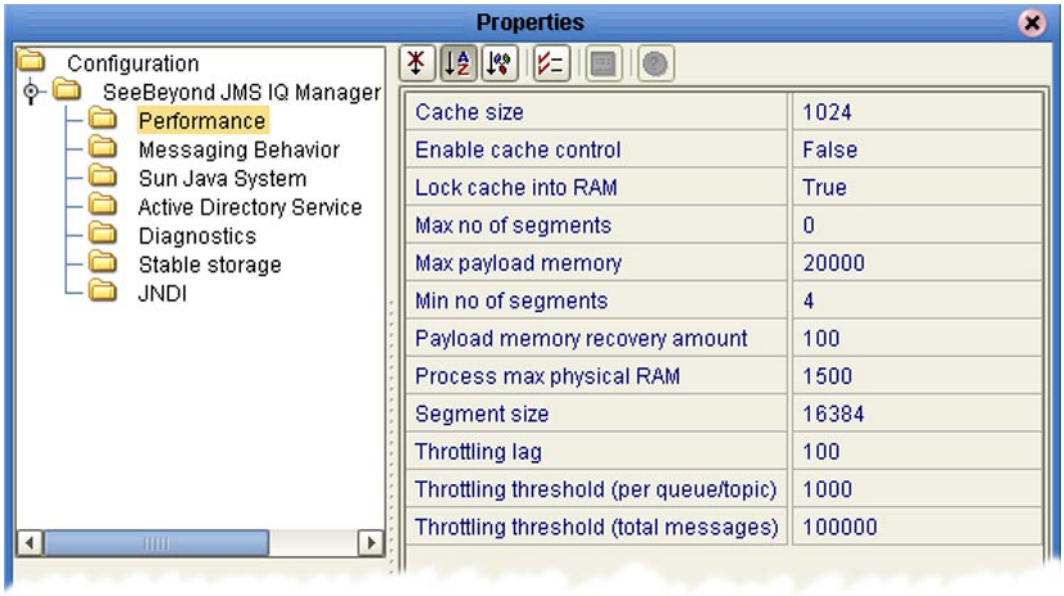


Table 91 Performance Configuration Properties List

Property	Description
Cache size	Specifies the total number of pages in the *read* cache (a page is 512 bytes on Windows and 1024 bytes on UNIX). The default value is **1024** pages (0.5 MB for Windows, 1 MB for UNIX).
Enable cache control	Specifies whether or not the JMS IQ Manager controls the cache synchronization to disk. The options are **True** and **False**; the default is **False**.
Lock cache into RAM	Specifies whether or not the cache is locked into physical memory. The options are **True** and **False**; the default is **True**.
Max. no. of segments	Specifies the upper limit for the number of database files that the JMS IQ Manager creates for its stable message storage. The default value is **0**, which causes the JMS IQ Manager to create files as needed—limited only by disk space.
Max. payload memory	Specifies the upper limit for the amount of cache, in kB, allocated for the server to keep message payloads in cache. The default value is **20,000** kB.

Table 91 Performance Configuration Properties List

Property	Description
Min. no. of segments	Specifies the minimum number of database files that the JMS IQ Manager initially creates for its stable message storage. The default value is **4** segments (files).
Payload memory recovery amount	Specifies the amount of cache, in kB, to recover in a recovery and cleanup operation. The default value is **100** kB.
Process max. physical RAM	Specifies the upper limit for the amount of RAM, in kB, allocated for use by the JMS IQ Manager as working memory (Windows platforms only). The default value is **1500** kB.
Segment size	Specifies the total number of pages in a single database file (segment). The default value is **16,384** pages (8 MB for Windows, 16 MB for UNIX).
Throttling lag	Specifies the number of messages that must be dequeued before message producers are no longer throttled. The default value is **100**.
Throttling threshold (per queue/topic)	Specifies the maximum number of messages after which all message producers for the message destination (topic or queue) are throttled. The default value is **1000** messages.
Throttling threshold (total messages)	Specifies the maximum number of messages for all message destinations, after which the JMS IQ Manager begins throttling producers. The default value is **100,000** messages.

Messaging Behavior

The Messaging Behavior properties allow you to configure the order of message delivery. See the *eGate Integrator JMS Reference Guide* for additional information.

Figure 275 Messaging Behavior Configuration Properties

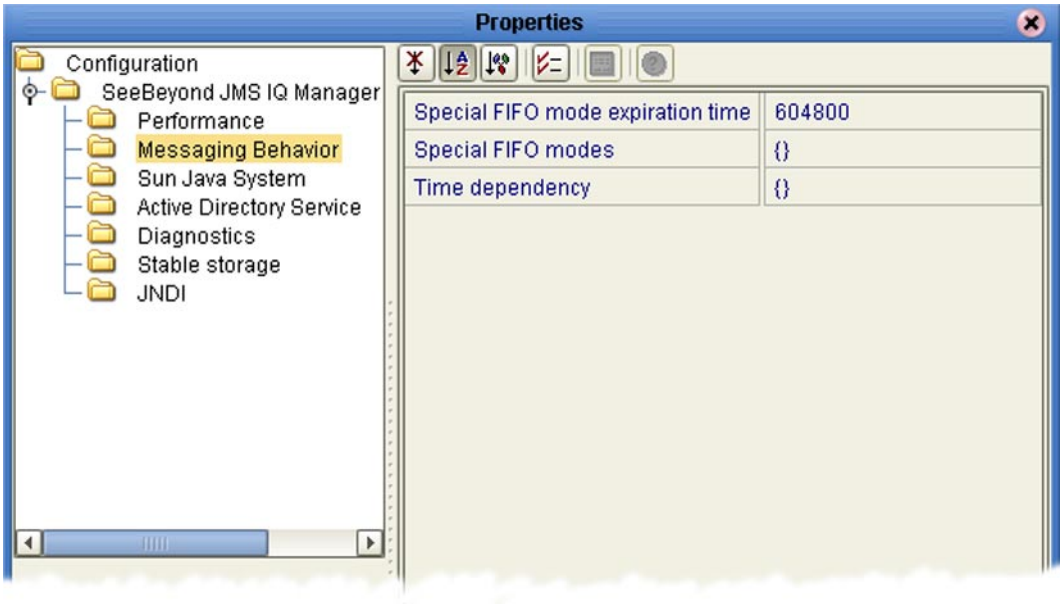


Table 92 Messaging Behavior Configuration Properties List

Property	Description
Special FIFO mode expiration time	Specifies the expiration time in seconds for first-in, first-out (FIFO) mode operation. The default value is **604,800** seconds (7 days).
Special FIFO modes	Specifies the FIFO delivery order: ▪ fully concurrent (**0**) ▪ protected concurrent (**1**) ▪ fully serialized (**2**) The default value is (**0**). See the *eGate Integrator JMS Reference Guide* for specification syntax.
Time dependency	Specifies whether or not the processing order is dependent on messages are associated with other destinations. The default value is **0**, indicating no such dependency. See the *eGate Integrator JMS Reference Guide* for syntax.

Sun Java System

These configuration properties are used only if you are using SunONE Directory Server as an LDAP server. See the *eGate Integrator System Administration Guide* for additional information.

Figure 276 Sun Java System Configuration Properties

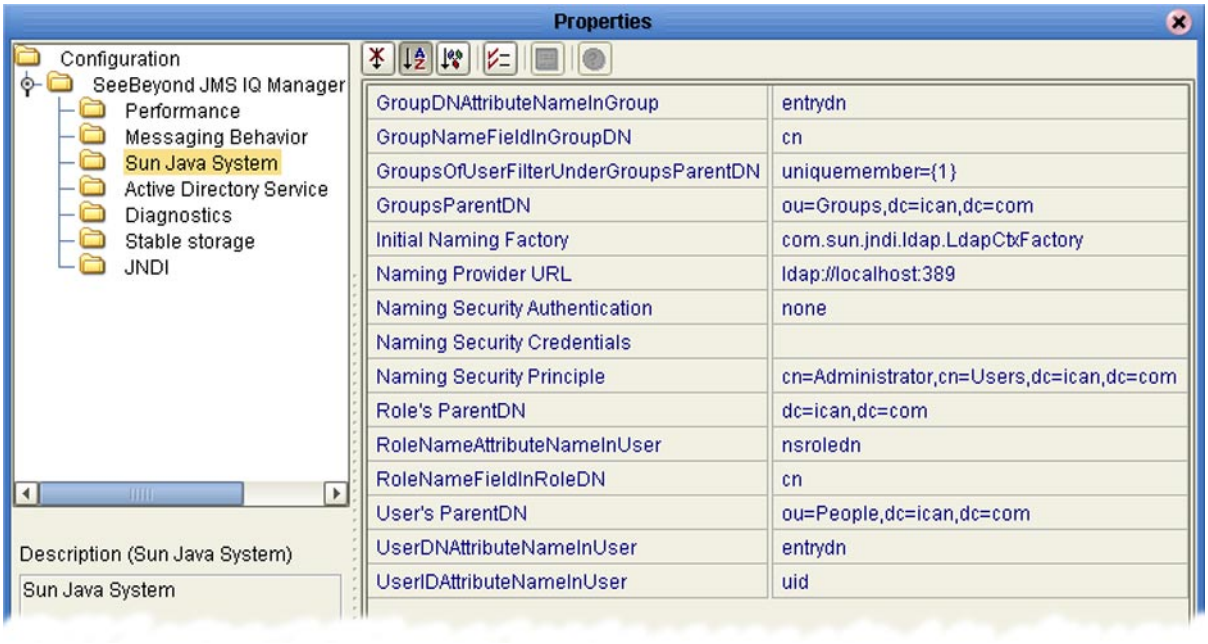


Table 93 Sun Java System Configuration Properties List

Property	Default Value
GroupDNAttributeNameInGroup	entrydn
GroupNameFieldInGroupDN	cn
GroupOfUserFilterUnderGroupsParentDN	uniquemember={1}
GroupsParentDN	cn=Groups,dc=ican,dc=com
Initial Naming Factory	com.sun.jndi.ldap.LdapCtxFactory
Naming Provider URL	ldap#10.18.73.63:489
Naming Security Authentication	none
Naming Security Credentials	
Naming Security Principle	cn=Administrator,cn=Users,dc=ican,dc=com
Role's ParentDN	dc=ican,dc=com
RoleDNAttributeNameInRole	nsroledn
RolesOfUserFilterUnderGroupsParentDN	cn
User's ParentDN	cn=People,dc=ican,dc=com
UserDNAttributeNameInUser	entrydn

Table 93 Sun Java System Configuration Properties List

Property	Default Value
UserIDAttributeNameInUser	uid

Active Directory Service

These configuration properties are used only if you are using the Microsoft Active Directory Service as an LDAP server. See the *eGate Integrator System Administration Guide* for additional information.

Figure 277 Active Directory Service Configuration Properties

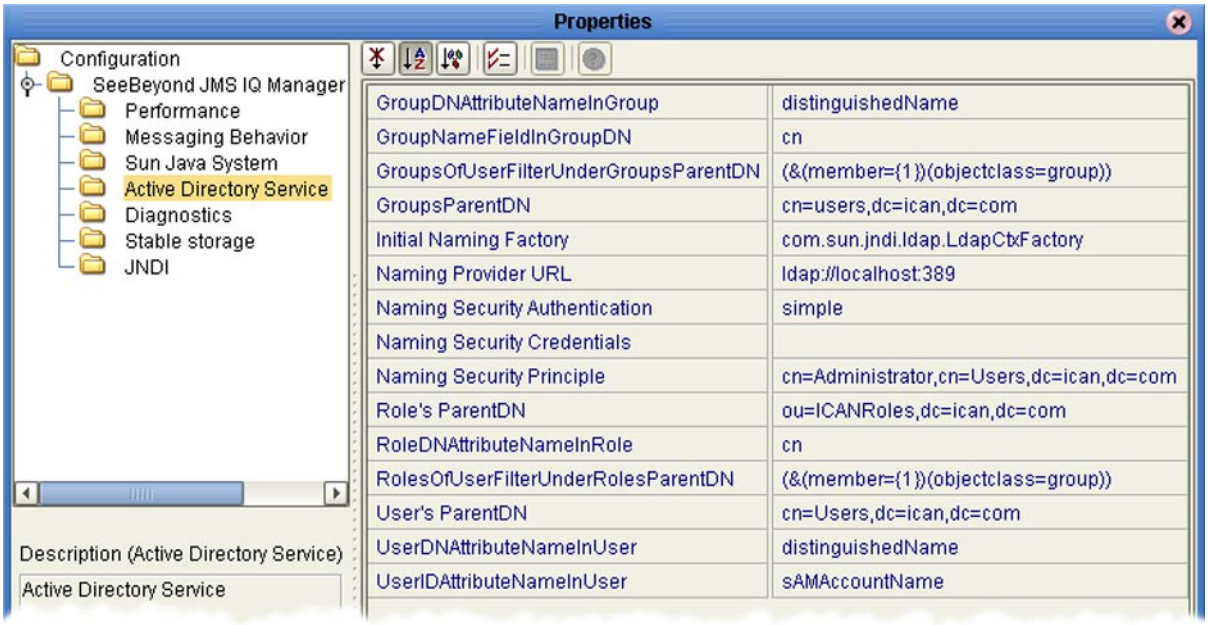


Table 94 Active Directory Service Configuration Properties List

Property	Default Value
GroupDNAttributeNameInGroup	distinguishedName
GroupNameFieldInGroupDN	cn
GroupOfUserFilterUnderGroupsParentDN	(&(member={1})(objectclass=group))
GroupsParentDN	cn=users,dc=ican,dc=com
Initial Naming Factory	com.sun.jndi.ldap.LdapCtxFactory
Naming Provider URL	ldap#10.18.73.63:389
Naming Security Authentication	simple
Naming Security Credentials	
Naming Security Principle	cn=Administrator,cn=Users,dc=ican,dc=com
Role's ParentDN	ou=ICANRoles,dc=ican,dc=com

Table 94 Active Directory Service Configuration Properties List

Property	Default Value
RoleDNAttributeNameInRole	cn
RolesOfUserFilterUnderGroupsParentDN	(&(member={1})(objectclass=group))
User's ParentDN	cn=users,dc=ican,dc=com
UserDNAttributeNameInUser	distinguishedName
UserIDAttributeNameInUser	sAMAccountName

Diagnostics

The Diagnostics properties allow you to configure the logging operations for the JMS IQ Manager. See the *eGate Integrator JMS Reference Guide* for additional information.

Figure 278 Diagnostics Configuration Properties

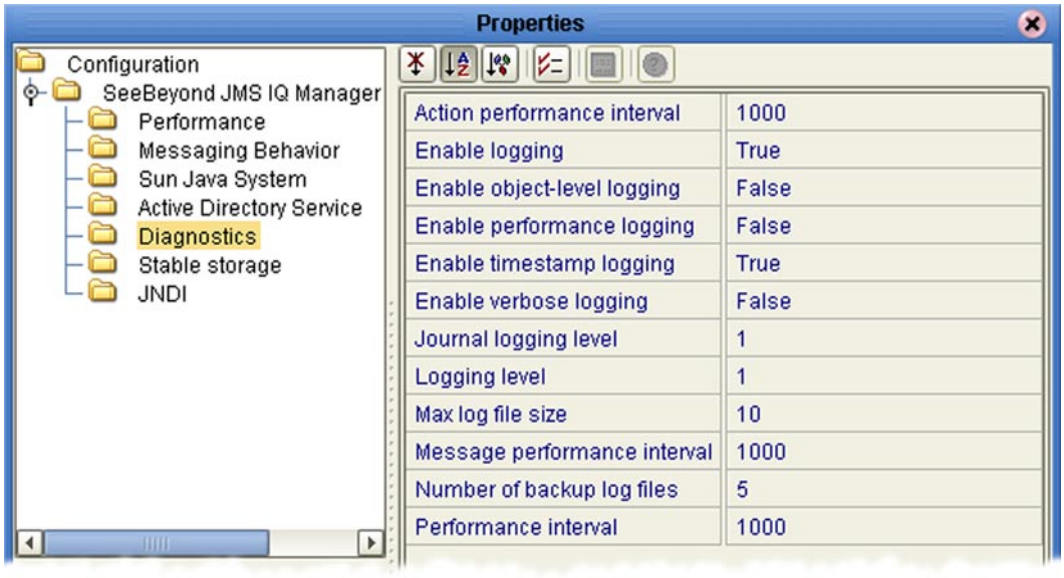


Table 95 Diagnostics Configuration Properties List

Property	Description
Action performance interval	Specifies how many times the ActionManager's *Update* function is called before logging its performance statistics. The default is **1000**.
Enable logging	Specifies whether or not diagnostic information is written to the JMS IQ Manager log file. **True** (the default) enables logging, **False** disables it.

Table 95 Diagnostics Configuration Properties List

Property	Description
Enable object-level logging	Specifies whether or not object (constructor/deconstructor) information is written to the JMS IQ Manager log file. **True** enables logging, **False** (the default) disables it.
Enable performance logging	Specifies whether or not performance information is written to the JMS IQ Manager log file. **True** enables logging, **False** (the default) disables it.
Enable timestamp logging	Specifies whether or not timestamp information is written to the JMS IQ Manager log file. **True** (the default) enables logging, **False** disables it.
Enable verbose logging	Specifies whether or not full-length messages are written to the JMS IQ Manager log file. **True** enables logging, **False** (the default) disables it.
Journal logging level	Specifies the threshold severity level at which the system issues informational, warning, and error messages, and writes them to the JMS IQ Manager journal log. ▪ all messages (**0**) ▪ warning, error, and fatal messages (**1**) ▪ error and fatal messages (**2**) ▪ fatal messages only (**3**) The default level is (**1**).
Logging level	Specifies the threshold severity level at which the system issues informational, warning, and error messages, and writes them to the JMS IQ Manager log. ▪ all messages (**0**) ▪ warning, error, and fatal messages (**1**) ▪ error and fatal messages (**2**) ▪ fatal messages only (**3**) The default level is (**1**).
Max. log file size	Specifies the maximum size of the JMS IQ Manager log file, in MB. The default is **10** MB.
Message performance interval	Specifies the number of messages the JMS IQ Manager processes before logging its performance statistics. The default is **1000**.
Number of backup log files	Specifies the maximum number of JMS IQ Manager backup log files in the stack. The default is **5**.
Performance interval	Specifies the number of IMessages the IMessageManager creates before logging its performance statistics. The default is **1000**.

Stable Storage

The Stable Storage properties allow you to configure the storage and journaling operations for the JMS IQ Manager. See the *eGate Integrator JMS Reference Guide* for additional information.

