

Oracle® Fusion Middleware

Oracle Enterprise Gateway Configuration Guide
11g Release 1 (11.1.1.5.0)

June 2011

The Oracle logo, consisting of the word "ORACLE" in a bold, red, sans-serif font. The letter "O" is stylized with a small registered trademark symbol (®) to its upper right.

Oracle Enterprise Gateway Configuration Guide, 11g Release 1 (11.1.1.5.0)

Copyright © 1999, 2011, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services. This documentation is in prerelease status and is intended for demonstration and preliminary use only. It may not be specific to the hardware on which you are using the software. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to this documentation and will not be responsible for any loss, costs, or damages incurred due to the use of this documentation.

The information contained in this document is for informational sharing purposes only and should be considered in your capacity as a customer advisory board member or pursuant to your beta trial agreement only. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle Software License and Service Agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

Contents

1. Install Guides	
1. System Requirements	26
Overview	26
Operating System Requirements	26
Specific Requirements	27
Default Ports	27
2. Installing the Enterprise Gateway on Windows	29
Prerequisites	29
Install Executable	29
Introductory screen	29
Choose Install Location	29
Installing	29
Installation Complete	29
Start the Enterprise Gateway	29
Start the Policy Studio	30
3. Installing the Enterprise Gateway on Linux	31
Prerequisites	31
Install the Enterprise Gateway	31
Start the Enterprise Gateway	31
Start the Policy Studio	31
4. Installing the Enterprise Gateway on Solaris	33
Prerequisites	33
Install the Enterprise Gateway	33
Start the Enterprise Gateway	33
Start the Policy Studio	33
5. Installing Policy Studio on Windows	35
Prerequisites	35
Download the Policy Studio	35
Start the Enterprise Gateway	35
Start the Policy Studio	35
6. Installing Policy Studio on Linux	37
Prerequisites	37
Download the Policy Studio	37
Start the Enterprise Gateway	37
Start the Policy Studio	37
7. Installing Policy Studio on Solaris	39
Prerequisites	39
Download the Policy Studio	39
Start the Enterprise Gateway	39
Start the Policy Studio	39
8. Upgrading a Previous Enterprise Gateway Installation	41
Overview	41
Prerequisites	41
Running the Upgrade Script	41
Updating your Enterprise Gateway Configuration	42
9. License Acknowledgments	44
Overview	44
Acknowledgments	44
10. Oracle Support	45
Contact Details	45
2. Getting Started	
1. Oracle Enterprise Gateway Overview	46

Overview	46
Performance	46
Governance	46
Integration	46
Security	47
2. Oracle Enterprise Gateway Architecture	48
Overview	48
Product Architecture	48
Product Concepts	50
3. Starting the Enterprise Gateway	56
Overview	56
Before you Begin	56
Starting the Enterprise Gateway	56
Accessing the Enterprise Gateway Tools	56
Starting Policy Studio	57
Getting Started Tutorial	57
4. Registering a Web Service	58
Overview	58
Accessing the Axis Web Services	58
Registering the Example Web Service	58
Accessing the Protected Web Service	59
Testing the Protected Web Service	59
Monitoring the Web Service	61
5. Monitoring a Web Service	62
Overview	62
Enabling Monitoring	62
Viewing Real-time Monitoring	62
Viewing Message Monitoring	63
Viewing Message Content	63
Detecting Malformed Messages	64
Monitoring a Remote Host	65
Using Service Monitor	66
Securing the Web Service	66
6. Securing a Web Service	67
Overview	67
Creating a Security Policy	67
Applying a Security Policy	68
Adding a User	68
Testing a Security Policy	68
Chaining Policies Together	70
Monitoring Traffic	73
7. Monitoring Traffic	74
Overview	74
Enabling Traffic Monitor	74
Viewing the Message Traffic Log	74
Viewing Transaction Details	75
Troubleshooting	77
8. Troubleshooting	78
Overview	78
Viewing Enterprise Gateway Trace Files	78
Setting Enterprise Gateway Trace Levels	78
Configuring Enterprise Gateway Trace Files	79
Running Trace at DEBUG level	80
Running Trace at DATA level	82
Integrating Trace Output with Apache log4J	84
Configuring Logging Output	84
Configuring Log Level and Message	84
Getting Help	85

3. Service Manager	
1. Getting Started with Service Manager	86
Introduction	86
Logging in to Service Manager	86
Logging out of Service Manager	86
Deploying to the Enterprise Gateway	87
Discarding from Service Manager	87
2. Managing Web Services	88
Overview	88
Web Service Groups	88
Registering a Web Service	88
Accessing a Virtualized Web Service	89
Assigning Policies to a Web Service	89
Removing an Assigned Policy	90
Editing a Web Service	90
Deleting a Web Service or Group	90
3. Managing Policies	91
Overview	91
Policy Containers	91
Adding a Composite Policy	91
Removing a Sub-Policy	92
Editing a Policy	92
Deleting a Policy or Container	92
4. Governance	
1. Configuring Security Policies from WSDL Files	93
Overview	93
Importing a WSDL file	93
Configuring Policy Settings	94
Configuring Policy Filters	94
Editing a Policy	96
Removing Security Tokens	97
Further Information	98
2. Securing a Virtual Service using Policies	99
Overview	99
Importing a WSDL File	99
Configuring a Security Policy	100
Configuring Policy Settings	101
Configuring Policy Filters	101
Editing a Security Policy	102
Using WCF WS-Policies	103
Removing Security Tokens	104
Further Information	105
3. Configuring Policies Manually	107
Overview	107
Configuration	107
4. Configuring Global Policies	109
Overview	109
Global Policy Roles	109
Selecting a Global Policy	110
Configuring Global Policies in a Policy Shortcut Chain	111
Configuring Global Policies for a Service	112
Showing Global Policies	113
5. Managing Deployments	
1. Introduction to Policy Management	114
Overview	114
Policy Management Features	114
2. Getting Started with Managing Deployments	115