Figure 279 Stable Storage Configuration Properties

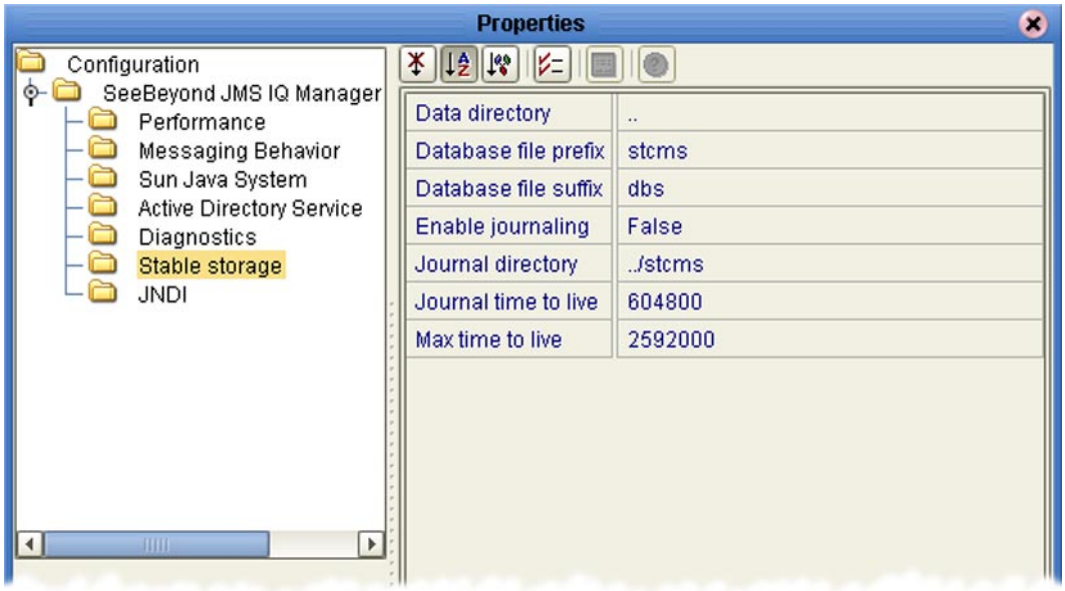


Table 96 Stable Storage Configuration Properties List

Property	Description
Data directory	Specifies the location for the JMS IQ Manager database files. The path can be either absolute or relative to the **logicalhost** directory. The default is **..** (one directory up from the **logicalhost** directory).
Database file prefix	Specifies the prefix for the JMS IQ Manager database file names. The default is **stcms** (for default file names of *stcms*.dbs*).
Database file suffix	Specifies the extension for the JMS IQ Manager database file names. The default is **dbs** (for default file names of *stcms*.dbs*).
Enable journaling	Enables or disables journaling, whereby every inbound message is automatically copied to the journal database. **True** enables, **False** disables; the default is **False**.
Journal directory	Specifies the location for the journal database files. The path can be either absolute or relative to the **logicalhost** directory. The default is **../stcms**.

Table 96 Stable Storage Configuration Properties List

Property	Description
Journal time to live	Specifies the maximum amount of time, in seconds, a journaled message persists before expiring. The default is **604,800** seconds (7 days).
Max. time to live	Specifies the maximum amount of time, in seconds, a live message persists before being removed from the queue. The default is **2,592,000** seconds (30 days).

JNDI

The JNDI configuration properties are not user-definable.

Chapter 10

Project Deployment

This chapter describes the process of creating deployment profiles and activating the deployed projects.

10.1 Deployment Profiles

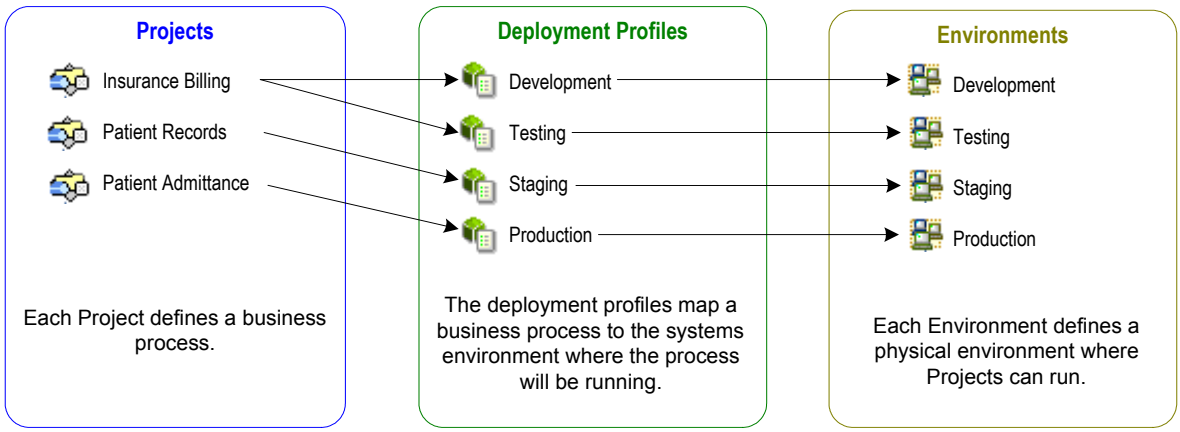
Deployment Profiles define specific instances of a Project in a particular Environment. A deployment profile contains information about the assignment of Services and Message Destinations to integration and message servers (JMS IQ Managers). It also contains version information for all relevant objects in the Project. The Enterprise Designer includes a Deployment Editor, which you can use to create and customize deployment profiles.

Note that:

- Each Project can have zero or more Deployment Profiles, but each of a Project's active Deployment Profiles must be in a separate Environment.
- Each Environment can have zero or more Deployment Profiles assigned to it, but any given Environment can have only one Deployment Profile from a given Project.

Repeating Figure 2 from the **System Overview** on page 27:

Figure 280 eGate Integrator Implementation Model



10.2 The Deployment Editor

The Deployment Editor (see Figure 281) allows you to create a new Deployment Profile or edit an existing one. 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**. To edit an existing Deployment Profile, simply click on its icon.

Figure 281 Deployment Editor Window

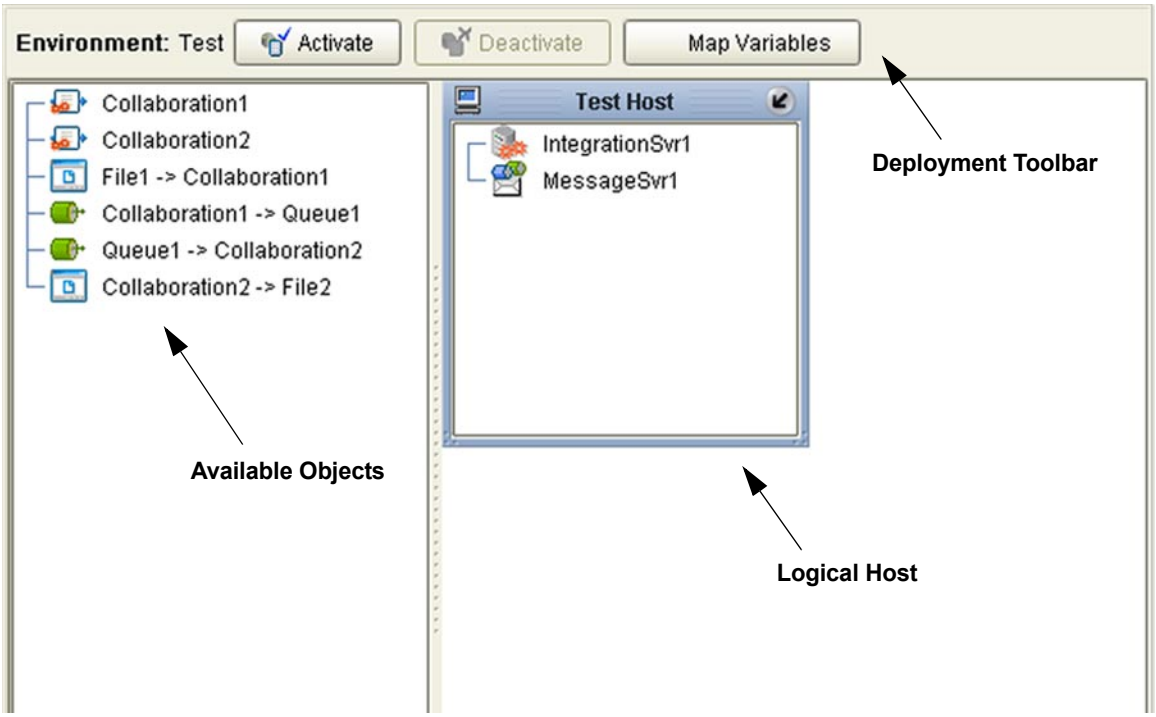


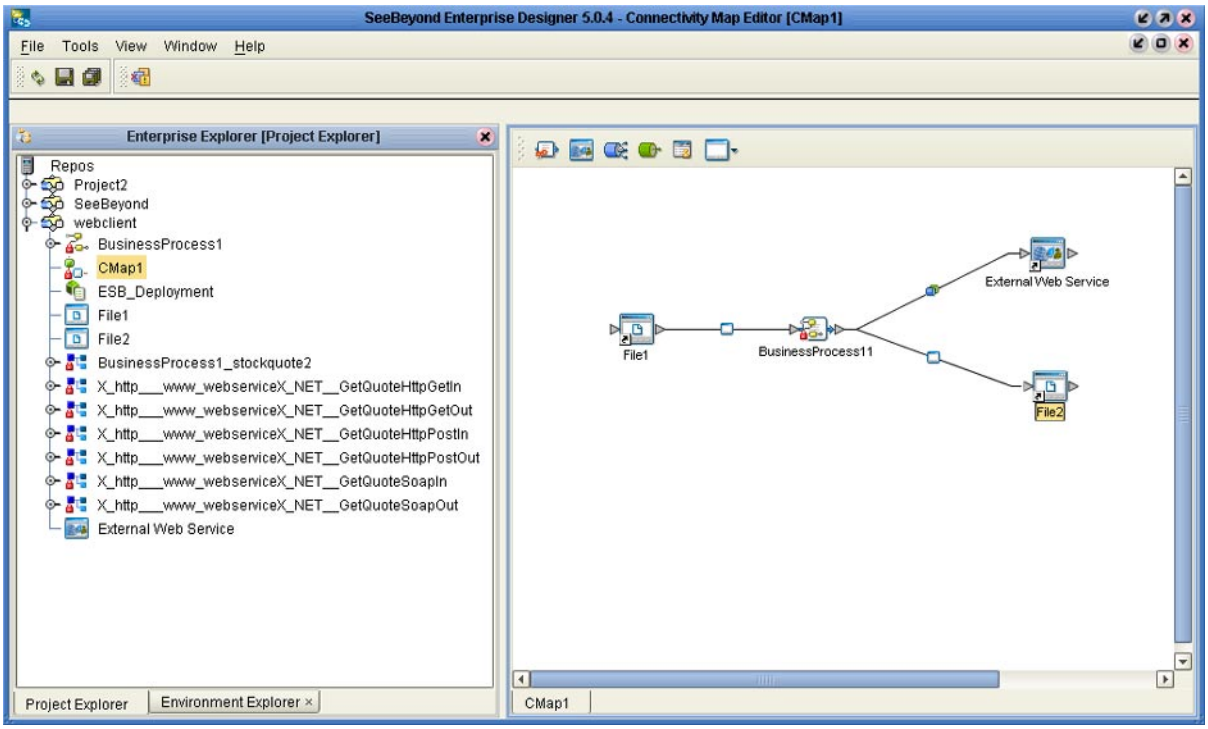
Table 97 Deployment Toolbar Buttons

Button	Function
Activate Reactivate	Starts the Project by creating an enterprise archive (EAR) file based on the Connectivity Map and linking this file with the application server. After activation, button changes to **Reactivate**. See **Activating and Deactivating Deployment Profiles** on page 337.
Deactivate	Stops the Project by terminating the link between the EAR file and the application server, sets the Deployment Profile to *inactive*, and saves to the Repository. See **Activating and Deactivating Deployment Profiles** on page 337.
Map Variables	Allows you to assign names and values to Project variables for the specific Deployment Profile. See **Mapping Variables** on page 340.

10.3 Creating a Deployment Profile

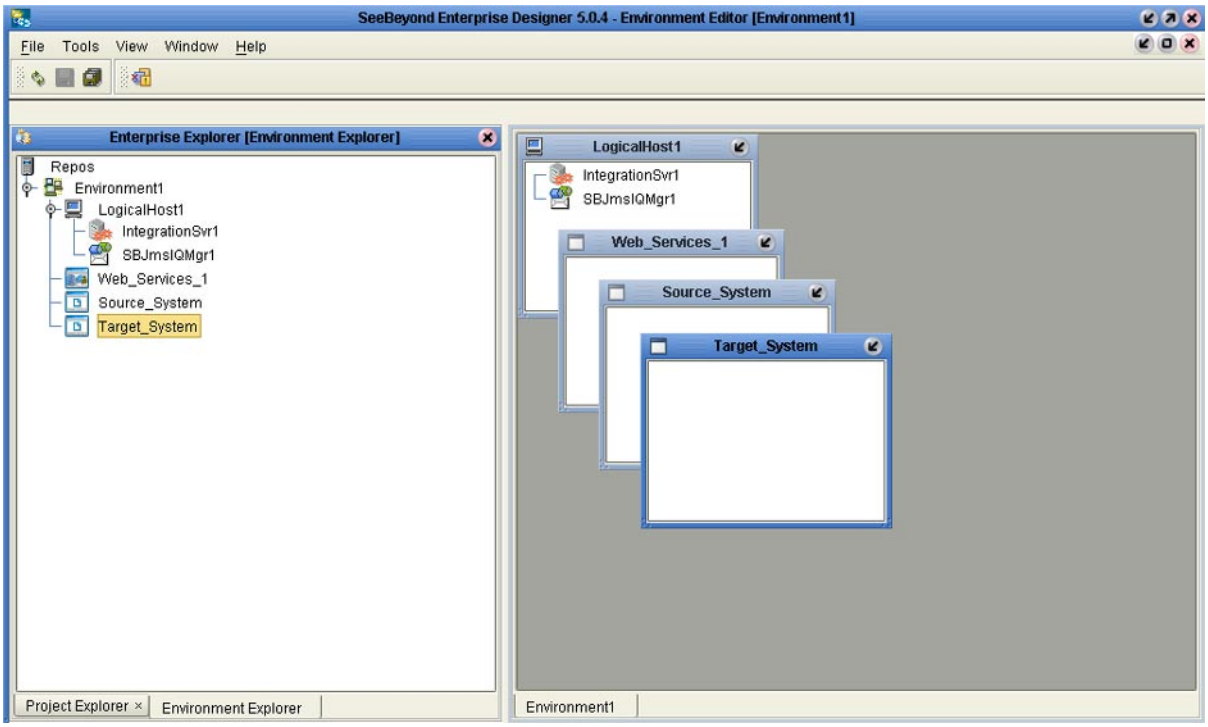
The Web Client Project shown in Figure 282 will be used as a deployment example.

Figure 282 Web Client Example Project



1. In the Environment Explorer, create an Environment and right-click on the Environment to display its context menu.
2. From the menu, select the Environment components you need and name them appropriately. They will appear as shown in Figure 283.

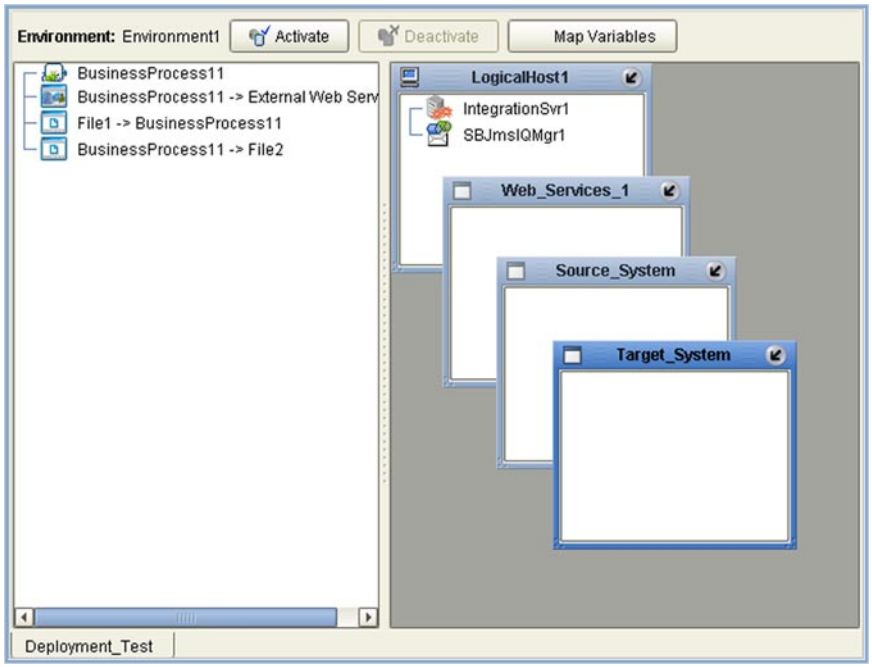
Figure 283 Web Client Example Environment



3 In the Project Explorer, right-click on the Project to display its context menu.

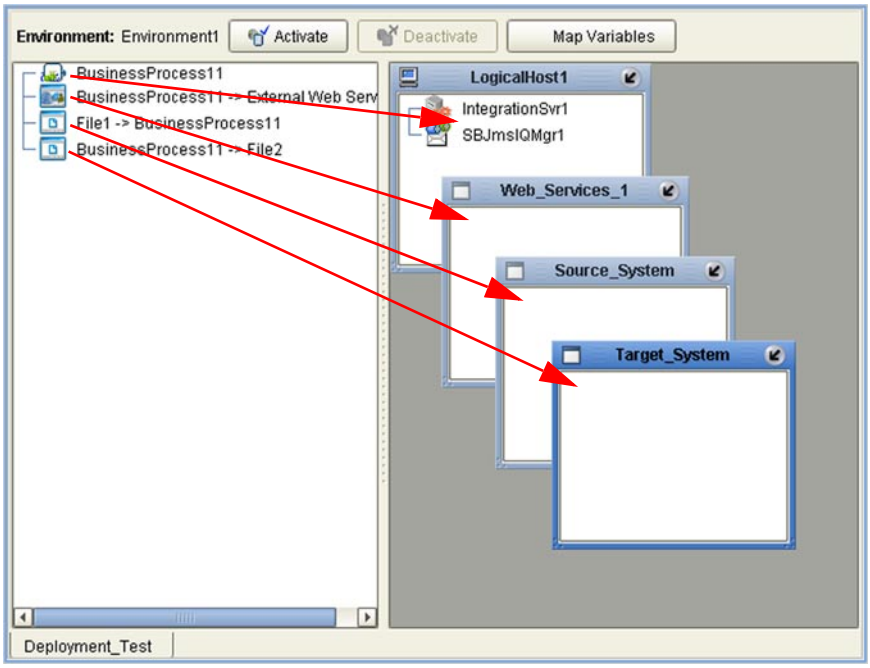
4 From the menu, select **New > Deployment Profile**. The Deployment Profile Editor appears, displaying the Environment you created previously (see Figure 284).

Figure 284 Example Deployment Profile (1)



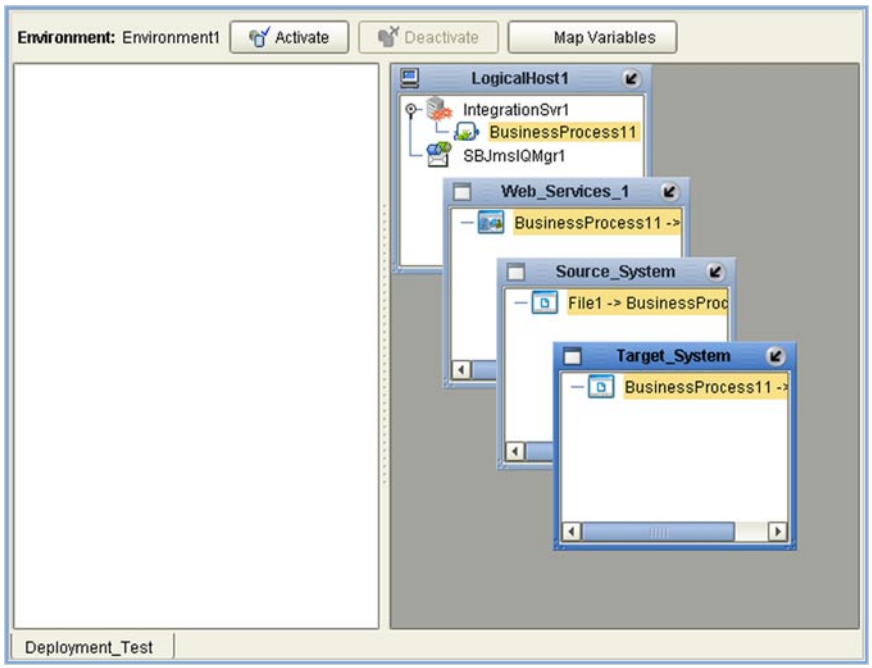
5 Drag the Project components from the left panel and drop them into the appropriate Environment components in the right panel, as illustrated in Figure 285.

Figure 285 Example Deployment Profile (2)



6 When the Environment components are fully populated, the left panel will be blank, as shown in Figure 286. You should now **Save** the profile.

Figure 286 Example Deployment Profile (3)



10.4 Activating and Deactivating Deployment Profiles

When activating or deactivating a Deployment Profile, you have the option of applying the changes to the Logical Host either immediately or at a later time. By activating the Deployment Profile without immediately applying the changes, you can check the validity of the entire Deployment Profile.

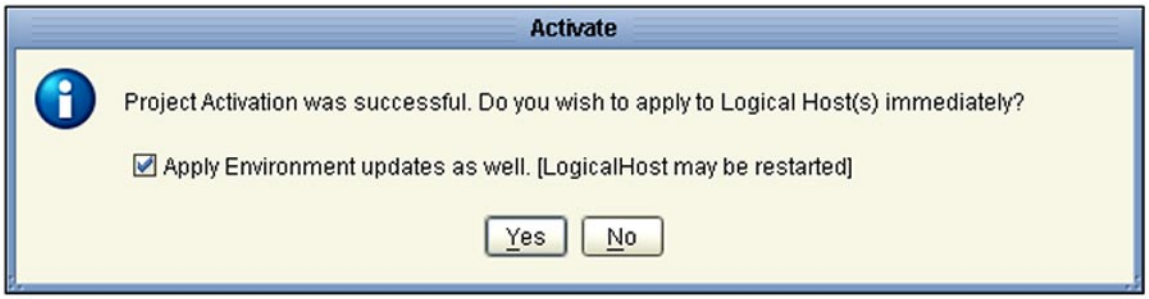
If you have multiple Deployment Profiles to deploy, you might work more efficiently by activating each of the Deployment Profiles without applying the changes, and then applying all of the changes to the Logical Host at a later time (*heed the warning given in the following procedures*).

10.4.1 Using Enterprise Designer

To activate a Deployment Profile

1. In the Project Explorer, select the Deployment Profile you want to activate.
2. Click the **Activate** button, which invokes the dialog box shown in Figure 287:

Figure 287 Activate Dialog Box

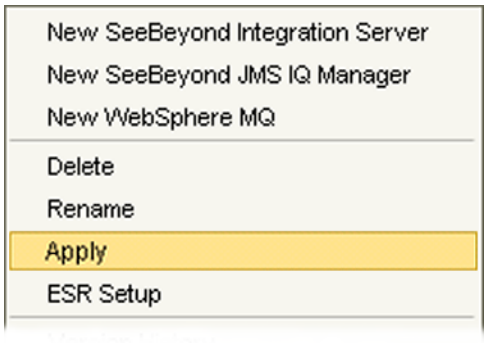


3. Select your response based on the following criteria:
   - If the Logical Host is running, and you wish to apply the changes immediately, click **Yes**.
   - If the Logical Host has not yet been bootstrapped, or you wish to apply the changes at a later time, click **No**.
4. If you click **Yes**, the information box shown in Figure 288 will appear after the changes have been applied. Click **OK** to proceed.

Figure 288 Success Information Box

5 To apply the changes at a later time, right-click the Logical Host and select **Apply** from the menu (see Figure 289). This will download the latest configuration from the Repository to that Logical Host.

Figure 289 Logical Host Context Menu - Apply



6 The information box shown in Figure 288 will appear after the changes have been completed. Click **OK** to proceed.

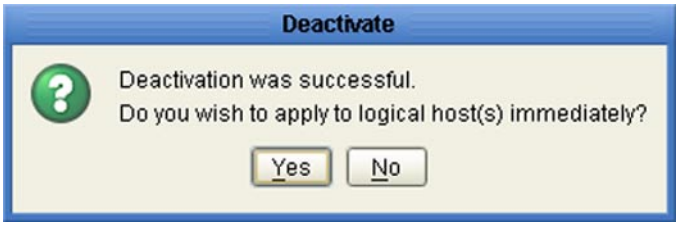
Important: *Use the ICAN Monitor to verify that your changes have been applied before again running the apply command.*

To deactivate a Deployment Profile

1 In the Project Explorer, select the Deployment Profile you want to deactivate.

2 Click the **Deactivate** button, which invokes the dialog box shown in Figure 290:

Figure 290 Deactivate Dialog Box



3 Select your response based on the following criteria:

- If the Logical Host is running, and you wish to apply the changes immediately, click **Yes**.
- If the Logical Host has not yet been bootstrapped, or you wish to apply the changes at a later time, click **No**.

4 To apply the changes at a later time, right-click the Logical Host and select **Apply** from the menu (see Figure 289). This will apply all of the changes for that Logical Host.

Important: *Use the ICAN Monitor to verify that your changes have been applied before again running the apply command.*

10.4.2 Using a Command-line Script

A script named **CmdLineUtil.bat** (or **.sh**) allows you to deploy and undeploy projects via the command line. The *apply* and *unapply* commands provide the same function as clicking **Activate** or **Deactivate**, and then clicking **Yes**, in Enterprise Designer.

This command-line utility allows you three options for deployment:

- **Apply to all Logical Hosts in the Environment** (you must provide the Environment name).
- **Apply to a single Logical Host** (you must provide the Environment name and the Logical Host name).
- **Apply to a specific Deployment Profile** (you must provide the Project name and the Deployment Profile name).

The command-line utility also allows you to activate or deactivate from any computer, not only the one on which the Logical Host is installed. User name/password authentication is performed before the utility can be used.

This utility must be downloaded using Enterprise Manager as described in the *SeeBeyond ICAN Suite Installation Guide*. Before running the utility:

- The JAVA_HOME environmental variable must be set on the host computer.
- The Repository must be running.
- You must first activate or deactivate the Project in Enterprise Designer.
- You must perform an initial bootstrap of the Logical Host(s) to download the components from the Repository.

Important: *Use the ICAN Monitor to verify that your changes have been applied before again running the apply or unapply command.*

10.5 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. Clicking the **Map Variable** button displays the Deployment Profile Mappings panel, where you can assign names (see Figure 291) and values (see Figure 292).

Figure 291 Deployment Profile Mappings

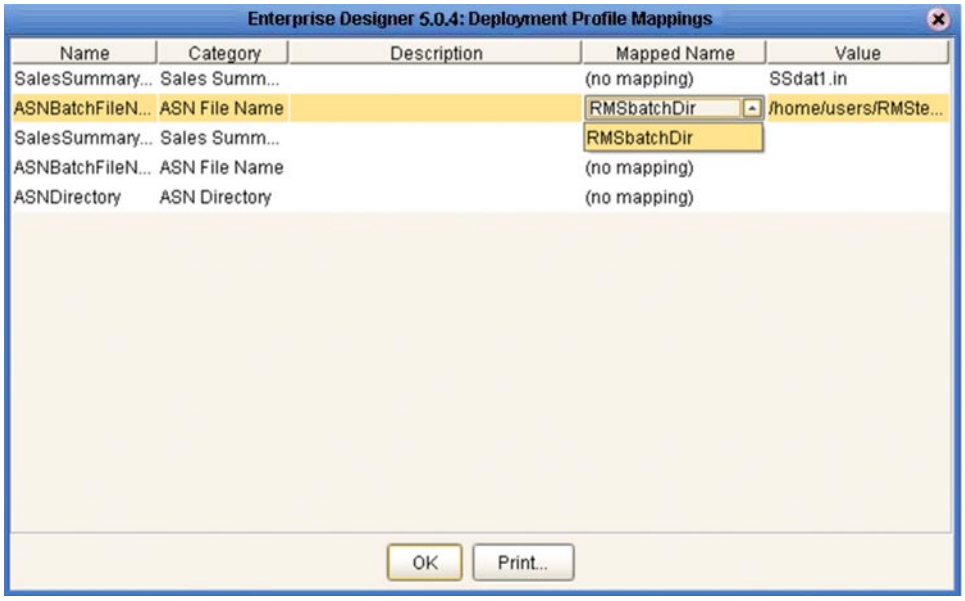
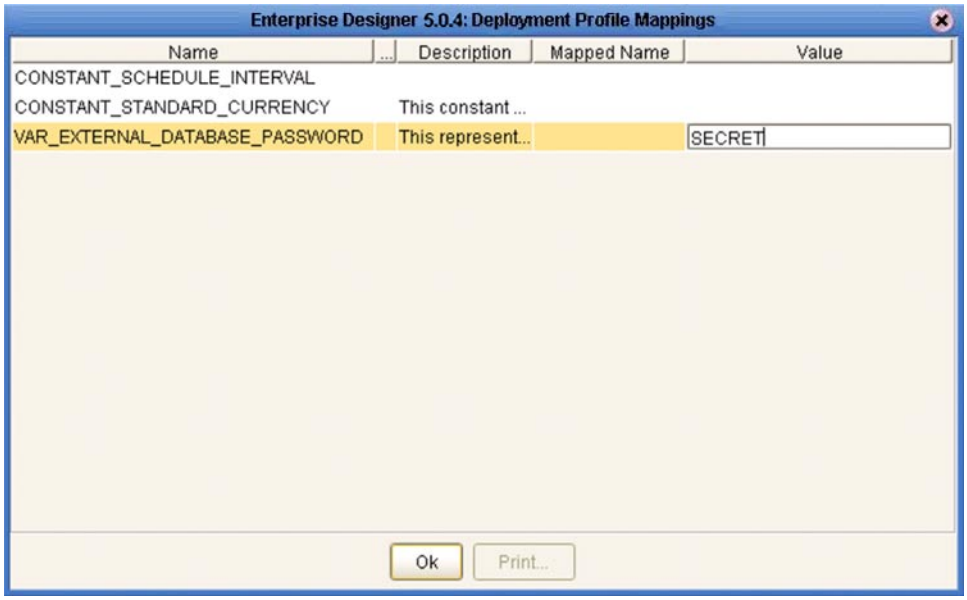


Figure 292 Project Variable Value Entry



10.6 Deploying Projects to Third-Party Servers

SeeBeyond's eGate Integrator allows you to develop Projects using Enterprise Designer and deploy them to a BEA WebLogic or IBM WebSphere environment. The SAR files for these third-party products must be installed prior to deployment, as described in the *eGate Integrator JMS Reference Guide*.

Because of the versions of the Java Connection Architecture supported by WebLogic and WebSphere, the following restrictions apply:

- Services deployed to WebLogic or WebSphere are restricted to those internal to eGate Integrator itself (between message destinations), and those associated with outbound eWays.
- Not all SeeBeyond eWays support third-party servers. Check the individual eWay User's Guides regarding such support, and also any additional configuration that may be necessary for compatibility with WebLogic or WebSphere.

10.6.1 BEA WebLogic

Note: *Before using the WebLogic JMS, you must install additional* ***.jar*** *files as described below. For additional information, see the eGate Integrator JMS Reference Guide.*

To install additional .jar files

1. Download the **log4j.jar** file from the location below (location subject to change).

   http://jakarta.apache.org/log4j/docs
2. Download the **xerces.jar** file from the location below (location subject to change).

   http://xml.apache.org/dist/xerces-j
3. Copy the following files from ***ICAN-root*****\edesigner\usrdir\modules**:
   - **com.stc.eventmanagementapi.jar**
   - **com.stc.eventmanagementimpl.jar**
   - **com.stc.jms.stcjms.jar**
4. Place all **.jar** files into the ***ICAN-root******weblogic8x*****\server\lib** directory.
5. Add the **.jar** files to the *set CLASSPATH* segment of the **startWLS.cmd** file located in the ***ICAN-root******weblogic8x*****\server\bin** directory. The text to be added is:

```
%WL_HOME%\server\lib\log4j.jar;%WL_HOME%\server\lib\xerces.jar;
%WL_HOME%\server\lib\ com.stc.eventmanagementapi.jar;
%WL_HOME%\server\lib\ com.stc.eventmanagementimpl.jar;
%WL_HOME%\server\lib\ com.stc.jms.stcjms.jar
```

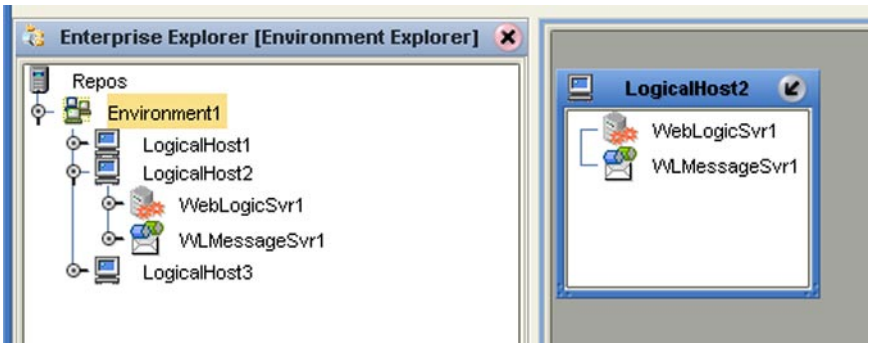
To deploy an eGate Project to a BEA WebLogic 8.0 or 8.1 environment

1. Create the following components in Enterprise Designer (see Figure 293):
   - A A new environment
   - B A Logical Host

C A WebLogic J2EE application server

D A WebLogic JMS message server

Figure 293 WebLogic Deployment (1)

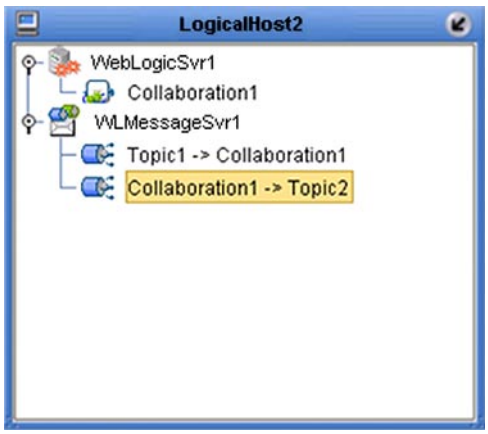


2 Create a new Deployment Profile to bind the Connectivity Map to the new WebLogic environment (see Figure 294).

A Drag the two topics and drop onto the WebLogic message server.

B Drag the Collaboration and drop onto the WebLogic application server.

Figure 294 WebLogic Deployment (2)



3 Activate the Deployment Profile.

Activating the Deployment Profile creates an Environment Archive (EAR) file, which contains all files necessary to create and run an application in WebLogic. This file can be found in the following location:

```
ICAN-root\repository\data\files\WLEnvironmrntName\
    ProjectName_DeploymentProfileName.ear
```

Note: *The remainder of this procedure is performed in the WebLogic user interface, and is only outlined here. Please refer to your BEA WebLogic documentation for current information regarding interface layout and deployment details.*

4 Start the BEA WebLogic server.

5 Navigate to **Server Administration Console > Deployments > Applications**.

6 Perform the following steps:

A Add a new JMS Connection Factory.

B Enter a JNDI name for the JMS Connection Factory:

```
jms\connectionfactory\xa-topic\
    LogicalHostName_MessageServerName
```

For example, the default name would be:

```
jms\connectionfactory\xa-topic\LogicalHost1_WLMessageSvr1
```

C Verify that the WebLogic JMS Server Destination names for topics match those in eGate.

D Select **Deploy a new Application**.

E Upload and install the EAR file described in step 3.