Overview	115
Connecting to a Server	115
Editing an Active Server Configuration	116
Managing Processes	116
Managing Deployed Configuration	116
Managing Configuration Versions	116
Editing Configuration Profiles	117
Comparing and Merging Configurations	117
Managing Policy Studio Users	117
Configuring Policies	117
3. Managing Processes	118
Overview	118
Viewing Process Details	118
Adding a New Process	118
Removing a Process	119
Changing a Process Owner	119
Editing Connection Details	119
Versioning Process Configuration	119
Deploying Configuration to a Process	120
4. Managing Configuration Profiles	121
Overview	121
Managing Deployed Configuration Profiles	121
Deploying Configuration Profiles to Multiple Processes	121
Managing All Configuration Profiles	122
Managing My Configuration Profiles	122
Managing a Component Store	123
Managing a Configuration Version	124
5. Managing Policy Studio Users	126
Overview	126
Super User Privileges	126
Adding a New User	126
Removing a User	127
Resetting a User Password	127
Managing User Roles	127
6. Comparing and Merging Configurations	128
Overview	128
Compare and Merge Example	128
Viewing Differences	128
Swapping the Source and Target	130
Filtering Differences	130
Selecting Differences for Merging	130
7. Migrating from a Staging to a Production Environment	131
Overview	131
Accessing Environments	131
Configuration that Differs between Environments	131
Naming Configuration Entities	131
Initial Transfer from Source to Target Environment	132
Configuration that may need to be Modified	133
Comparison of Source and Target after Initial Transfer	134
Merging Updates from Source to Target Environment	135
8. Managing the Audit Trail	137
Overview	137
Setting up the Audit Trail	137
Configuration Audit Trail	137
Process Deployment Audit Trail	138
User Audit Trail	138
9. Configuring Role-Based Access Control	139
Overview	139

Server acl.policy File	139
Enterprise Gateway Welcome Page	140
Protect Management and Policy Director Interfaces Policy	140
Policy Center User Store	141
System Administrators Role	142
PD Superuser Privilege	143
Management Service Roles and URIs	144
6. General Configuration	150
1. Startup Instructions	150
Overview	150
Setting Passphrases	150
Start in Console Mode	150
Start in Service Mode	151
2. Connection Details	153
Overview	153
Connecting to a Server URL	153
Connecting to a Server Configuration File	154
Unlocking a Server Connection	154
3. Policy Studio Preferences	155
Overview	155
Management Services	155
Policy Colors	155
Proxy Settings	155
Runtime Dependencies	156
Server Connection	156
Trace Level	156
Web and XML	156
WS-I Settings	157
4. Global Configuration	159
Overview	159
Enterprise Gateway Configuration Options	159
Web Services Repository	159
Processes	160
Policies	160
Certificates	160
Enterprise Gateway User Store	161
System Alerts	161
External Connections	161
Schema Cache	162
Black list and White list	162
Caches	162
5. Enterprise Gateway Configuration Options	163
Overview	163
Deploy	163
Default Settings	163
Default Logging Settings	163
MIME/DIME Settings	163
Namespace Settings	163
Change Passphrase	163
6. Web Service Repository	164
Overview	164
Testing WS-I Compliance	164
Registering the WSDL File	164
Loading the WSDL File	165
Selecting WSDL Operations	165
WSDL Import Settings	165
Deploy Policy	165

Secure Virtual Service	166
WSDL Import Summary	166
What is Created?	166
Publishing the WSDL	167
7. Setting the Encryption Passphrase	169
Encryption Passphrase Overview	169
Setting the Passphrase for the First Time	169
Connecting to the Enterprise Gateway with Policy Studio	169
Connecting to Enterprise Gateway Configuration Data	170
Changing the Passphrase	170
8. Retrieving WSDL Files from a UDDI Registry	172
Overview	172
UDDI: A Brief Introduction	172
UDDI Definitions	172
Configuring a Registry Connection	174
WSDL Search	175
Quick Search	175
Name Search	176
Advanced Search	176
Advanced Options	178
Publish	180
9. Connecting to a UDDI Registry	181
Overview	181
Configuring a Registry Connection	181
Securing a Connection to a UDDI Registry	182
10. Publishing WSDL Files to a UDDI Registry	184
Overview	184
Finding WSDL Files	184
Publishing WSDL Files	184
Step 1: Enter Virtualized Service Address and WSDL URL for Publishing in UDDI Registry	184
Step 2: View WSDL to UDDI Mapping Result	185
Step 3: Select a Registry for Publishing	186
Step 4: Select a Duplicate Publishing Approach	186
Step 5: Create or Search for Business	187
Step 6: Publish WSDL	188
11. Default Settings	189
Overview	189
Settings	189
12. Namespace Settings	192
Overview	192
Signature ID Attribute	192
WSSE Namespace	193
SOAP Namespace	193
13. Find Filter Dialog	194
Overview	194
Configuration	194
7. Reporting	195
1. Service Monitor Installation Guide	195
Overview	195
Prerequisites	195
Extracting Service Monitor	195
Setting up the Database	196
Configuring the Service Monitor Database	196
Configuring the Service Monitor TCP Port	198
Configuring the Enterprise Gateway	198
Launching Service Monitor	200
Further Information	200