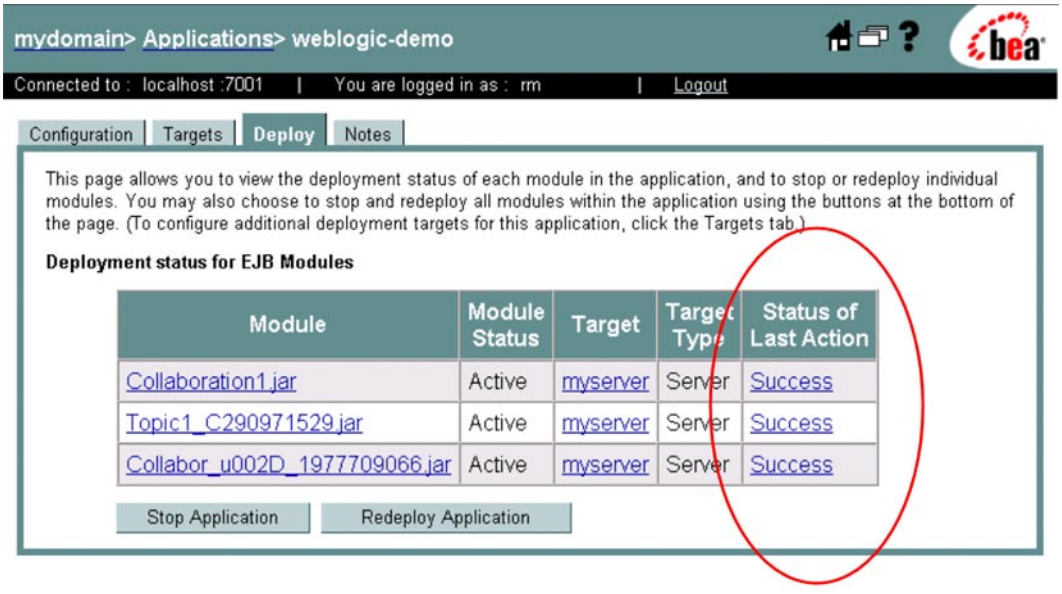
F Select the EAR file you just installed as the archive for the new application.

G Enter a name for the new application.

H Click **Deploy**.

I Verify the success of the deployment (see Figure 295, which shows a WebLogic 8.1 example).

Figure 295 WebLogic Deployment Verification



10.6.2 IBM WebSphere

Note: *Before using the WebSphere JMS, you must install additional* ***.jar*** *files as described below. For additional information, see the eGate Integrator JMS Reference Guide.*

To install log4.jar

1 Download the **log4j.jar** file from the location below (location subject to change).

http://jakarta.apache.org/log4j/docs

2 Copy the following files from ***ICAN-root*\edesigner\usrdir\modules**:

- **com.stc.antlrimpl.jar**
- **com.stc.eventmanagementapi.jar**
- **com.stc.eventmanagementimpl.jar**
- **com.stc.jms.stcjms.jar**

3 Place all **.jar** files into the ***ICAN-root*\WebSphere\AppServer\lib** directory.

To deploy an eGate Project to an IBM WebSphere 5.0.x or 5.1 environment

1 Create the following components in Enterprise Designer (see Figure 296):

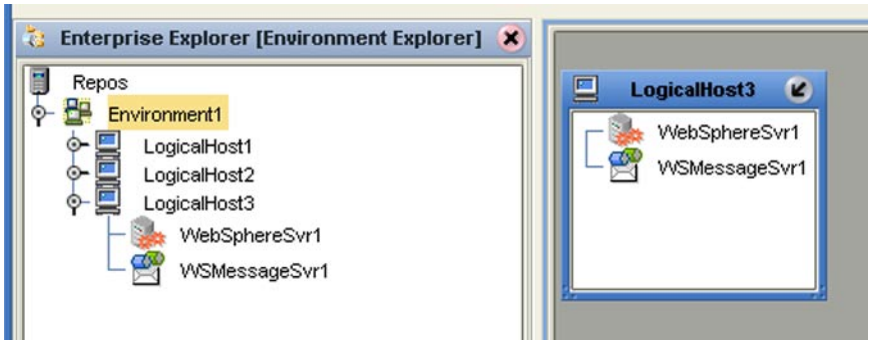
A A new environment

B A Logical Host

C A WebSphere J2EE application server

D A WebSphere JMS message server

Figure 296 WebSphere Deployment (1)

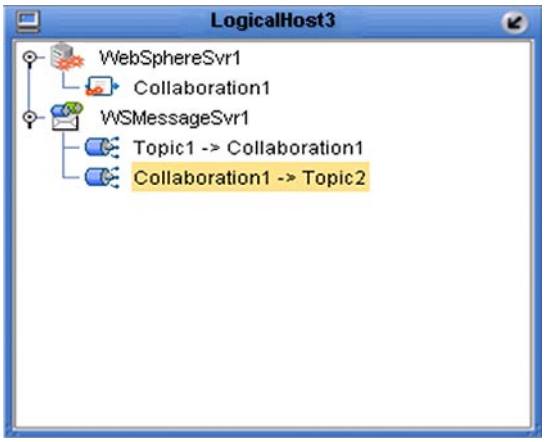


2 Create a new Deployment Profile to bind the Connectivity Map to the new WebSphere environment (see Figure 297).

A Drag the two topics and drop onto the WebSphere message server.

B Drag the Collaboration and drop onto the WebSphere application server.

Figure 297 WebSphere Deployment (2)



3 Activate the Deployment Profile.

The activated Deployment Profile creates an Environment Archive (EAR) file, which contains all files necessary to create and run an application in WebSphere. This file can be found in the following location:

ICAN-root\repository\data\files***WSEnvironmentName***\
ProjectName_DeploymentProfileName.ear

Note: *The remainder of this procedure is performed in the WebSphere user interface, and is only outlined here. Please refer to your IBM WebSphere documentation for current information regarding interface layout and deployment details.*

4 Start the IBM WebSphere server.

5 From the Administrative Console, navigate to **Servers > Application Servers > *server_name* > Message Listener Service > Listener Ports**.

6 Add a new Listener port.

7 Enter a Connection Factory JNDI name for the new port:

jms\connectionfactory\xa-topic\
LogicalHostName_MessageServerName

For example, the default name would be:

jms\connectionfactory\xa-topic***LogicalHost1_WSMessageSvr1***

This binds the JNDI name with the WebSphere Message Server Listener port.

8 From the Administrative Console, navigate to **Applications > Enterprise Applications > Install New Application**.

9 In *Preparing for the application installation*:

A Enter the path for the EAR file described in step 3 and click **Next**.

B Select **Generate Default Bindings** and click **Next**.

10 In *Step 1, Provide options ...*:

A Check **Deploy EJBs**.

B Enter the application name.

C Click **Next**.

11 In *Step 2, Provide options ...*, click **Next**.

12 In *Step 3, Provide Listener Ports ...*, accept the default value and click **Next**.

Note: *The Listener port number should match the port number entered in step 6.*

13 In *Step 4, Provide JNDI Names ...*, accept the default value and click **Next**.

14 In *Step 5, Provide EJB references ...*, accept the default value and click **Next**.

15 In *Step 6, Map resource references ...*, enter the JNDI name from step 7, and click **Next**.

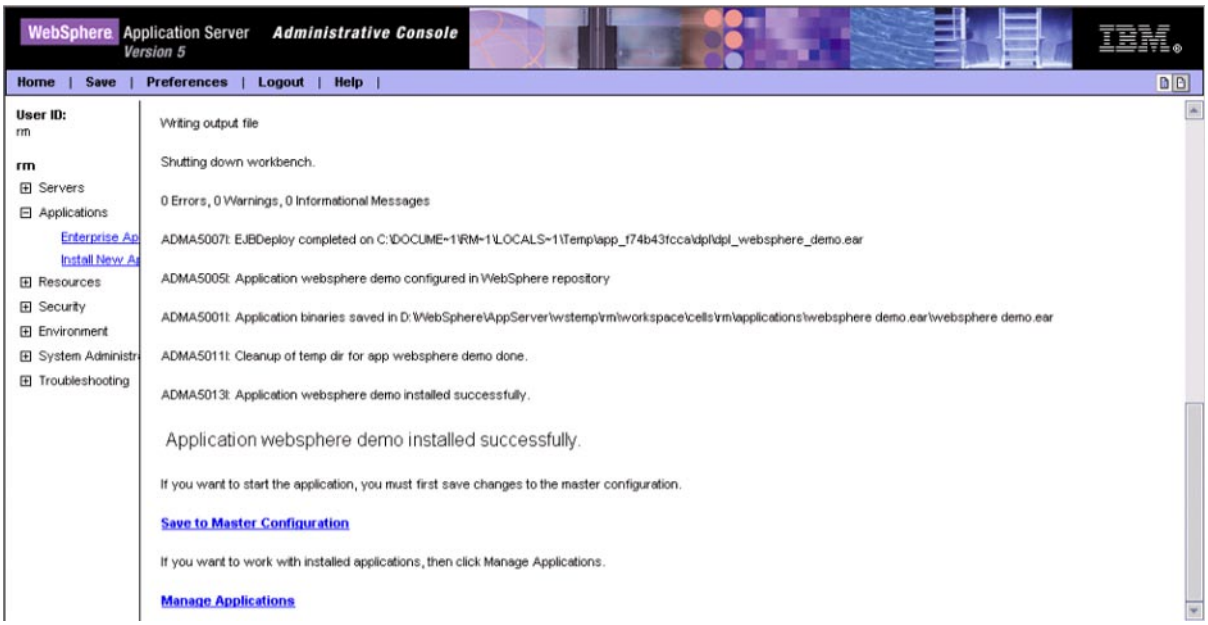
16 In *Step 7, Map modules ...*, check all modules and click **Next**.

17 In *Step 8,* (protection levels), check all modules and click **Next**.

18 In *Step 9, Summary,* click **Finish**.

19 Verify the success of the deployment (see Figure 298, which shows a WebSphere 5 example).

Figure 298 WebSphere Deployment Verification



Chapter 11

Web Services

This chapter describes the use of the Web Services capability of eGate Integrator, acting with other components of the ICAN Suite.

11.1 Overview

Basically, Web Services enables communication and data transfer between diverse applications using the Internet. In doing so, it provides a means for implementing EAI (Enterprise Application Integration) within an organization, or B2B (Business-to-Business) integration between partner organizations. This capability is achieved by wrapping back-end systems to present a common, standardized interface to the connecting network.

Four related technologies are used to transform and transport data within Web Services:

- **XML** (Extensible Markup Language)

  Provides a language for defining both the data itself and the way to process it.

- **WSDL** (Web Services Description Language)

  Defines the interfaces, data types, interactions, and mappings used in the Web Services. WSDL files are used to invoke and operate Web services on the Internet and to access and invoke remote applications and databases.

- **SOAP** (Simple Object Access Protocol)

  Defines a communications envelope that is mappable to HTTP and provides a format for transmitting XML documents over a network.

- **UDDI** (Universal Description, Discovery, and Integration)

  Provides a mechanism for storing and categorizing information that allows publication of services and discovery of external services.

11.2 SeeBeyond Web Services

eGate Integrator provides the capability to create either a client or a server to receive WSDL file from a remote server, or send WSDL files to a remote client. eGate works in conjunction with eInsight Business Process Manager, in which the associated business processes are developed. See **Building a Web Client** on page 357 and **Building a Web Server** on page 364.

The ICAN Suite contains the following components that implement the Web Services capability:

- **WSDL Wizard**

  The WSDL Wizard creates an OTD from a WSDL file. See **Using the WSDL Wizard** on page 132.

- **WSDL Editor**

  See the *eInsight Business Process Manager User's Guide*.

- **WSDL Interface Designer**

  See the *eInsight Business Process Manager User's Guide*.

- **WSDL Viewer**

  See the *eInsight Business Process Manager User's Guide*.

- **UDDI Registry**

  All objects represented in the ICAN Repository that can be accessed as Web services are presented via a UDDI registry. See **UDDI Registry** on page 349.

11.3 UDDI Registry

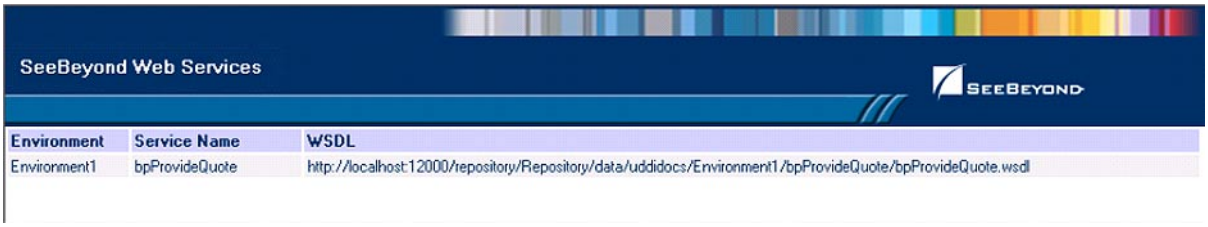
In general, all ICAN objects that expose themselves as Web services (such as eInsight business processes) are listed in a UDDI registry. The deployment activation process automatically publishes entries to the UDDI registry and creates the necessary sections in the WSDL files to expose it as a Web service.

This registry can be viewed on the SeeBeyond Web Services page (see Figure 299), part of Enterprise Manager. The URL of the SeeBeyond Web Services page is:

```
http://hostname:portnumber/stcuddi
```

*Note: The **hostname** is the fully-qualified, network-addressable host name of the server where you installed the Repository. The **portnumber** is the number of the port you entered during installation of the Repository.*

Figure 299 SeeBeyond Web Services Page



Each entry on the SeeBeyond Web Services page includes:

- The ICAN environment name.
- The actual (Web) Service name.
- The URL for the associated WSDL file.

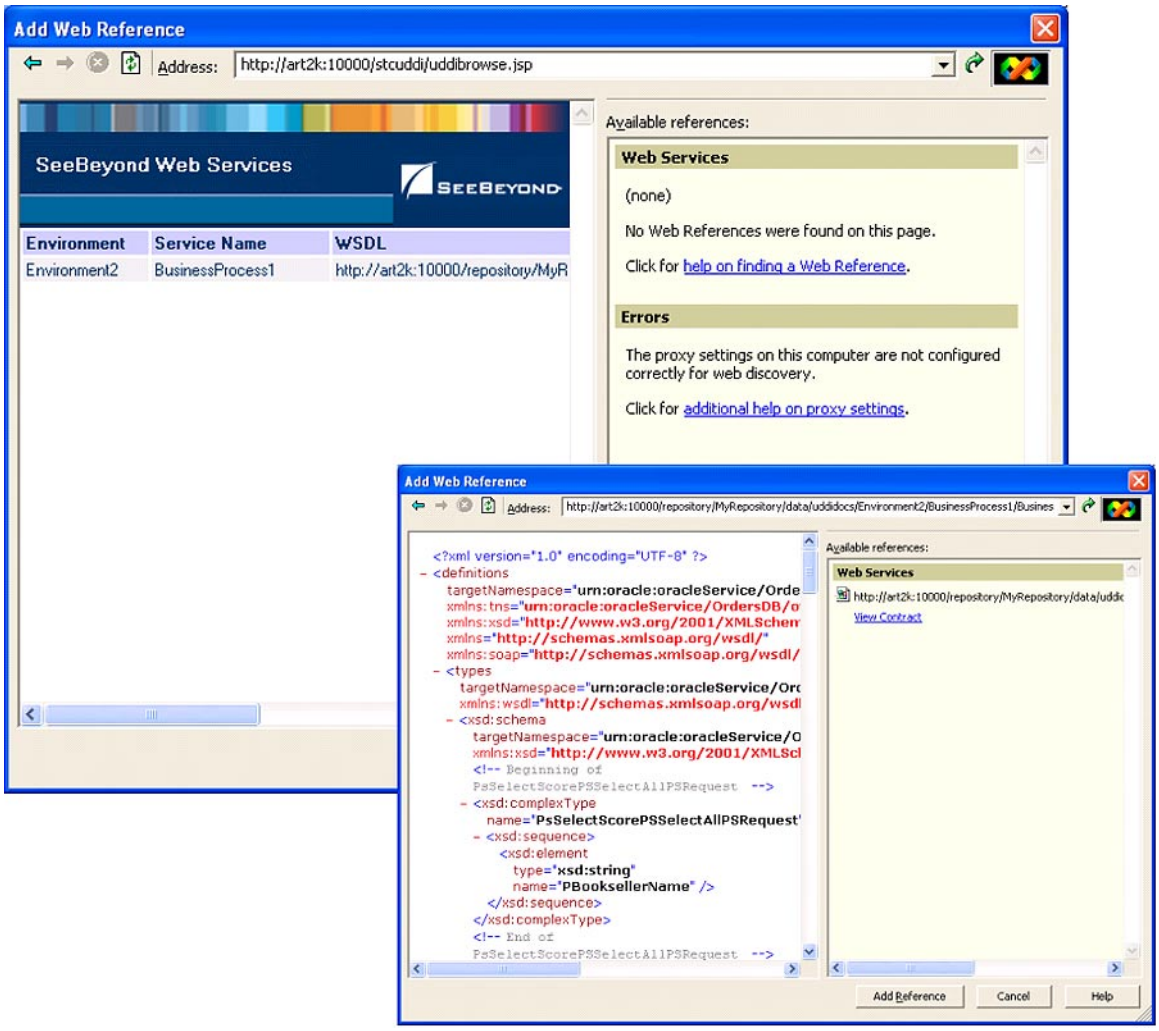
Select an entry (line item) to display its WSDL file, as shown in Figure 300.

Figure 300 Example Web Service WSDL File

```
<?xml version="1.0" encoding="UTF-8" ?>
- <definitions targetNamespace="http://seebeyond/quoteservice" xmlns:tns="http://seebeyond/quoteservice"
    xmlns:sbynpx="http://bpel.seebeyond.com/hawaii/5.0/privateExtension/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  - <types targetNamespace="http://seebeyond/quoteservice">
    - <xsd:schema targetNamespace="http://seebeyond/quoteservice"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <xsd:element type="xsd:string" name="QuoteRequest" />
        <xsd:element type="xsd:string" name="QuotePrice" />
        <xsd:element type="xsd:string" name="QuoteCompany" />
      </xsd:schema>
    </types>
  - <message name="msgProvideQuoteRequest">
      <part name="TICKER" element="tns:QuoteRequest" />
    </message>
  - <message name="msgQuoteResponse">
      <part name="price" element="tns:QuotePrice" />
      <part name="company" element="tns:QuoteCompany" />
    </message>
  - <portType name="ptProvideQuote">
```

The SeeBeyond UDDI Registry can be used in a third party tool, for example Microsoft Visual Studio (see Figure 301). In Visual Studio's Solution Explorer, right-click on *References* and enter the URL of the SeeBeyond Web Services page as a Web Reference.

Figure 301 Microsoft Visual Studio Example



eGate Integrator can exchange data with Internet and Web Services applications using the Web Services Description Language (WSDL). This language is XML-based and is used to define Web services and describe how to access them. The WSDL OTD Wizard is used to build OTDs that are used in the Project Collaborations (see **Using the WSDL Wizard** on page 132).

11.3.1 Using UDDI Browsers

A third-party UDDI Browser can be used to access the SeeBeyond UDDI Registry for publishing and inquiry. To add the SeeBeyond UDDI Registry to the browser, you need to enter the information given in Table 98 into the browser's *Add UDDI Registry* facility.

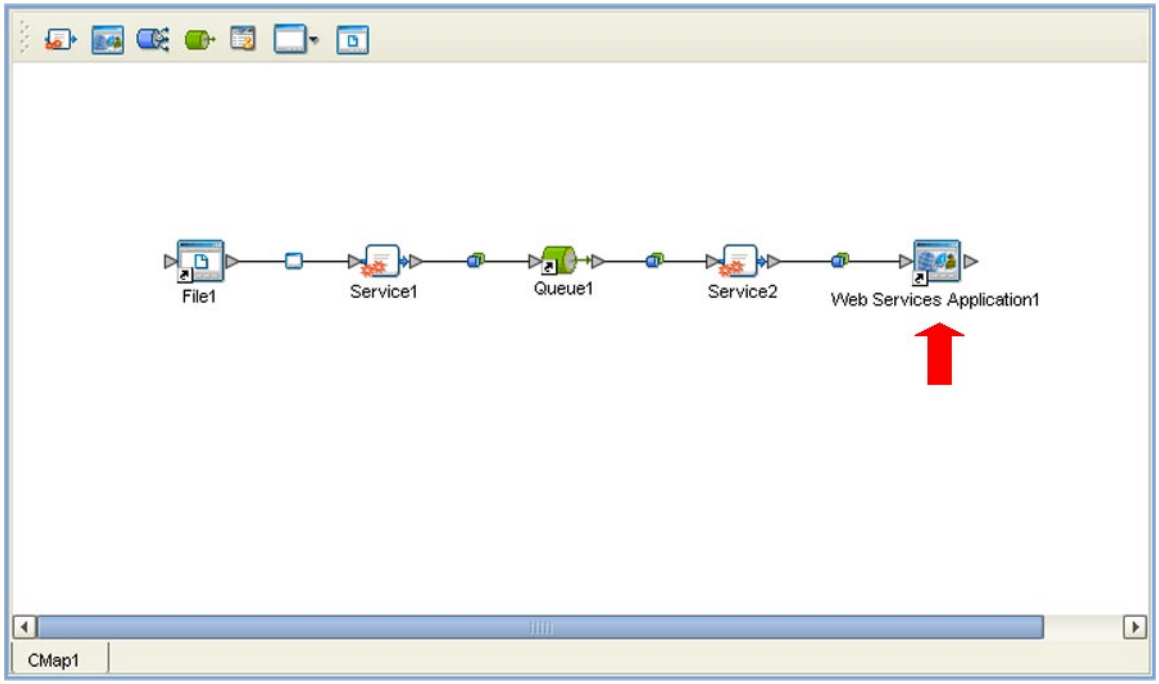
Table 98 UDDI Registry Information

Parameter	Value
Name	SeeBeyond UDDI
Inquiry URL	http://***hostname:portnumber***/stcuddi/inquiry
Publish URL	http://***hostname:portnumber***/stcuddi/publish
Username	***your_username***
Password	***your_password***

11.4 Web Services Application

A Web Services Application (see Figure 302) represents either an external Web Service that can be invoked by a Web Client containing an eInsight Business Process, or a Web Server containing a Business Process that is exposed as a Web Service. In the case of the Web Server, the exposed Web Service is listed in the SeeBeyond UDDI Registry.

Figure 302 Enterprise Designer - Connectivity Map Editor



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. See **Web Services External System** on page 353.

11.5 Web Services External System

A Web Services External System represents a system containing a Web Services Application. The Properties dialog box (see Figure 303) appears automatically when you create a New Web Services External System in the Environment Explorer. You can also invoke the dialog box by right-clicking the Web Services Application in the Enterprise Explorer and selecting **Properties**. You should assign the desired values for the listed properties.

Figure 303 Web Service External System Properties Dialog Box

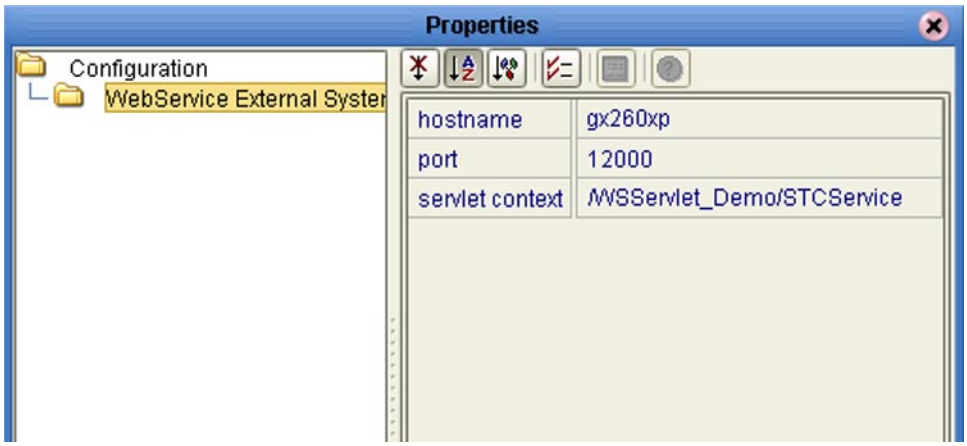


Table 99 Web Service Application Properties

Property	Description
hostname	Name of the computer hosting the application.
port	Port on which the application can be accessed.
servlet context	The path and name of the Web Services Application.

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 appropriately. 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**.

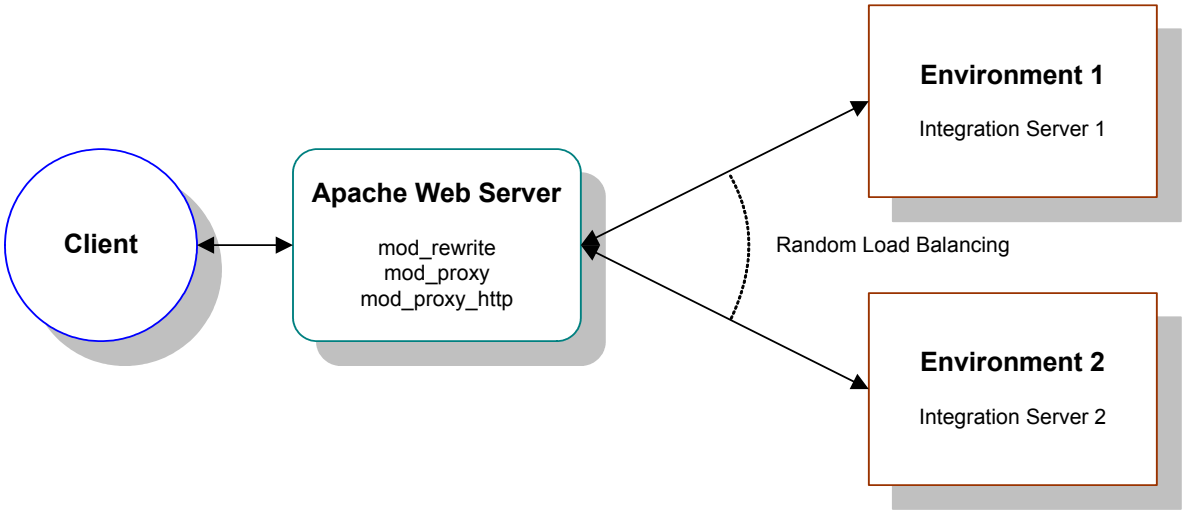
11.6 Load Balancing

You can build a system that will load balance HTTP requests in a way suitable for ICAN Web Services by using an Apache-based Web server. To accomplish this you will need the following:

1. An Apache-based HTTP Web server such as Apache 2.0 or IBM HTTP 2.0.
2. Modify the Apache configuration file to enable **mod_rewrite**, **mod_proxy**, and **mod_proxy_http**.
3. Create a mapping file that includes the name of your business process that is published as a Web Service and a list of URL endpoints where it is available.
4. Add the **mod_rewrite** rewrite rule to your Apache configuration file.

Note: *This scheme only provides simple random load-balancing. It does not provide fail-over nor does it provide any heuristics or any other ability to distribute work based on server load.*

Figure 304 Load Balancing Example



11.6.1 Configuring the Apache Server

The main configuration file for Apache, **httpd.conf**, is located in *APACHE*/config. To enable **mod_proxy**, **mod_proxy_http**, and **mod_rewrite**, uncomment the following lines shown in bold text (these options are usually disabled—commented out—by default).

```
#LoadModule mime_magic_module modules/mod_mime_magic.so
LoadModule proxy_module modules/mod_proxy.so
#LoadModule proxy_connect_module modules/mod_proxy_connect.so
LoadModule proxy_http_module modules/mod_proxy_http.so
#LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule setenvif_module modules/mod_setenvif.so
```

11.6.2 Creating the Mapping File

Before you create the Web Services and the mapping file you need to be aware of the mapping rule that will be used. The URL generated for the SeeBeyond Web Services endpoint is of the following form (you must rename the ***bold_italic*** section in the servlet context property as described in **Web Services External System** on page 353):

```
WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_bpCalc1/STCService
```

The mapping file (**ws_server.txt**.) uses the business process name as the key to find a list of URL endpoints on which that service will be available. This file is a simple text file of the following form:

```
Key alternate_URL1|alternate_URL2
```

The content below is a typical example—however, it should actually appear on a single line with no space between the alternate URLs. The alternate URLs contain the actual endpoints for the Web Services, including the host IP address and port.

```
bpCalc1
     127.0.0.1:18004/WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_bpCalc1/STCService|
     127.0.0.1:19004/WSServlet_WSCalc_u02F_IS2_WSC_WSCALC_bpCalc1/STCService
```

The mapping file must be stored on the same system that is running Apache, and is usually found in the configuration directory, for example:

```
C:\Program Files\IBM HTTP Server 2.0\conf\ws_server.txt.
```

11.6.3 Adding the Rewrite Rule

The code section below is the rewrite rule; lines beginning with '**#**' are comments:

```
# Start of ICAN rewrite rules
<IfModule mod_rewrite.c>
#RewriteLog "C:\Program Files\IBM HTTP Server 2.0\logs\rewrite.log"
#RewriteLogLevel 5
RewriteMap SERVICES "rnd:C:\Program Files\IBM HTTP Server
2.0\conf\ws_server.txt"

<Location ~ /WSServlet_ >
    RewriteEngine On

    RewriteRule "([^/_]+(?=/STCService))" "http://${SERVICES:$1}" [P,L]
    RewriteRule "(.*)" "http://localhost:80/notfound.html" [G]
</Location>
</IfModule>
# End of rewrite directives.
```

The purpose of this rule is to intercept requests for ICAN-formatted URLs of the form

```
http://LOADBALANCER:9999/WSServlet_XXXXXXX_bpCalc1/STCService
```

and rewrite them to a form using the entries in the mapping file **ws_server.txt**. It does this by extracting the key (shown in bold) from the requested URL. This key is the last element of the variable part of the endpoint URL created by ICAN.

- If it cannot find a suitable match, it will return a *gone* (410) error.
- If it finds a suitable key but does not find a match in the mapping file, it will return a *not found* (404) error.
- If a key is matched but the service is not available on that URL, it will return a *bad gateway* (502) error.

11.6.4 Debugging

There are a number of ways to debug the internal processing of Apache, the rewrite logging probably being the most useful. You can specify an additional rewrite log and a log level within the rewrite section of the **httpd.conf** file. The logging level is on a scale of 0 to 5, with 0=off and 5=maximum debugging.

An example of a successful rewrite request log is shown below, using log level 5 (timestamps omitted):

```
[rid#596d60/initial] (3) [per-dir /WSServlet_/] add path info postfix: C:/
    Program Files/IBM HTTP Server 2.0/htdocs/en_US/WSServlet_WSCalc_
    u02F_IS1_WSC_WSCALC_bpCalc1 -> C:/Program Files/IBM HTTP Server 2.0/
    htdocs/en_US/WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_bpCalc1/STCService

[rid#596d60/initial] (3) [per-dir /WSServlet_/] applying pattern '([^/_]+(?=/
    STCService))' to uri 'C:/Program Files/IBM HTTP Server 2.0/htdocs/en_US/
    WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_bpCalc1/STCService'

[rid#596d60/initial] (5) cache lookup OK: map=SERVICES[txt] key=bpCalc1 ->
    val=127.0.0.1:18004/WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_bpCalc1/
    STCService

[rid#596d60/initial] (5) randomly choosen the subvalue `127.0.0.1:18004/
    WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_bpCalc1/STCService'

[rid#596d60/initial] (2) [per-dir /WSServlet_/] rewrite C:/Program Files/IBM
    HTTP Server 2.0/htdocs/en_US/WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_
    bpCalc1/STCService -> http://127.0.0.1:18004/WSServlet_WSCalc_u002F_
    IS1_WSC_WSCALC_bpCalc1/STCService

[rid#596d60/initial] (2) [per-dir /WSServlet_/] forcing proxy-throughput with
    http://127.0.0.1:18004/WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_bpCalc1/
    STCService

[rid#596d60/initial] (1) [per-dir /WSServlet_/] go-ahead with proxy request
    proxy:http://127.0.0.1:18004/WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_
    bpCalc1/STCService [OK]
```

11.7 Building a Web Client

Here we briefly demonstrate the procedure for building a Web client. The steps involved are:

1. Build an Object Type Definition (OTD).
2. Develop a business process.
3. Create the eGate Project.
4. Deploy the Project to the selected Environment.

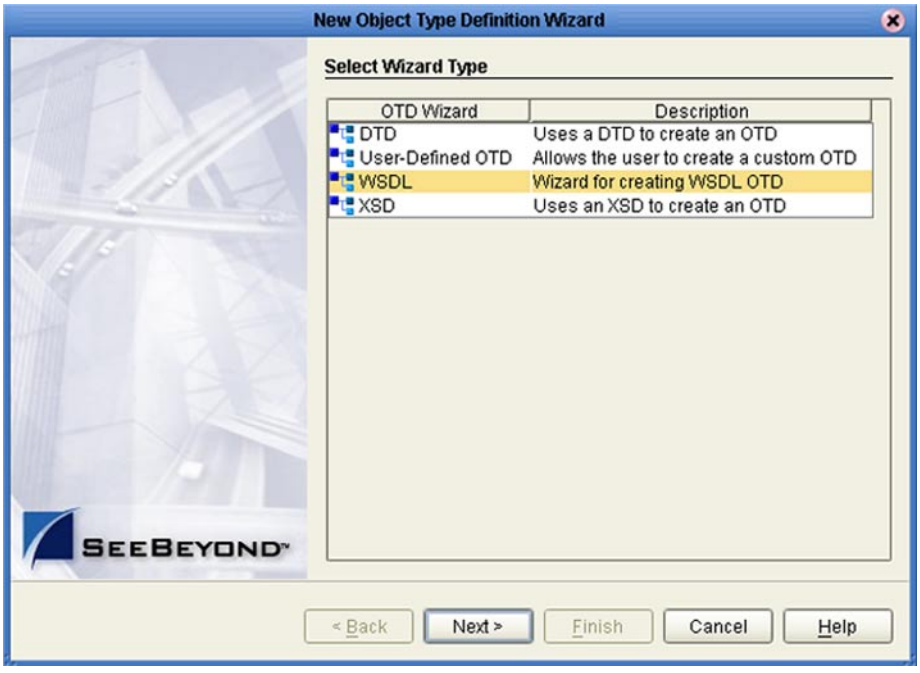
The Project used in the following example is contained in the *eGate User's Guide Sample* listed on the Enterprise Manager's Documentation page. To use this example Project, download the sample file **eGate_User_Guide_Sample.zip** and extract the contents to a convenient directory. Import the file **webclient.zip** into your Repository following the procedure described in **Project/Environment Import** on page 72. The files **input.txt** and **output.txt** are simple text files for testing purposes.

11.7.1 Object Type Definition

To create a Web Client OTD

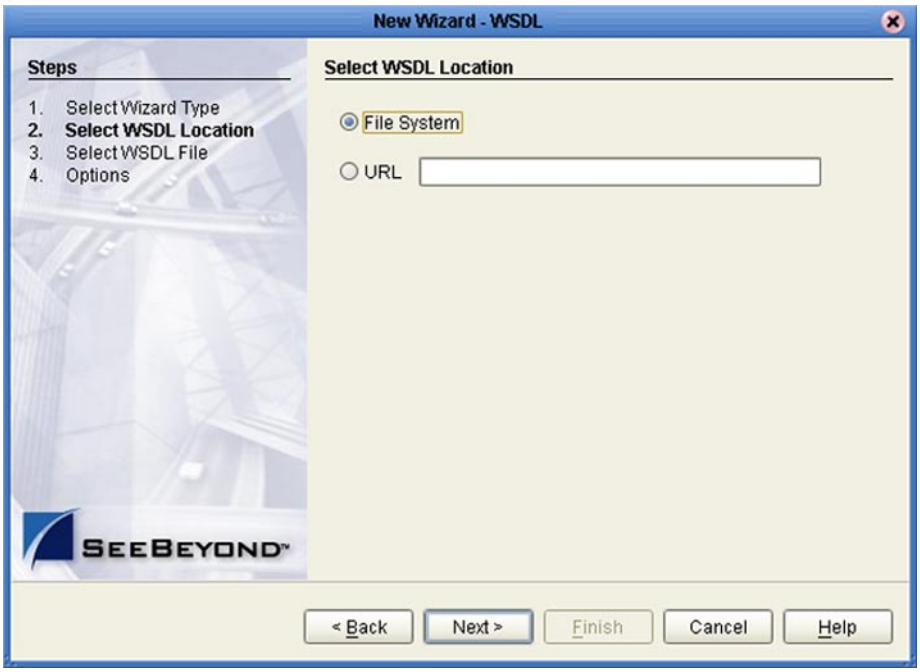
1. Select **New > Object Type Definition** from your Project's context menu.
2. Select **WSDL** from the initial Wizard dialog (see Figure 305) and click **Next**.