2. Service Monitor User Guide	201
Overview	201
Topology Overview	201
Reports	202
Real-Time Monitoring	203
Audit Trail	207
3. Scheduled Reports	208
Overview	208
Database Configuration	208
Scheduled Report Configuration	208
Advanced Options	209
Configuring a Time Schedule	209
8. Processes	210
1. Configuring Processes	210
Overview	210
Add HTTP Services	210
Add SMTP Services	210
Add Policy Execution Scheduler	210
Messaging System	210
Directory Scanner	210
POP Client	210
Add Remote Host	211
TIBCO	211
Process Settings	211
Process Logging	211
Monitoring	211
Cryptographic Acceleration	211
Tivoli	211
Kerberos	211
Oracle Security Service Module Settings	212
Audit Trail	212
2. Configuring HTTP Services	213
Overview	213
HTTP Services Groups	213
HTTP and HTTPS Interfaces	214
HTTPS Interfaces Only	216
Relative Paths	217
Web Service Resolvers	219
Static Content Provider	220
Servlet Applications	221
Packet Sniffers	222
Management Services	223
Changing the Management Services Port	224
3. Configuring SMTP Services	226
Overview	226
Adding an SMTP Service	226
Adding an SMTP Interface	227
Configuring Circuit Handlers for SMTP Commands	227
Adding an HELO/EHLO Circuit Handler	228
Adding an AUTH Circuit Handler	228
Adding a MAIL Circuit Handler	229
Adding a RCPT Circuit Handler	230
Adding a DATA Circuit Handler	231
SMTP Authentication	231
SMTP Content-Transfer-Encoding	232
Deployment Example	232
4. Policy Execution Scheduling	237

Overview	237
Cron Expressions	237
Adding a Schedule	239
Adding a Policy Execution Scheduler	239
5. Directory Scanner	240
Overview	240
Directory to Scan	240
Directory for Output	240
Completed Directory	241
Working Directory	241
Policy to Use	241
6. Packet Sniffers	242
Overview	242
Configuration	242
7. Messaging System	244
Overview	244
Configuring a JMS Service	244
Configuring a JMS Session	245
Configuring a JMS Consumer	245
Configuring the JMS Wizard	246
8. Remote Host Settings	247
Overview	247
General Settings	247
Address Settings	248
Advanced Settings	248
Configuring Watchdogs	249
9. Configuring an HTTP Watchdog	251
Overview	251
Configuration	251
10. Configuring Conditions for HTTP Interfaces	252
Overview	252
Requires Endpoint Condition	252
Requires Link Condition	253
11. POP Client	254
Overview	254
Configuration	254
12. TIBCO Integration	255
Overview	255
TIBCO Rendezvous Integration	255
TIBCO Enterprise Messaging System Integration	255
13. Real-Time Monitoring Settings	257
Overview	257
Configuring Monitoring Settings	257
Configuring Service Monitor Settings	257
14. Configuring Traffic Monitoring	259
Overview	259
Configuration	259
15. Cryptographic Acceleration	261
Overview	261
General Configuration	261
Conversations for Crypto Engines	262
16. Cryptographic Acceleration - Conversation - Request:Response	263
Conversations for Crypto Engines	263
17. TIBCO Rendezvous Daemon	264
Overview	264
Configuration	264
18. TIBCO Rendezvous Listener	266
Overview	266

Configuration	266
19. TIBCO Enterprise Messaging System Consumer	267
Overview	267
Configuration	267
20. Oracle Security Service Module Settings	269
Overview	269
Settings	269
Name Authority Definition	269
Further Information	270
9. Resources	271
1. Certificate Store	271
Overview	271
Configuring a Certificate and Private Key	271
X.509 Certificate	271
Private Key	271
Global Options	272
2. Enterprise Gateway Users	273
Overview	273
Users	273
Adding Users	273
Attributes	273
Groups	274
Adding Groups	274
Updating Users or Groups	274
3. System Alerting	275
Overview	275
Configuring an Alert Destination	275
Configuring an Alert Filter	279
4. Global Schema Cache	281
Overview	281
Adding Schemas to the Cache	281
Testing WSDL Files for WSI Compliance	281
Organizing Schemas with Schema Containers	282
Schema Validation	283
5. External Connections	284
Overview	284
Authentication Repository Profiles	284
Connection Sets	284
Database Connections	285
Kerberos Connections	285
LDAP Connections	285
OCSP Connections	286
Proxy Servers	286
RADIUS Clients	287
SiteMinder	287
SMTP Servers	287
SOA Security Manager	287
Syslog Servers	287
TIBCO	288
Tivoli	288
URL Connection Sets	288
XKMS Connections	289
6. Global Caches	290
Overview	290
Local Caches	290
Distributed Caches	291
Cache Settings	292

Example of Caching Response Messages	293
10. Attributes	
1. Retrieve Attribute from Database	296
Overview	296
General Configuration	296
Database	296
Advanced	296
2. Retrieve Attributes from Directory Server	297
Overview	297
General Configuration	297
Retrieve Unique User Identity	297
Retrieve Attributes	297
3. Retrieve Attribute from HTTP Header	299
Overview	299
Configuration	299
4. Retrieve Attribute from SAML Attribute Assertion	300
Overview	300
Details	300
Trusted Issuers	301
Subject Configuration	301
Lookup Attributes	302
5. SAML PDP Attributes	303
Overview	303
Request Configuration	303
Response Configuration	305
6. Retrieve Attribute from Message	306
Overview	306
Configuration	306
7. Retrieve Attribute from User Store	307
Overview	307
Configuration	307
8. Insert SAML Attribute Assertion	308
Overview	308
General Configuration	309
Assertion Details	309
Assertion Location	309
Subject Confirmation Method	310
Advanced	312
9. Extract REST Request Attributes	314
Overview	314
Configuration	314
10. Extract WSS Timestamp	316
Overview	316
Configuration	316
11. Extract WSS UsernameToken	317
Overview	317
Configuration	317
12. Extract WSS Header	318
Overview	318
Configuration	318
11. Authentication	
1. Authentication Repository	319
Overview	319
Local Repositories	319
LDAP Repositories	319
CA SiteMinder Repositories	322
Database Repositories	323