Figure 305 Select WSDL Wizard



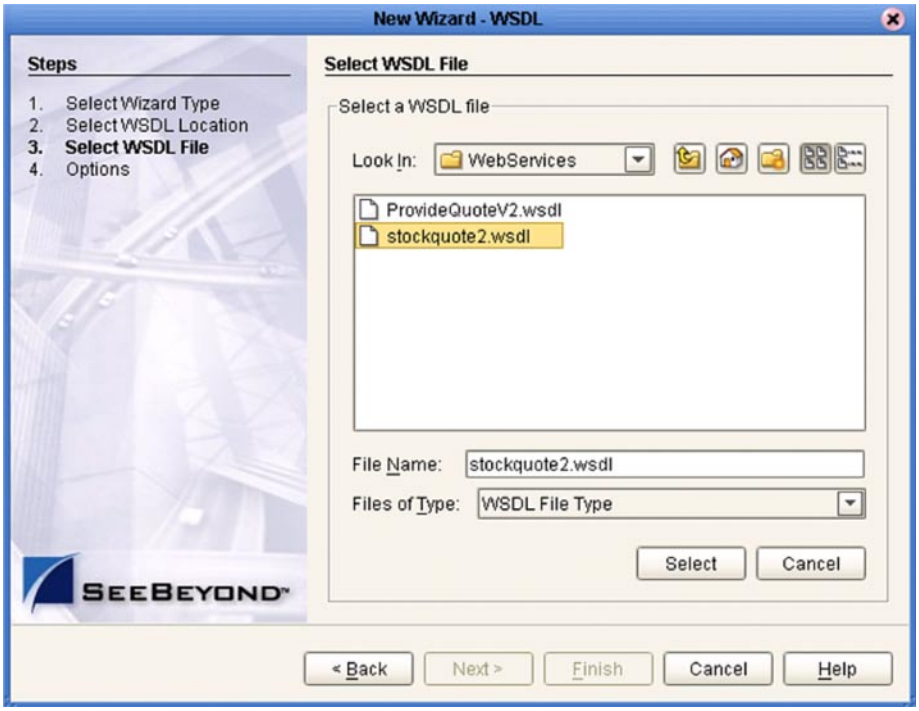
3. Select the WSDL file location; in this example, the local file system (see Figure 306).
4. Click **Next**.

Figure 306 Select File Location



5 Select the WSDL file on which you want to base the OTD (see Figure 307) and click **Next**.

Figure 307 Select WSDL File

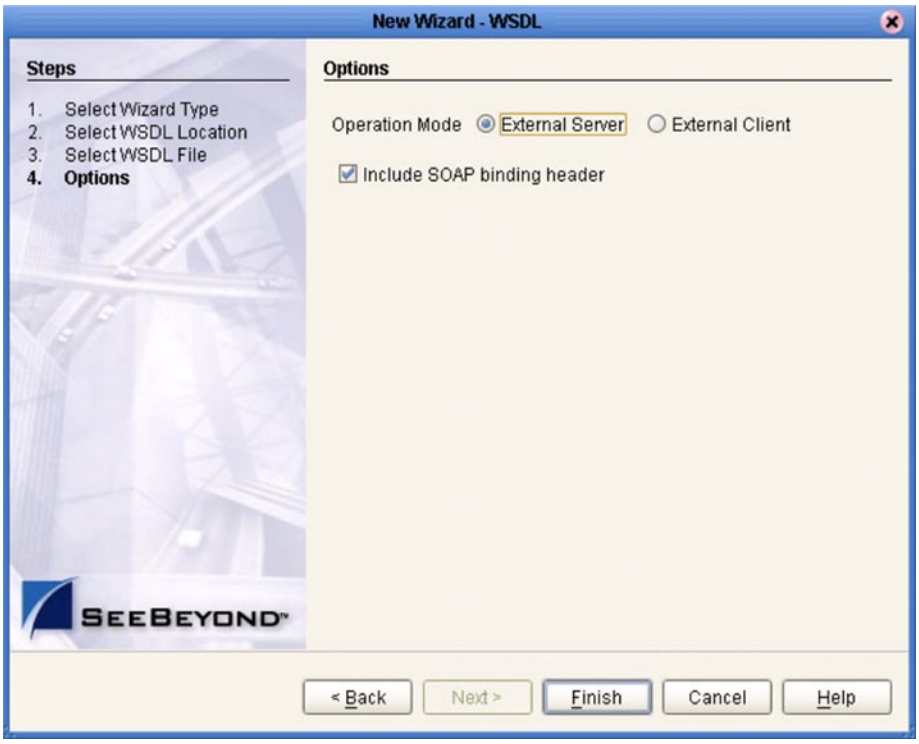


6 For a Web Client, select the following options (see Figure 308):

A Select **External Server** as the Operation Mode.

B Select **Include SOAP binding header** .

Figure 308 Select External Server



7 Click **Finish**.

11.7.2 eInsight Business Process

The example business process, developed in eInsight Business Process Manager, is shown in Figure 309 (see the *eInsight Business Process Manager User's Guide* for details).

Figure 309 Web Client Business Process

The **receive** rule for the business process is shown in Figure 310, and the **write** rule is shown in Figure 311.

Figure 310 Web Client Business Process *Receive* Rule

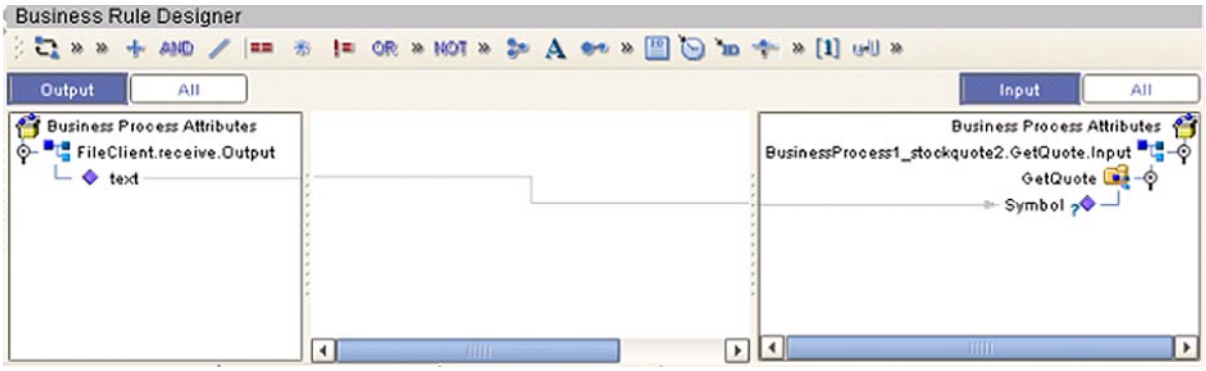
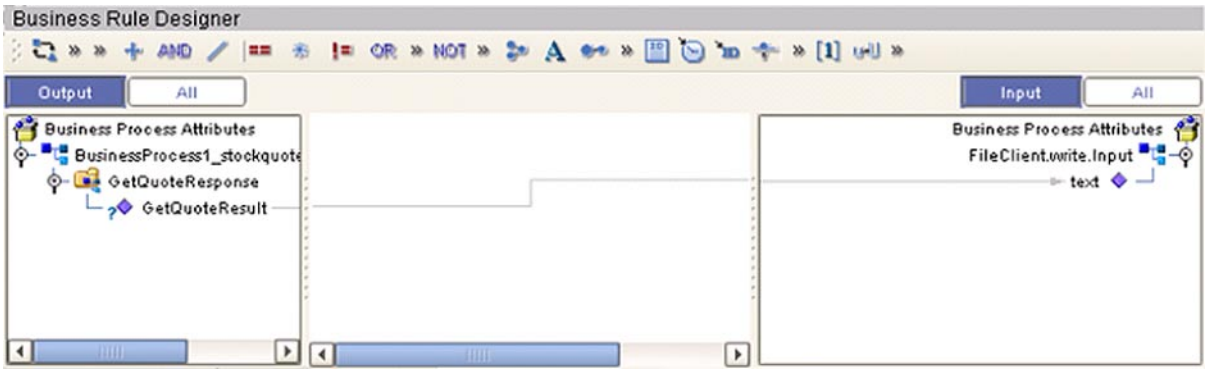


Figure 311 Web Client Business Process *Write* Rule



The WSDL file describing the business process is shown in Figure 312.

Figure 312 Sample WSDL File

```
<process name="BusinessProcess1"
         targetNamespace="http://127.0.0.1:12000/repository/webclient/BusinessProcess1"
         sbynpxp:end_YLoc="123.0"
         sbynpxp:start_YLoc="120.0"
         sbynpxp:linkStyle="angular"
         sbynpxp:start_XLoc="50.0"
         sbynpxp:end_XLoc="508.0"
         xmlns:tns="http://127.0.0.1:12000/repository/webclient/BusinessProcess1"
         xmlns:sbynpx="http://bpel.seebeyond.com/hawaii/5.0/privateExtension/"
         xmlns:slink="ServiceLinkTypes/SeeBeyond/eInsight/e3273l:f8eaf3f6cf:-7fff"
         xmlns:ns0="http://www.webserviceX.NET/"
         xmlns:sbynruntime="http://bpel.seebeyond.com/hawaii/5.0/privateExtension/runtime/"
         xmlns:sbyncreation="http://bpel.seebeyond.com/hawaii/5.0/privateExtension/creation"
         xmlns:nsl="urn:fileservice"
         xmlns:sbynpxp="http://bpel.seebeyond.com/hawaii/5.0/privateExtension/presentation/"
         xmlns="http://schemas.xmlsoap.org/ws/2002/07/business-process/"
         xmlns:sbyntracing="http://bpel.seebeyond.com/hawaii/5.0/privateExtension/tracing/"
         xmlns:sbyninc="http://bpel.seebeyond.com/hawaii/5.0/privateExtension/incompleteModel">
    <!-- partners definition -->
    <partners>
        <partner name="BusinessProcess1_stockquote2"
```

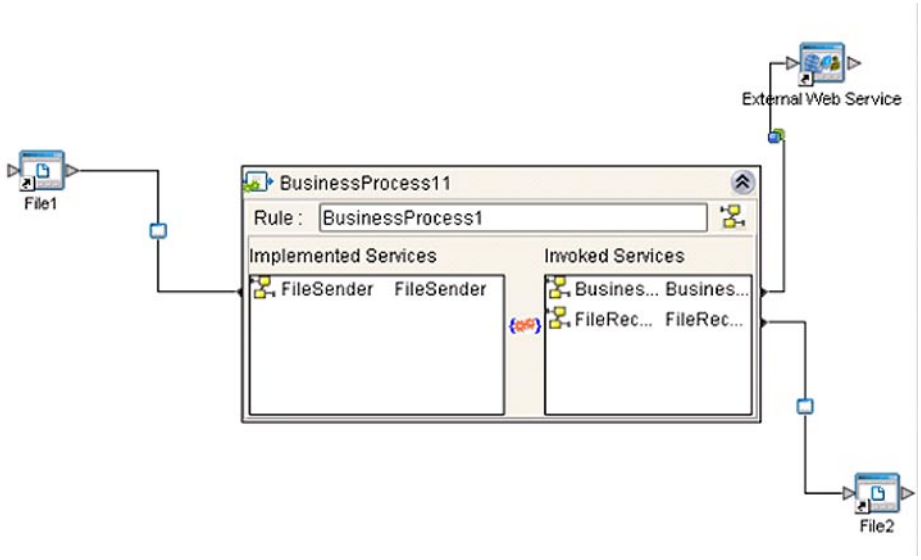
11.7.3 eGate Project

The Project components are created and mapped in the Enterprise Designer Connectivity Map Editor. The example Project contains:

- Two external files and accompanying File eWays.
- An External Web Service.
- A service, into which you drag and drop the eInsight business process from the Project Explorer.

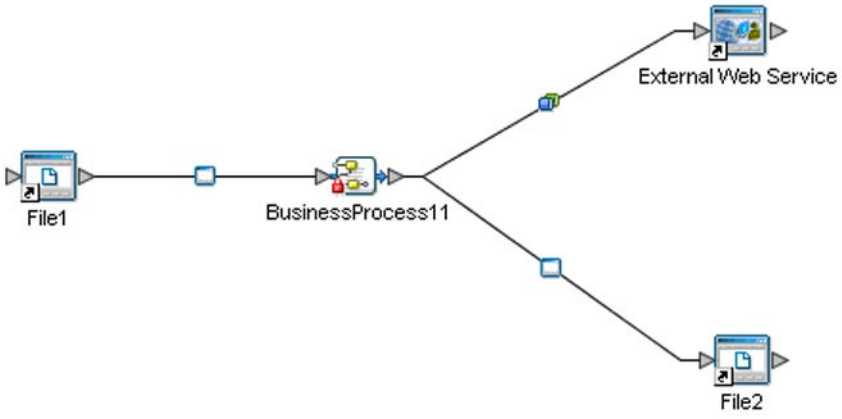
The business process is connected as shown in Figure 313.

Figure 313 Map Business Process



The completed Connectivity Map for the example Project is shown in Figure 314.

Figure 314 Web Client Connectivity Map



The Web client example Project appears in the Project Explorer as shown in Figure 315.

Figure 315 Web Client Example Project

The example Project is deployed to a run-time Environment as shown in Figure 316.and Figure 317

Figure 316 Web Client Deployment (1)

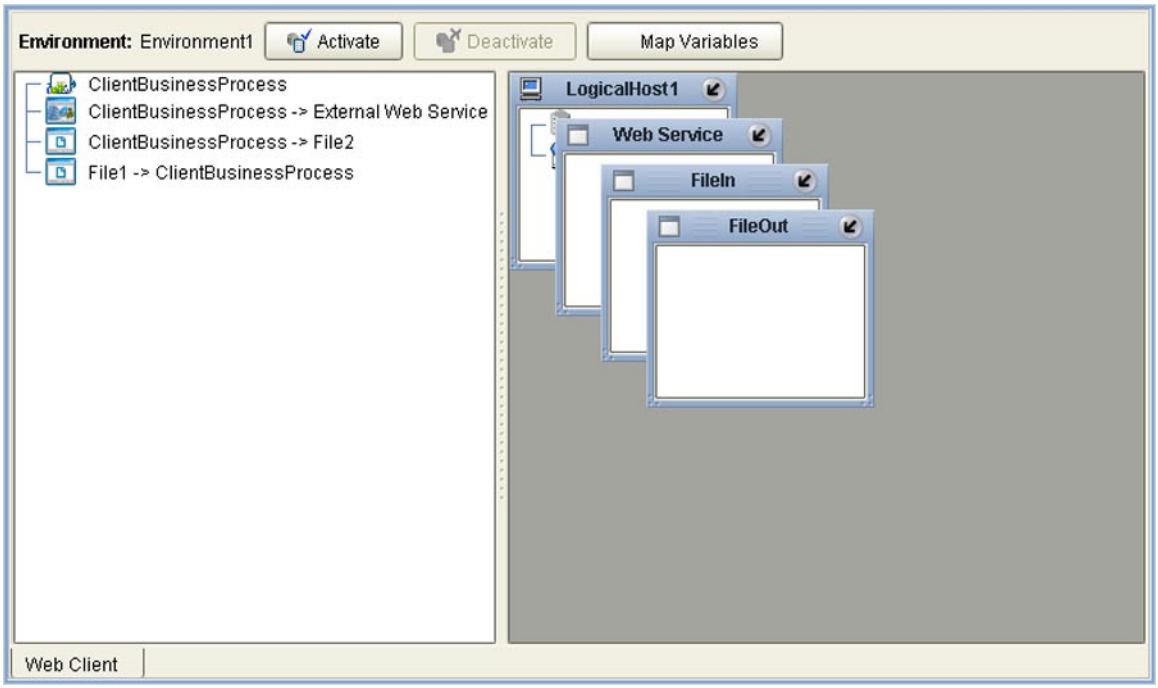
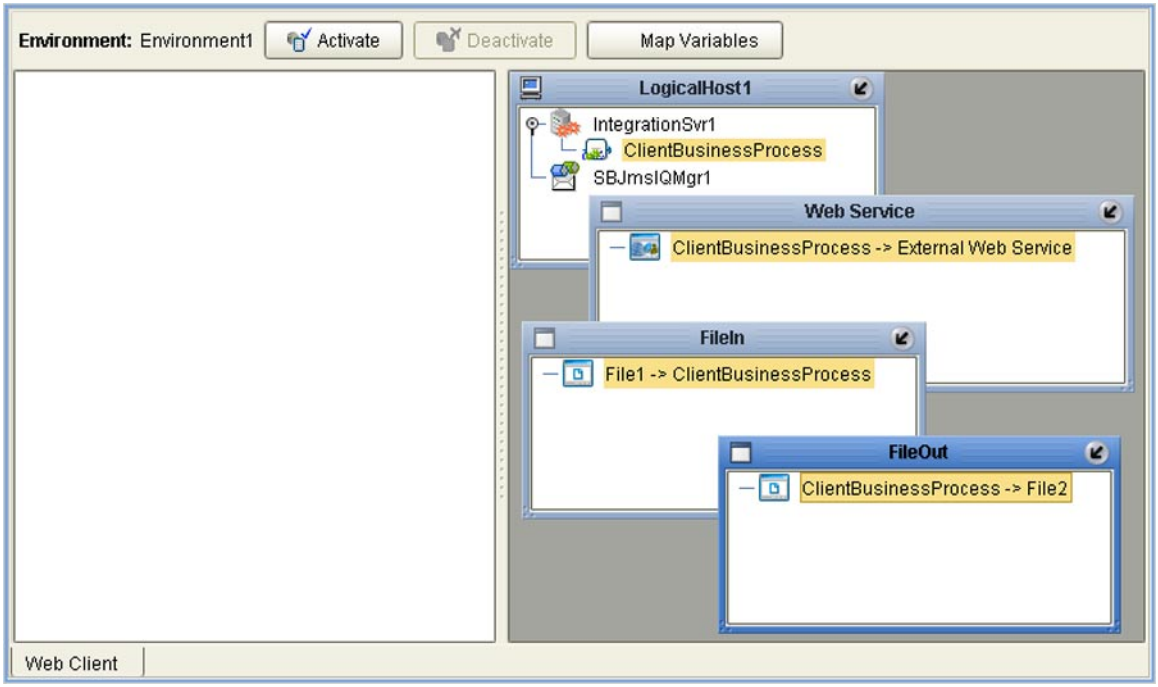


Figure 317 Web Client Deployment (2)



11.8 Building a Web Server

Here we briefly demonstrate the procedure for building a Web server. As with the Web client, the steps involved are:

1. Build an Object Type Definition (OTD).
2. Develop a business process.
3. Create the eGate Project.
4. Deploy the Project to the selected Environment.