Entrust GetAccess Repositories	325
Oracle Access Manager Repositories	326
Oracle Entitlements Server Repositories	327
RADIUS Repositories	327
RSA Access Manager Repositories	327
Tivoli Repositories	328
2. HTML Form-based Authentication	329
Overview	329
Configuration	329
3. HTTP Basic Authentication	330
Overview	330
Configuration	330
4. HTTP Digest Authentication	332
Overview	332
Configuration	332
5. HTTP Header Validation	333
Overview	333
Configuration	333
6. IP Address	334
Overview	334
Configuration	334
Configuring Subnet Masks	334
7. SSL Authentication	337
Overview	337
Configuration	337
8. Attribute Authentication	338
Overview	338
Configuration	338
9. SAML Authentication	340
Overview	340
General Settings	341
Details	341
Trusted Issuers	341
10. SAML PDP Authentication	343
Overview	343
Request Configuration	343
Response Configuration	345
11. Insert SAML Authentication Assertion	346
Overview	346
General Configuration	347
Assertion Details	347
Assertion Location	347
Subject Confirmation Method	348
Advanced	350
12. WS-Security Username Authentication	352
Overview	352
General Configuration	352
Token Validation	353
Token Verification via Repository	353
13. Insert WS-Security Username Token	355
Overview	355
General Configuration	355
Credential Details	355
Advanced	356
14. Insert Timestamp	357
Overview	357
Configuration	357
15. Kerberos Client Authentication	358

Overview	358
Kerberos Client	358
Kerberos Token Profile	359
16. Kerberos Service Authentication	360
Overview	360
Kerberos Service	360
Kerberos Standard	360
Message Level	361
Transport Level	361
Advanced SPNEGO	361
17. Kerberos Configuration	362
Overview	362
Kerberos Configuration File - krb5.conf	362
Advanced Settings	362
Native GSS Library	362
18. Kerberos Clients	364
Overview	364
Ticket Granting Ticket Source	364
Kerberos Principal	365
Secret Key	365
Advanced Tab	366
19. Kerberos Services	368
Overview	368
Kerberos Endpoint Tab	368
Advanced Tab	369
20. Kerberos Principals	370
Overview	370
Configuration	370
21. Kerberos Keytab	372
Overview	372
Configuration	372
22. CA SOA Security Manager Authentication	374
Overview	374
Prerequisites	374
Agent Configuration	374
Message Details Configuration	375
XmlToolkit.properties File	375
23. RADIUS Clients	377
Overview	377
Configuration	377
12. Authorization	378
1. Attributes	378
Overview	378
Configuration	378
2. Certificate Attributes	380
Overview	380
Configuration	380
3. Check Group Membership	382
Overview	382
Configuration	382
Possible Paths	382
4. RSA Access Manager Authorization	383
Overview	383
Prerequisites	383
General Details	383
Connection Details	383
Authorization Details	384

5. Entrust GetAccess Authorization	386
Overview	386
GetAccess WS-Trust STS	386
GetAccess SAML PDP	386
6. Insert SAML Authorization Assertion	388
Overview	388
General Configuration	388
Assertion Details	389
Assertion Location	389
Subject Confirmation Method	390
Advanced	392
7. RBAC Filter	394
Overview	394
Configuration	394
8. SAML Authorization Assertion	395
Overview	395
General Settings	395
Details	395
Trusted Issuers	396
Optional Settings	396
9. SAML PDP Authorization	397
Overview	397
Request Configuration	397
Response	399
10. Tivoli Integration	400
Overview	400
Integration Architecture	400
Prerequisites	401
Global Tivoli Configuration	404
Tivoli Authorization	406
Tivoli Authentication	407
Tivoli Attribute Retrieval	408
11. Tivoli Authorization	409
Overview	409
Adding a Tivoli Client	409
Adding Users and Web Services to Tivoli	410
Configuring Tivoli Authorization	410
Tivoli Authentication Refresh	411
12. CA SOA Security Manager Authorization	412
Overview	412
Prerequisites	412
Configuration	412
13. XACML Policy Enforcement Point	413
Overview	413
Example XACML Request	414
General Settings	414
XACML Settings	414
Routing Settings	417
Advanced Settings	418
13. CA SiteMinder	
1. SiteMinder Certificate Authentication	419
Overview	419
Prerequisites	419
Configuration	419
2. SiteMinder Session Validation	421
Overview	421
Prerequisites	421