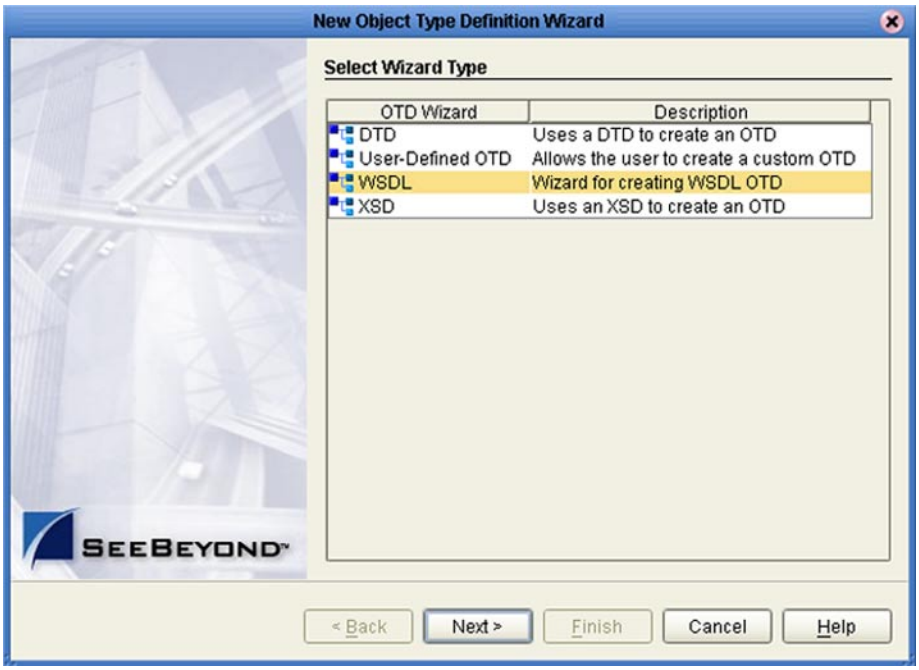
The Project used in the following example is contained in the *eGate User's Guide Sample* listed on the Enterprise Manager's Documentation page. To use this example Project, download the sample file **eGate_User_Guide_Sample.zip** and extract the contents to a convenient directory. Import the file **webserver.zip** into your Repository following the procedure described in **Project/Environment Import** on page 72.

11.8.1 Object Type Definition

To create a Web Server OTD

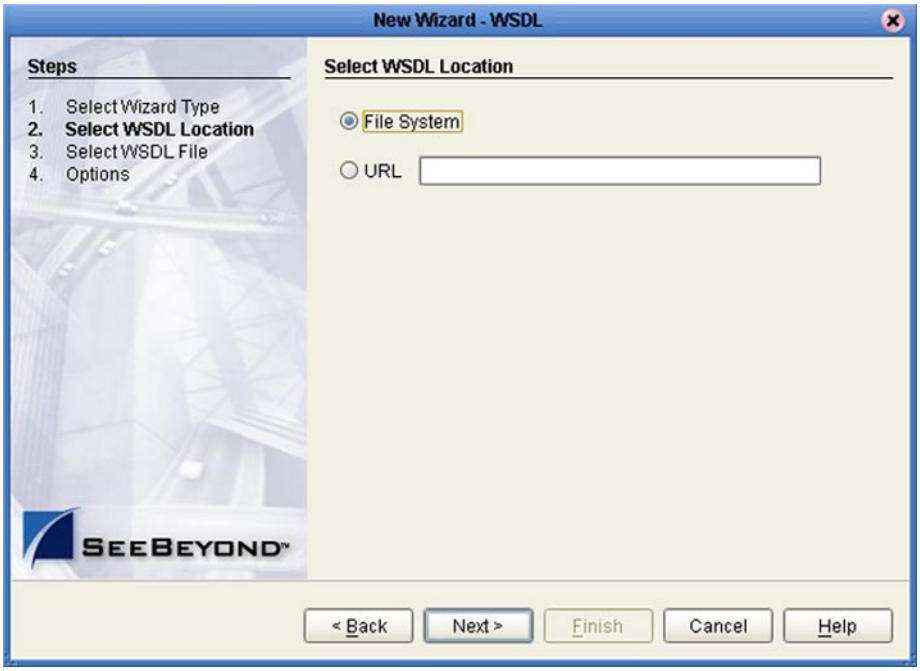
1. Select **New > Object Type Definition** from your Project's context menu.
2. Select **WSDL** from the initial Wizard dialog (see Figure 305) and click **Next**.

Figure 318 Select WSDL Wizard



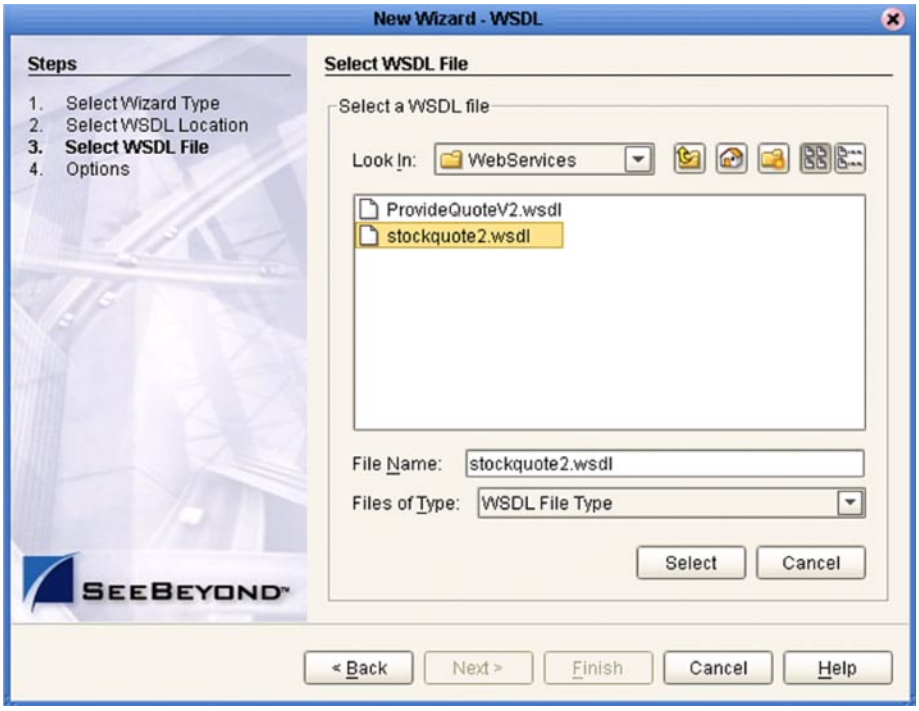
3 Select the WSDL file location; in this example, the local file system (see Figure 319) and click **Next**.

Figure 319 Select File Location



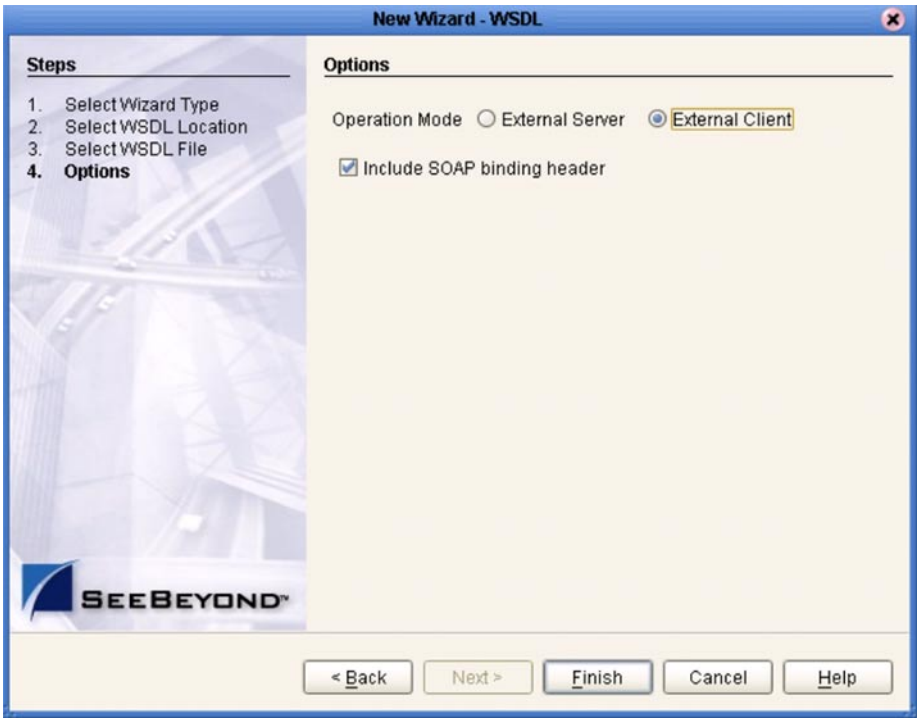
4 Select the WSDL file you want to use for the OTD (see Figure 320) and click **Next**.

Figure 320 Select WSDL File



5 For a Web server, select External Client (see Figure 321) and click **Finish**.

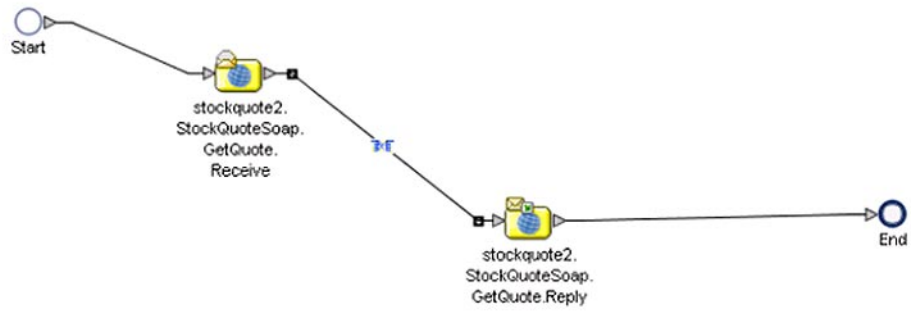
Figure 321 Select External Client



11.8.2 eInsight Business Process

The example business process, developed in eInsight Business Process Manager, is shown in Figure 322 (see the *eInsight Business Process Manager User's Guide* for details).

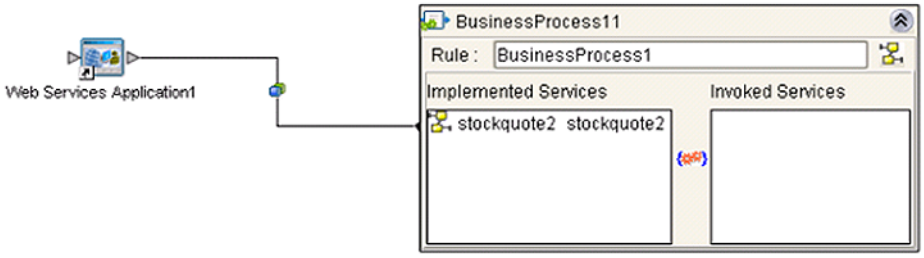
Figure 322 Web Server Business Process



11.8.3 eGate Project

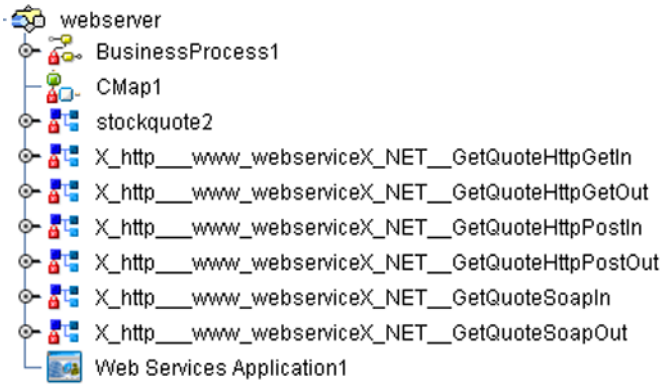
The business process is connected as shown in Figure 323, using the Enterprise Designer Connectivity Map Editor.

Figure 323 Connectivity Map



The Web server example Project appears in the Project Explorer as shown in Figure 324.

Figure 324 Web Server Example Project



The Project is deployed to a run-time Environment as shown in Figure 325 and Figure 326.

Figure 325 Web Server Deployment (1)

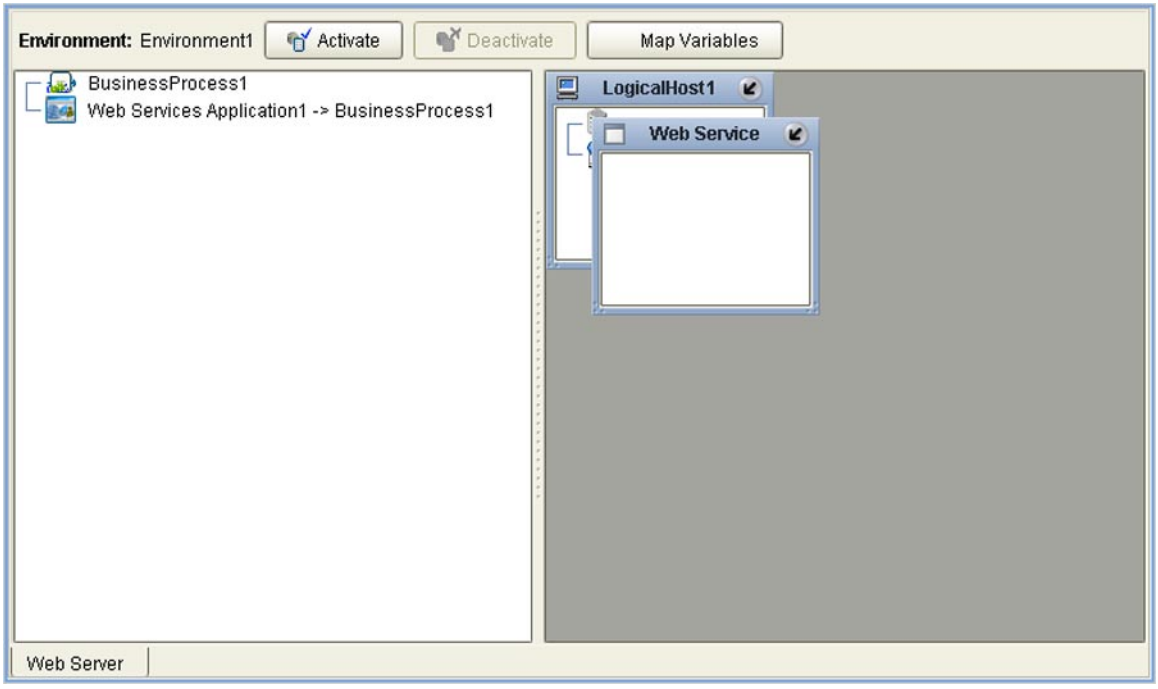
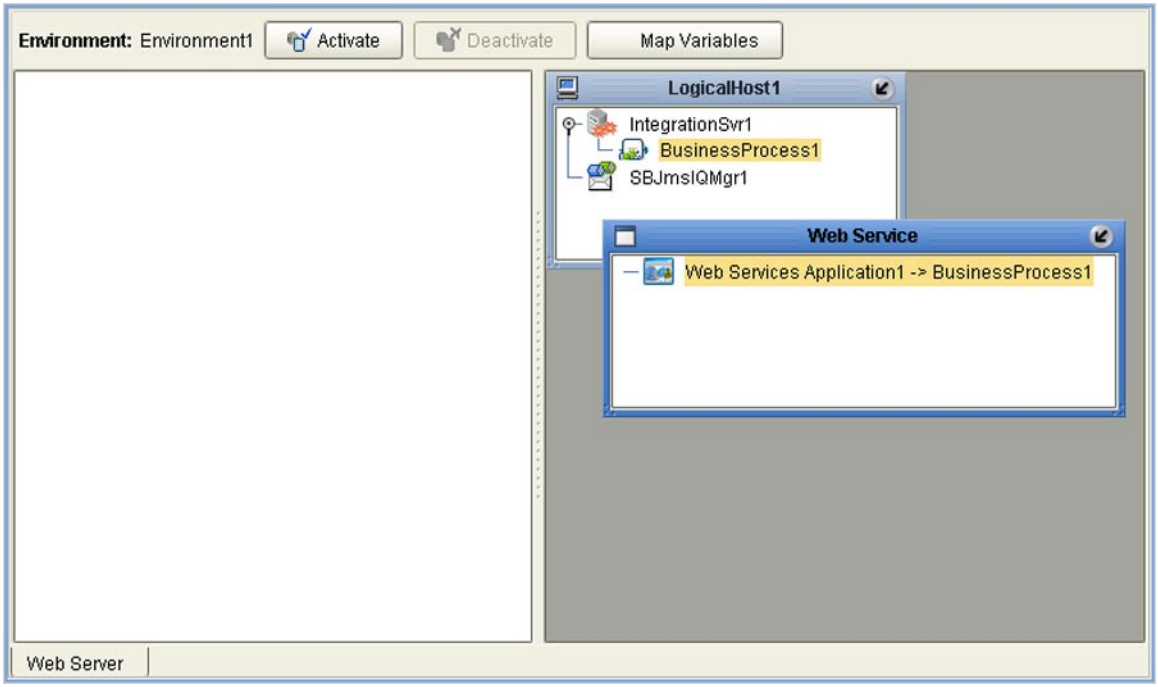


Figure 326 Web Server Deployment (2)



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 **Collaboration**s, **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**.

CRM
Acronym for Customer Relations Management.

Data Cleansing
Refers to transforming data for accurate and effective use in a database or data management system by cleansing “dirty” or redundant data. Data must be cleansed of errors in structure and content before it is useful in data warehousing and integration.

Data Dictionary
Defines the organization of a database and lists all files in the database, the number of records in each file, and the names and types of each field. Although the dictionary

does not contain actual data, it does contain essential information for managing the database. The data dictionary is often hidden from end users

Data Integrity
Refers to the accuracy and validity of data. Data integrity can be compromised in many ways, including human error through data entry, or through faulty logic in programming. Computer viruses, software bugs and many other factors can also compromise data integrity.

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 Transformation
Data transformation is necessary after extracting data from legacy data formats, or any format that requires **Data Cleansing**. Data is transformed for efficient use for Business-to-Business Enterprise Data Integration.

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.

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 **Project** components. An Environment contains Logical Hosts and external systems.

ERM
Acronym for Enterprise Resource Management.

ETL
A three-phase (extract/transform/load) process used, for example, to generate and maintain a central **Metadata** repository.

- **Extract** is the process of reading data from a source database and obtaining the desired subset of data.
- **Transform** is the process of converting the extracted data from its previous form into the desired form.
- **Load** is the process of writing the transformed data into the target database.

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 ICAN Suite.

External System
A representation of a computer system hosting an application external to the ICAN Suite.

Extraction
The process of reading data from a source database and obtaining the desired subset of data (see **ETL**).

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.

ICAN Suite
The SeeBeyond Integrated Composite Application Network Suite.

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-compliant 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 contains the eGate run-time components, including integration servers and message servers, that are installed on a host hardware platform.

Management Agent
Uses J2EE technology to manage and monitor an eGate 5.0 deployment that may contain other application servers in addition to the SeeBeyond Integration Server. Defines management interfaces and services designed for distributed environments, focusing on providing functionality for managing networks, systems, and applications.

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 Map**s.

Metadata
Metadata describes the structure and format of a particular set of data. **ETL** tools are used to generate and maintain a central metadata repository.

Non-normalized Data
Data that has not been converted to normalized data (see next entry).

Normalized Data
Data that has been processed to remove redundant or incorrect data structure and organization, thereby creating a maintainable data set that can be cross-referenced. Normalized data is not only easier to analyze but also easier to expand.

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 system in SeeBeyond's **Enterprise Designer**.

Query
A request for information from a database. There are three basic query methods:

- **Choose** – With this simplified method, the database system presents a list of parameters from which you can choose. This method is the least flexible of the three methods.
- **Query by example (QBE)** – With this method, the system lets you specify fields and values to define a query.
- **Query language** – With this method, you have the ability to make requests for information in the form of a stylized query using a query language. This is the most complex and powerful of the three methods.

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 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**.

Raw Data

Unprocessed data, as obtained from the source (see also **Non-normalized Data**).

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 **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 5.0 that allows **Collaboration**s developed in e*Gate 4.x to be used in and controlled from eGate 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.

SI/SO

Acronym for Shift-In/Shift-Out.

SOAP

SOAP (Simple Object Access Protocol) enables a program running in one operating system to communicate with another program running in either the same or a different operating system, using **HTTP** and **XML** as the mechanisms for information exchange.

Subproject

An independent **Project** that is included as part of another Project, and is displayed in the Enterprise Explorer tree as a "branch" beneath the main Project.

Table

Refers to data arranged in rows and columns, as in a spreadsheet. In **Relational Database (RDBMS)** systems, all information is stored in tables.

Topic

A JMS topic is a shareable object that conforms to the *publish-and-subscribe* (pub/sub) messaging domain, where one publisher broadcasts messages to one or more subscribers. When the SeeBeyond **JMS IQ Manager** publishes a message on a topic, it ensures that all subscribers receive the message.

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 (see **Normalized Data**), and other sub-processes.

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.

eGate 5.0 Terminology

Table 100 lists terminology that is new with eGate release 5.0 along with equivalent terms from eGate release 4.x, where applicable.

Table 100 Terminology Cross-Reference

eGate 5.0 Term	Equivalent e*Gate 4.x Term
Connection	e*Way Connection
Connectivity Map	Schema Network View (closest)
Deployment	Running the Control Broker
Deployment Profile	<none> (part of Schema)
Enterprise Designer	Enterprise Manager
Enterprise Manager	Enterprise Monitor
Environment	Schema (physical layer only)
eWay	e*Way, e*Way Connection
eWay Configuration	e*Way Connection Configuration
External Application	e*Way Connection
External System	e*Way Connection
JMS Connection	e*Way Connection
ICAN Monitor	Enterprise Monitor
Integration Server	<none>
Link	JMS e*Way Connection
Linked Message Destination	<none>
Logical Host	Participating Host
Message Destination	Topic or queue
Message Server	MS IQ Manager
Object Type Definition (OTD)	Event Type Definition (ETD)
Process Manager	Control Broker
Project	Schema (logical layer only)
Queue	MS queue
Repository	Registry
Subproject	Schema (logical layer only)
Topic	JMS topic

Index

A

ACL properties **97–99**, **101**, **103**, **295–298**
activating
  Deployment Profile **337**
addition Collaboration method **197**, **269**
Advanced Mode command **180**
AND Collaboration method **193**, **269**
arrayAccess Collaboration method **205**
arrayAssign Collaboration method **206**
arrayLength Collaboration method **206**
Auto Layout command **191**
Automerge **210**, **288**

B

BEA WebLogic **341**
bitNot operator **207**
boolean Collaboration method **277**
BPEL
  definition **369**
Break command **183**
business rules
  tree **183**
Business Rules on Left command **180**
Business Rules on Top command **180**

C

Call Java Method command **190**
Call New Constructor command **190**
cast Collaboration method **199**
ceiling Collaboration method **275**
charAt Collaboration method **202**
check in **85**
  without revisions **86**
check out **87–88**
Collaboration
  definition **369**
  derived **370**
Collaboration Definition
  definition **369**
Collaboration definition
  Java **168**
  XSLT **259**
Collaboration method box
  Java **192**
  XSLT **268**
Collaboration method palette
  Java **192**
  XSLT **268**
Collapse All Methods command **191**
Commit Changes command **184**
Commit Code Changes command **267**
component version
  checked in **83**
  checked out **83**
  history **84**
  latest version **90**
  retrieved **83**
components
  copying and pasting **100–101**
  cutting and pasting **100–101**
concat Collaboration method **202**, **272**
concat-sequence-format Collaboration method **284**
connection **369**
Connectivity Map
  definition **369**
  Editor **104**
constants **369**
  Environmental **299**
contains Collaboration method **277**
Continue command **183**
Control Broker **376**
conventions
  path name separator **25**
  Windows **25**
  writing **25**
copying and pasting components **100–101**
count Collaboration method **279**
Create Literal command **189**
CRM **369**
current Collaboration method **279**
currentDateTime Collaboration method **284**
customizer **111**
cutting and pasting components **100–101**

D

data
  cleansing **369**
  dictionary **369**
  integrity **370**
  mapping **370**
  non-normalized **372**
  normalized **372**
  raw **373**
  transformation **370**
Data Type Definition (DTD) **126**

database, relational **373**
deactivating
  Deployment Profile **337**
debugger
  Collaboration (Java) **227**
Deployment Editor **333**
Deployment Profile **332**
  activating **337**
  creating **334**
  deactivating **337**
  definition **370**
  map variables **340**
derived Collaboration **370**
diff file
  create **209**, **288**
  generate **209**, **288**
  merge **210**, **289**
division Collaboration method **197**, **269**
document Collaboration method **279**
DTD
  definition **370**
  Wizard **126**

E

Editor
  Connectivity Map **104**
  Deployment **333**
  Java Collaboration **179**
  OTD **118**
  XSLT Collaboration **266**
element-available Collaboration method **277**
Encoding OTD command **120**
endsWith Collaboration method **202**
Enterprise Designer
  definition **370**
  enterprise explorer **64**
  menu bar **60**
  starting **58**
Enterprise Explorer
  Environments **65**, **291**
  Projects **64**, **93**, **291**
Enterprise Manager
  Documentation **39**
  Interface **37**
  starting **36**
Enterprise Monitor **40**, **376**
Enterprise Service Bus (ESB)
  definition **370**
Environment **31**
  constants **299**
  definition **370**
Environment Explorer **65**, **291**
EPR **370**
equal Collaboration method **194**, **270**
equals Collaboration method **200**
ETD **376**
ETL **371**
Event Type Definition **376**
eWay
  definition **371**
Expand All Methods command **191**
Export Java Rule command **180**
external application **107**
  definition **371**
external system **31**
  definition **371**
extraction **371**

F

false Collaboration method **278**
Field command **182**
Find command **185**
Find Next command **185**
Find Previous command **185**
floor Collaboration method **275**
For command **182**
format_number Collaboration method **272**
formatDate Collaboration method **284**
formatDateTime Collaboration method **284**
formatMessage Collaboration method **282**
function_available Collaboration method **278**

G

generate-id Collaboration method **280**
getAppID Collaboration method **283**
getAppValue Collaboration method **283**
getCenturyFromDate Collaboration method **284**
getCenturyFromDateTime Collaboration method **285**
getCommonID Collaboration method **283**
getCommonValue Collaboration method **283**
getDayFromDate Collaboration method **285**
getDayFromDateTime Collaboration method **285**
getHourFromDateTime Collaboration method **285**
getHourFromTime Collaboration method **286**
getMinutesFromDateTime Collaboration method **286**
getMinutesFromTime Collaboration method **286**
getMonthFromDate Collaboration method **285**
getMonthFromDateTime Collaboration method **285**
getSecondsFromDateTime Collaboration method **286**
getSecondsFromTime Collaboration method **286**
getTimezoneFromDate Collaboration method **286**
getTimezoneFromDateTime Collaboration method

286
getTimezoneFromTime Collaboration method **286**
getYearFromDate Collaboration method **285**
getYearFromDateTime Collaboration method **285**
greater_or_equal Collaboration method **194**, **270**
greater_than Collaboration method **194**, **270**

H

HTTP
  definition **371**

I

IBM WebSphere **344**
ICAN Suite **371**
Icons
  JCE Business Rules Designer toolbar **189**
  JCE Business Rules toolbar **182**, **184**
  JCE toolbar **180**
  XCE toolbar **267**
id Collaboration method **280**
if Collaboration method **196**
If-Then command **182**
Impact Analyzer
  definition **371**
  usage **81**
Import a Local File command **180**
Import JAR File command **180**
Import Static Field command **189**
Import XSL from a Local File command **267**
indexOf Collaboration method **203**
instanceOf Collaboration method **200**
Integration Server
  definition **371**
Interfaces
  Enterprise Manager **37**

J

Java Collaboration
  Debugger **227**
  definitions **168**
  Editor (JCE) **179**
  method box **192**
  method palette **192**
  Wizard **169**
Java Collaboration methods
  addition **197**
  AND **193**
  arrayAccess **205**
  arrayAssign **206**
  arrayLength **206**
  cast **199**
  charAt **202**
  concat **202**
  division **197**
  endsWith **202**
  equal **194**
  equals **200**
  greater_or_equal **194**
  greater_than **194**
  if **196**
  indexOf **203**
  instanceOf **200**
  length **203**
  lesser_or_equal **195**
  lesser_than **195**
  multiplication **197**
  NOT **193**
  not_equal **195**
  OR **193**
  remainder **198**
  replace **203**
  return **196**
  substring **203**
  subtraction **198**
  throw **196**
  toLowerCase **203**
  toString **201**
  toUpperCase **203**
  trim **204**
  while **196**
Java Collaboration operators
  bitNot **207**
  negative **207**
  positive **207**
  postDecrement **207**
  postIncrement **208**
  preDecrement **208**
  preIncrement **208**
JCE commands
  Advanced Mode **180**
  Auto Layout **191**
  Break **183**
  Business Rules on Left **180**
  Business Rules on Top **180**
  Call Java Method **190**
  Call New Constructor **190**
  Collapse All Methods **191**
  Commit Changes **184**
  Continue **183**
  Create Literal **189**
  Expand All Methods **191**
  Export Java Rule **180**
  Field **182**
  Find **185**

Find Next **185**
Find Previous **185**
For **182**
If-Then **182**
Import a Local File **180**
Import JAR File **180**
Import Static Field **189**
Local Variable **182**
Method **182**
New Array **191**
Redo **185**
Refresh Collaboration **180**
Replace **185**
Return **182**
Roll Back Changes **185**
Rule **182**
Source Code Mode **180**
Standard Mode **180**
Throw **182**
Try **182**
Undo **185**
Validate **180**
While **182**
JCE Tester **186**
JMS IQ Manager
definition **371**

K

key Collaboration method **280**

L

language Collaboration method **278**
last Collaboration method **280**
length Collaboration method **203**
lesser_or_equal Collaboration method **195**, **270**
lesser_than Collaboration method **195**, **270**
link **372**
load balancing
Web services **354**
Local Variable command **182**
local-name Collaboration method **280**
Logical Host **31**
definition **372**

M

Management Agent **372**
map variables **340**
menu bar **60**
message destination
definition **372**
linked **372**
metadata **372**
method box
Java Collaboration **192**
XSLT Collaboration **268**
Method command **182**
method palette
Java Collaboration **192**
XSLT Collaboration **268**
Monitor
Enterprise **40**
multiplication Collaboration method **197**, **270**

N

name Collaboration method **280**
namespace-uri Collaboration method **280**
negative Collaboration method **271**
negative operator **207**
New Array command **191**
normalize_space Collaboration method **272**
NOT Collaboration method **193**, **278**
not_equal Collaboration method **195**, **271**
number Collaboration method **275**
number-literal Collaboration method **276**

O

Object Type Definition **372**
wizard **127**, **134**, **137**
Object Type Definition (OTD) **114**
Open File command **120**
OR Collaboration method **193**, **271**
OTD **372**
Editor **118**
tester **119**
OTD Editor commands
Open File **120**
Refresh OTD **120**
Run Tester **120**
Save as New Name **119**
Save File **120**
Specify Encoding **120**
Tester **119**
Toggle Reference Tab Panel **119**
OTD Wizard
DTD **126**
User-Defined **142**
WSDL **132**
XSD **136**

P

palette
    Java Collaboration methods **192**
    XSLT Collaboration methods **268**
parseDate Collaboration method **287**
parseDateTime Collaboration method **287**
Participating Host **376**
position Collaboration method **281**
positive operator **207**
postDecrement operator **207**
postIncrement operator **208**
preDecrement operator **208**
preIncrement operator **208**
Profile, Deployment **332**
Project
    definition **372**
Project Explorer **64**, **93**, **291**
proxy server **56**

Q

query **372**
queue
    definition **373**

R

rdbms **373**
Redo command **185**
Refresh Collaboration command **180**
Refresh OTD command **120**
Registry **376**
relational database **373**
remainder Collaboration method **198**, **271**
replace Collaboration method **203**
Replace command **185**
Repository **31**, **376**
    definition **373**
reset() method **126**, **136**, **142**
return Collaboration method **196**
Return command **182**
Roll Back Changes command **185**
Roll Back Code Changes command **267**
round Collaboration method **276**
Rule command **182**
Run Tester command **120**

S

Save as New Name command **119**
Save File command **120**
Save XSL to a Local File command **267**
Scheduler **108**
Schema **376**
Schema Runtime Environment (SRE) **41**
    definition **373**
security **305**
service
    definition **373**
servlet context property **353**
setCommonID Collaboration method **284**
Show Mapping Only command **267**
Show Maps and Code command **267**
Show XSLT Code Only command **267**
SOAP **347**
    definition **374**
Source Code Mode command **180**
SRE Monitor **41**
Standard Mode command **180**
starting
    Enterprise Designer **58**
    Enterprise Manager **36**
starts_with Collaboration method **278**
stateless session beans, deploying to IS **318**
string Collaboration method **273**
string_length Collaboration method **273**
string_literal Collaboration method **273**
subproject
    definition **374**
substring Collaboration method **203**, **273**
substring-after Collaboration method **273**
substring-before Collaboration method **273**
subtraction Collaboration method **198**, **271**
sum Collaboration method **276**
supporting documents **25**
system-property Collaboration method **274**

T

table **374**
terminology **376**
tester
    Collaboration (Java) **227**
    OTD **119**
    XSLT **266**
Tester command **119**
throw Collaboration method **196**
Throw command **182**
Toggle Reference Tab Panel command **119**
toLowerCase Collaboration method **203**
topic
    definition **374**
toString Collaboration method **201**
toUpperCase Collaboration method **203**
transformation **374**
translate Collaboration method **274**
trim Collaboration method **204**

true Collaboration method **278**
Try command **182**

U

UANExtension **268**
UANExtension method
  concat-sequence-format **284**
  currentDateTime **284**
  formatDate **284**
  formatDateTime **284**
  formatMessage **282**
  getAppID **283**
  getAppValue **283**
  getCenturyFromDate **284**
  getCenturyFromDateTime **285**
  getCommonID **283**
  getCommonValue **283**
  getDayFromDate **285**
  getDayFromDateTime **285**
  getHourFromDateTime **285**
  getHourFromTime **286**
  getMinutesFromDateTime **286**
  getMinutesFromTime **286**
  getMonthFromDate **285**
  getMonthFromDateTime **285**
  getSecondsFromDateTime **286**
  getSecondsFromTime **286**
  getTimezoneFromDate **286**
  getTimezoneFromDateTime **286**
  getTimezoneFromTime **286**
  getYearFromDate **285**
  getYearFromDateTime **285**
  parseDate **287**
  parseDateTime **287**
  setCommonID **284**
UDDI **347**
  definition **374**
  registry **349**
undo check-out **86**
Undo command **185**
unparsed-entity-uri Collaboration method **274**
User-Defined OTD Wizard **142**

V

Validate command **180**
variables
  mapping **340**
version control **83**
  check in **83**, **85**
  check out **83**, **87–88**
  command-line utilities **91**
  create diff **209**, **288**
  definition **374**
  generate diff **209**, **288**
  history **84**
  latest version **90**
  merge diff **210**, **289**
  retrieve **83**
  undo check out **86**

W

WebLogic **341**
WebSphere **344**
while Collaboration method **196**
While command **182**
Wizard
  Java Collaboration **169**
  OTD (DTD) **126**
  OTD (User-Defined) **142**
  OTD (WSDL) **132**
  OTD (XSD) **136**
  XSLT Collaboration **260**
WSDL **347**
  definition **374**
  Wizard **132**

X

XCE commands
  Commit Code Changes **267**
  Import XSL from a Local File **267**
  Roll Back Code Changes **267**
  Save XSL to a Local File **267**
  Show Mapping Only **267**
  Show Maps and Code **267**
  Show XSLT Code Only **267**
XML **347**
  definition **374**
XSD
  definition **374**
  Wizard **136**
XSLT
  definition **375**
XSLT Code Tester **266**
XSLT Collaboration
  definitions **259**
  Editor **266**
  method box **268**
  method palette **268**
  Wizard **260**
XSLT Collaboration methods
  addition **269**
  AND **269**
  boolean **277**
  ceiling **275**

concat **272**
concat-sequence-format **284**
contains **277**
count **279**
current **279**
currentDateTime **284**
division **269**
document **279**
element-available **277**
equal **270**
false **278**
floor **275**
format_number **272**
formatDate **284**
formatDateTime **284**
formatMessage **282**
function_available **278**
generate-id **280**
getAppID **283**
getAppValue **283**
getCenturyFromDate **284**
getCenturyFromDateTime **285**
getCommonID **283**
getCommonValue **283**
getDayFromDate **285**
getDayFromDateTime **285**
getHourFromDateTime **285**
getHourFromTime **286**
getMinutesFromDateTime **286**
getMinutesFromTime **286**
getMonthFromDate **285**
getMonthFromDateTime **285**
getSecondsFromDateTime **286**
getSecondsFromTime **286**
getTimezoneFromDate **286**
getTimezoneFromDateTime **286**
getTimezoneFromTime **286**
getYearFromDate **285**
getYearFromDateTime **285**
greater_or_equal **270**
greater_than **270**
id **280**
key **280**
language **278**
last **280**
lesser_or_equal **270**
lesser_than **270**
local-name **280**
multiplication **270**
name **280**
namespace-uri **280**
negative **271**
normalize_space **272**
NOT **278**
not_equal **271**
number **275**
number-literal **276**
OR **271**
parseDate **287**
parseDateTime **287**
position **281**
remainder **271**
round **276**
setCommonID **284**
starts_with **278**
string **273**
string_length **273**
string_literal **273**
substring **273**
substring-after **273**
substring-before **273**
subtraction **271**
sum **276**
system-property **274**
translate **274**
true **278**
unparsed-entity-uri **274**