Configuration	421
3. SiteMinder Logout	423
Overview	423
Prerequisites	423
Configuration	423
4. SiteMinder Authorization	424
Overview	424
Prerequisites	424
Configuration	424
5. SiteMinder/SOA Security Manager Connection	425
Overview	425
SiteMinder and SOA Security Manager Connection Details	425
SOA Security Manager Connection Details Only	426
14. Certificates	427
1. Static CRL Certificate Validation	427
Overview	427
Configuration	427
2. Certificate CRL - Dynamic	428
Overview	428
Configuration	428
3. CRL LDAP Validation	429
Overview	429
Configuration	429
4. CRL Responder	430
Overview	430
Configuration	430
5. Create Thumbprint from Certificate	431
Overview	431
Configuration	431
6. Certificate Validity	432
Overview	432
Configuration	432
7. Find Certificate	433
Overview	433
Configuration	433
8. Extract Certificate Attributes	435
Overview	435
Generated Message Attributes	435
Configuration	437
9. Certificate Chain Check	438
Overview	438
Configuration	438
10. OCSP Certificate Validation	439
Overview	439
Configuration	439
11. Validate Certificates in Gateway's Store	440
Overview	440
Configuration	440
Deployment Example	440
12. XKMS Certificate Validation	443
Overview	443
Configuration	443
15. Cache	444
1. Cache Attribute	444
Overview	444
Configuration	444
2. Create Key	445

Overview	445
Configuration	445
3. Is Cached?	446
Overview	446
Configuration	446
4. Removed Cached Attribute	447
Overview	447
Configuration	447
16. Content Filtering	448
1. Content Validation	448
Overview	448
Manual XPath Configuration	448
XPath Wizard	449
2. Message Size	451
Overview	451
Configuration	451
3. Throttling	452
Overview	452
General Settings	452
Cache Settings	453
4. Content Type Filtering	454
Overview	454
Allow or Deny Types	454
Configuring MIME/DIME Types	454
5. HTTP Header Validation	455
Overview	455
Configuring HTTP Header Regular Expressions	455
Configuring Threatening Content Regular Expressions	456
6. Query String Validation	458
Overview	458
Request Query String	458
Configuring Query String Attribute Regular Expressions	459
Configuring Threatening Content Regular Expressions	460
7. Schema Validation	461
Overview	461
Schema to Use	461
Part of Message to Match	461
Advanced	462
Reporting Schema Validation Errors	464
8. Validate Message Attributes	467
Overview	467
Configuring Message Attribute Regular Expressions	467
Threatening Content Regular Expressions	468
9. Validate Timestamp	469
Overview	469
Configuration	469
10. Validate REST Request	471
Overview	471
General Configuration	471
REST Request Parameter Restrictions	471
11. XML Complexity	474
Overview	474
Configuration	474
12. Threatening Content	475
Overview	475
Scanning Details	475
MIME Types	475

13. WS-Security Policy Layout	476
Overview	476
Configuration	476
14. McAfee Virus Scanner	477
Overview	477
Prerequisites	477
Configuring a McAfee Anti-Virus Filter	477
Configuring Custom Options	478
Reporting Message Status	479
Loading McAfee Updates	479
15. ClamAV Anti-Virus	481
Overview	481
Configuration	481
16. Sophos Anti-Virus Filter	482
Overview	482
Prerequisites	482
General Settings	482
Sophos Configuration Settings	482
17. Conversion	484
1. Add HTTP Header	484
Overview	484
Configuration	484
2. Add XML Node	485
Overview	485
General Configuration	485
Configure where to Insert the New Nodes	485
Node Source	485
Configure New Node Details	486
Attribute Node Details	486
Examples	486
3. Contivo Transformation	488
Overview	488
Configuration	488
4. Multipart Bodypart Conversion	489
Overview	489
Configuration	489
5. Create REST Request	490
Overview	490
Configuration	490
6. Set HTTP Verb	491
Overview	491
Configuration	491
7. Insert MTOM Attachment	492
Overview	492
Configuration	493
8. Extract MTOM Attachment	494
Overview	494
Configuration	495
9. Load File	496
Overview	496
Configuration	496
10. Remove Attachments	497
Overview	497
Configuration	497
11. Remove HTTP Header	498
Overview	498
Configuration	498

12. Remove XML Node	499
Overview	499
Configuration	499
13. Set Message	500
Overview	500
Configuration	500
14. XSLT Transformation	502
Overview	502
Stylesheet Location	502
Stylesheet Parameters	502
Advanced	503
18. Encryption	
1. XML-Encryption Settings	504
Overview	504
XML Encryption Overview	504
Encryption Key	506
Key Info	507
Recipients	511
What to Encrypt	512
Advanced	513
Auto-generation using the XML Encryption Settings Wizard	514
2. XML Encryption Wizard	515
Overview	515
Configuration	515
3. XML-Encryption	516
Overview	516
Configuration	516
Auto-generation using the XML Encryption Settings Wizard	516
4. XML-Decryption Settings	517
Overview	517
XML Encryption Overview	517
Node(s) to Decrypt	519
Decryption Key	520
Options	520
Auto-generation using the XML Decryption Wizard	521
5. XML-Decryption	522
Overview	522
Configuration	522
Auto-generation using the XML Decryption Wizard	522
6. SMIME Encryption	523
Overview	523
General Configuration	523
Recipients	523
Advanced	523
7. SMIME Decryption	524
Overview	524
Configuration	524
19. Fault Handlers	
1. SOAP Fault	525
Overview	525
SOAP Fault Format	525
SOAP Fault Contents	525
Customized SOAP Faults	527
20. Integrity	
1. XML Signature Generation	529
Overview	529
Signing Key	529

What to Sign	534
Where to Place Signature	537
Additional	538
Algorithm Suite	540
Options	540
2. XML Signature Verification	544
Overview	544
Signature Verification	544
What Must Be Signed	544
Advanced	545
3. SMIME Sign	546
Overview	546
Configuration	546
4. SMIME Verify	547
Overview	547
Configuration	547
21. Monitoring	
1. Logging Configuration	548
Overview	548
Configuring Log Output	548
Log to Text File	548
Log to XML File	549
Log to Database	550
Log to Local Syslog	550
Log to Remote Syslog	550
Log to System Console	551
2. Log Level and Message	552
Overview	552
Configuration	552
3. Log Message Payload	554
Overview	554
Configuration	554
4. Log Access Filter	555
Overview	555
Configuration	556
5. Service Level Agreement (SLA) Filter	557
Overview	557
Response Time Requirements	557
HTTP Status Requirements	558
Communications Failure Requirements	559
Select Alerting System	560
6. Set Service Name	561
Overview	561
Configuration	561
22. Oracle Access Manager	
1. Oracle Access Manager Authorization	562
Overview	562
Configuration	562
2. Logout from Oracle Access Manager SSO Session	563
Overview	563
Configuration	563
3. Oracle Access Manager SSO Token Validation	564
Overview	564
Configuration	564
23. Oracle Entitlements Server	
1. Oracle Entitlements Server Authorization	565
Overview	565

General	565
Settings	565
Application Context	566
2. Get Roles from Oracle Entitlements Server	567
Overview	567
General	567
Settings	567
Application Context	567
24. Resolvers	
1. Relative Path Resolver	568
Overview	568
Configuration	568
2. SOAPAction Resolver	569
Overview	569
Configuration	569
3. SOAP Operation Resolver	570
Overview	570
Configuration	570
25. Routing	
1. Getting Started with Routing Configuration	571
Overview	571
Proxy or Endpoint Server	571
Service Virtualization	571
Choosing the Correct Routing Filters	572
Case 1: Proxy without Service Virtualization	572
Case 2: Proxy with Service Virtualization	573
Case 3: Endpoint without Service Virtualization	574
Case 4: Endpoint with Service Virtualization	575
Case 5: Simple Redirect	577
Case 6: Routing on to an HTTP Proxy	577
Summary	578
2. Routing Wizard	580
Overview	580
Configuration	580
3. Call Internal Service	582
Overview	582
4. Connection	583
Overview	583
General Configuration	583
Trusted Certificates	583
Client SSL Authentication	583
HTTP Authentication	583
Kerberos Authentication	583
Behavior	584
Advanced	585
5. Connect to URL	587
Overview	587
General Configuration	587
Trusted Certificates	587
Client SSL Authentication	587
HTTP Authentication	587
Kerberos Authentication	588
Behavior	588
Advanced	590
6. Dynamic Router	592
Overview	592
Configuration	592

7. HTTP Status Code	593
Overview	593
Configuration	593
8. Insert WS-Addressing	594
Overview	594
Configuration	594
9. Messaging System Filter	595
Overview	595
Request Settings	595
Response Settings	597
10. Read WS-Addressing	598
Overview	598
Configuration	598
11. Rewrite URL	599
Overview	599
Configuration	599
12. Save to File	600
Overview	600
Configuration	600
13. SMTP Routing	601
Overview	601
General Settings	601
Message Settings	601
14. Static Router	602
Overview	602
Configuration	602
15. TIBCO Rendezvous Routing	603
Overview	603
Configuration	603
16. TIBCO Enterprise Messaging System Routing Filter	604
Overview	604
Connection	604
Request	604
Response	605
17. TIBCO Enterprise Messaging System Connection	607
Overview	607
Configuration	607
18. Wait for Response Packets	609
Overview	609
Packet Sniffer Configuration	609
Sniffing Response Packets	610
19. Proxy Servers	611
Overview	611
Configuration	611
26. Security Services	
1. DSS Signature Generation Service	612
Overview	612
Configuration	612
2. DSS Signature Verification Web Service	613
Overview	613
Configuration	613
3. Encrypt and Decrypt Web Services	614
Overview	614
Configuration	614
4. STS Web Service	615
Overview	615
Configuration	615

27. WS-Trust	
1. Consume WS-Trust Message	616
Overview	616
Consume WS-Trust Message Types	616
Message Consumption	616
Advanced	617
2. Create WS-Trust Message	618
Overview	618
Create WS-Trust Message Type	618
Message Creation	618
RST Creation	619
RSTR Creation	619
Advanced Settings	619
28. Extensibility	
1. Writing a Custom Filter using the Oracle Enterprise Gateway SDK	621
Tutorial Overview	621
Policies, Filters, and Message Attributes	621
Oracle Enterprise Gateway SDK Overview	622
Tutorial Prerequisites	622
Oracle Enterprise Gateway SDK Sample Overview	623
Step 1: Create the Typedocs	623
Step 2: Create the Filter Class	626
Step 3: Create Processor Class	628
Step 4: Create Policy Studio Classes	631
Step 5: Build Classes	637
Step 6: Load TypeDocs	638
Step 7: Construct a Policy	646
Step 8: Configure the SimpleFilter	649
Conclusion	652
2. Scripting Language Filter	653
Overview	653
Writing a Script Filter	653
Configuring a Script Filter	653
29. Utility	
1. Abort Filter	655
Overview	655
Configuration	655
2. Configuration Web Service	656
Overview	656
3. Copy/Modify Attributes	657
Overview	657
Configuration	657
4. Execute External Process	658
Overview	658
Configuration	658
5. False Filter	660
Overview	660
Configuration	660
6. HTTP Parser	661
Overview	661
Configuration	661
7. Invoke Policy per Message Body	662
Overview	662
Configuration	662
8. Locate XML Nodes	663
Overview	663
Configuration	663

9. Pause Filter	665
Overview	665
Configuration	665
10. Policy Shortcut	666
Overview	666
Configuration	666
11. Policy Shortcut Chain	667
Overview	667
General Configuration	667
Add a Policy Shortcut	667
Edit a Policy Shortcut	668
12. Quote of the Day	669
Overview	669
Configuration	669
13. Reflect Filter	670
Overview	670
Configuration	670
14. Reflect Message And Attributes Filter	671
Overview	671
Configuration	671
15. Set Response Status	672
Overview	672
Configuration	672
16. Set Message Attribute	673
Overview	673
Configuration	673
17. String Replace Filter	674
Overview	674
Configuration	674
18. Switch on Attribute Value	675
Overview	675
Configuration	675
Adding a Switch Case	675
19. Time Filter	677
Overview	677
General Configuration	677
Basic Time Options	677
Advanced Time Options	678
20. Trace Filter	679
Overview	679
Configuration	679
21. True Filter	680
Overview	680
Configuration	680
30. Web Services	
1. Web Service Filter	682
Overview	682
General Settings	682
Routing	682
Validation	682
Message Interception Points	683
WSDL	685
Monitoring	686
2. Return WSDL	687
Overview	687
Configuration	687
3. Set Web Service Context	688

Overview	688
Configuration	688
31. Common Configuration	
1. Certificate Chain Check	689
Overview	689
Configuration	689
2. Certificate Validation	691
Overview	691
Configuration	691
Configuring URL Groups	692
3. Database Connection	694
Overview	694
Configuring the Database Connection	694
Database Connection Pool Settings	695
Connection Validation	695
4. Database Query	696
Overview	696
Configuration	696
5. Configuring LDAP Directories	698
General Configuration	698
Authentication Configuration	698
6. Signature Location	700
Overview	700
Configuration	700
7. What To Sign	703
Overview	703
ID Configuration	703
Node Locations	705
XPath Configuration	705
XPath Predicates	706
Message Attribute	707
8. Configuring XPath Expressions	708
Overview	708
Manual Configuration	708
XPath Wizard	710
9. MIME/DIME Settings	711
Overview	711
Configuration	711
10. Configuring Connection Groups	712
Overview	712
Configuring a Connection Group	712
Configuring a Connection	712
11. Configuring URL Groups	713
Overview	713
Configuration	713
12. LDAP User Search	714
Configure Directory Search	714
13. Configuring a Transparent Proxy	715
Overview	715
Configuring Transparent Proxy Mode for Incoming Interfaces	715
Configuring Transparent Proxy Mode for Outgoing Calls	715
Configuration Example	715
32. Reference	
Message Attribute Reference	718
Message Filter Reference	752
Glossary of Terms	788

System Requirements

Overview

This topic provides the system requirements for the Oracle Enterprise Gateway and specific requirements for other components. For details on Enterprise Gateway components and concepts, see [Oracle Enterprise Gateway Architecture](#).

Operating System Requirements

This section describes the operating system requirements for the Enterprise Gateway:

Platform	Supported Versions	Hardware Prerequisites
Windows	<ul style="list-style-type: none">Windows Server 2003 SP2Windows Server 2003 R2+Windows Server 2008 SP1+Windows XP SP2+Windows 7	<ul style="list-style-type: none">AMD Opteron, Intel Pentium® at 500 MHz or faster, or Intel EM64TMinimum 500 MB free disk space for installation, 10 GB recommendedMinimum 1 GB physical memory, 4 GB recommended
Solaris	<ul style="list-style-type: none">Solaris 9 Update 9+Solaris 10 Update 4+	<ul style="list-style-type: none">Solaris compatible SPARC processor at 440 MHz, or fasterMinimum 1000 MB free disk space for installationMinimum 1 GB physical memory, 4 GB recommended
Linux	<ul style="list-style-type: none">Oracle Linux 4 (UL7+)Oracle Linux 5 (UL3+)Red Hat Enterprise Linux 4 (UL7+)Red Hat Enterprise Linux 5 (UL3+)SUSE Linux Enterprise Server 10 (SP1+)SUSE Linux Enterprise Server 11 (all SP levels) <p>Oracle software may not run on systems that do not meet these requirements (see Important Note below).</p>	<ul style="list-style-type: none">AMD Opteron, Intel Pentium® at 500 MHz or faster, or Intel EM64TMinimum 500 MB free disk space for installation, 10 GB recommendedMinimum 1 GB physical memory, 4 GB recommended

Important Note

When new Linux kernels and distributions are release, Oracle modifies and tests its products for stability and reliability on these platforms. Oracle makes every effort to add support for new kernels and distributions in a timely manner. However, until a kernel or distribution is added to this list, its use with Oracle products is not supported. If you are running any popular Linux distribution with a 2.6 kernel and libc version 6, Oracle endeavors to support you if issues arise.

Specific Requirements

This section describes requirements for specific components:

Component	Requirement
Policy Studio	Runs on the same platforms as the Enterprise Gateway with the following additional requirements on Linux and Solaris: <ul style="list-style-type: none"> • X-Windows environment • GTK+ 2
Service Manager	Supports the following browsers: <ul style="list-style-type: none"> • Internet Explorer 7.x, 8.x • Firefox 3.5, 3.6 • Safari 4, 5
Real-time Monitoring Console	Requires the Adobe Flash 10.0.x plugin installed in your browser. Supports the same browser versions as Service Manager. Recommended screen resolution is 1024 x 768.
Service Monitor	Server component has the same platform requirements as the Enterprise Gateway. Supports the following databases: <ul style="list-style-type: none"> • MySQL Server 5.1 • Microsoft SQL Server 2000, 2005, 2008 • Oracle 10.2.0.4+, 11.1.0.7+, 11.2.0.1 • IBM DB2 9.1 Client component has same requirements as Real-time Monitoring Console.
Policy Center	Same platform requirements as the Enterprise Gateway, with a recommended free disk space of 50 GB.

Default Ports

This section describes the default ports used by specific components.

Enterprise Gateway

The default ports used by the Enterprise Gateway are as follows:

Description	Port Number
Traffic port	8080
Management port	8090

Policy Center

The default port used by the Policy Center for traffic and management is 8060.

Service Monitor

The default port used by the Service Monitor for viewing reports and management is 8040.

Policy Studio

The default URL addresses used by the Policy Studio to connect to other components are as follows:

Component	Address
Enterprise Gateway server	ht- tp://localhost:8090/configuration/deployments/DeploymentService
Policy Center server	ht- tp://localhost:8060/configuration/deployments/DeploymentService
Service Monitor server	ht- tp://localhost:8040/configuration/deployments/DeploymentService

Installing the Enterprise Gateway on Windows

Prerequisites

This document describes how to install the Enterprise Gateway on Windows platforms. Please see the [System Requirements](#) to ensure that the target machine is of a suitable specification. For details on Enterprise Gateway components and concepts, see the [Oracle Enterprise Gateway Architecture](#).

Install Executable

Download the Enterprise Gateway install executable for Windows, and double-click to begin installing.

Introductory screen

After double clicking the install executable, you are presented with an introductory screen. Click **Next** to continue with the installation.

Choose Install Location

Click **Browse** to specify the directory where you wish to install the Enterprise Gateway. The default directory name is `enterprisegateway`.

Click **Install** to install in the specified directory.

Installing

When you click **Install**, a progress screen is displayed showing the progress of the installation.

Installation Complete

After the installer has finished, an **Installation Complete** screen is displayed. Click **Finish** to complete the installation.

Start the Enterprise Gateway

You can start the Enterprise Gateway in the following ways:

- Console Mode - MS-DOS prompt is visible
- Windows Services - run the Enterprise Gateway as a Windows Service

Console Mode

Select the following menu options:

Start > Programs > Oracle Enterprise Gateway > Start (console mode)

Windows Services

Select the following menu options:

Start > Programs > Oracle Enterprise Gateway > Service

Choose from the following:

- **Install** - install the Enterprise Gateway as a service
- **Start** - start the Enterprise Gateway as a service (requires install option to be run first)
- **Stop** - stop the Enterprise Gateway as a service
- **Remove** - remove the Enterprise Gateway as a service

Windows 7/Vista

On Windows 7 and Windows Vista, when logged in as administrator, you do not have administrator privileges for an application by default, due to Windows User Access Control. To run an application in administrator mode, you must right click the application icon, and select **Run as an administrator**.

To workaroud this issue, you can open a command prompt as an administrator by right clicking the Command Prompt shortcut, and selecting **Run as Administrator**. If successful, the command prompt title is **Administrator: Windows Command Processor**. You can then install the service by running the `mdce install` command.

Important Note:

You can encrypt all sensitive Enterprise Gateway configuration data with an encryption passphrase. For example, you can specify this passphrase in your Enterprise Gateway configuration file, or on the command line when the Enterprise Gateway is starting up. For more details on configuring this passphrase, see [Setting the Encryption Passphrase](#).

Start the Policy Studio

To start the Policy Studio, perform the following steps:

1. Open a command prompt.
2. Change to your Policy Studio installation directory.
3. Start `polycystudio`.

Making a Server Connection

When the Policy Studio starts up, click a link to a server session to display the **Open Connection** dialog. You can use this dialog to specify **Connection Details** (for example, host, port, user name, and password), or to specify **Saved Sessions**.

If you wish to connect to the server using a non-default URL, click **Advanced**, and enter the **URL**. The default Enterprise Gateway server URL is:

```
http://localhost:8090/configuration/deployments/DeploymentService
```

For more details on the settings in the **Open Connection** dialog, see [Connection Details](#).

Installing the Enterprise Gateway on Linux

Prerequisites

This document describes how to install the Enterprise Gateway components on Linux platforms. Please see the [System Requirements](#) to ensure that the target machine is of a suitable specification. For details on Enterprise Gateway components and concepts, see the [Oracle Enterprise Gateway Architecture](#).

Install the Enterprise Gateway

To install the Enterprise Gateway, perform the following steps:

1. Download the Enterprise Gateway `tar.gz` file.
2. Use the appropriate tools on your platform to untar and unzip the archive distribution to a clean directory. The relevant files are installed to the directory from which you run these commands.

Start the Enterprise Gateway

To start the Enterprise Gateway from a shell prompt:

1. Start a shell prompt.
2. Change directory to the `/posix/bin` directory of your Enterprise Gateway installation.
3. Start `./enterprisegateway`.

```
prompt# cd /usr/home/enterprisegateway/posix/bin
prompt# ./enterprisegateway -d
```

Important Note:

You can encrypt all sensitive Enterprise Gateway configuration data with an encryption passphrase. For example, you can specify this passphrase in your Enterprise Gateway configuration file, or on the command line when the Enterprise Gateway is starting up. For more details on configuring this passphrase, see [Setting the Encryption Passphrase](#).

Start the Policy Studio

Follow these steps to start the Policy Studio from a shell prompt:

1. Start a shell prompt.
2. Change your Policy Studio installation directory.
3. Start `./policystudio`.

```
prompt# cd /usr/home/policystudio
prompt# ./policystudio
```

Making a Server Connection

When the Policy Studio starts up, click a link to a server session to display the **Open Connection** dialog. You can use this dialog to specify **Connection Details** (for example, host, port, user name, and password), or to specify **Saved Sessions**.

If you wish to connect to the server using a non-default URL, click **Advanced**, and enter the **URL**. The default Enterprise Gateway server URL is:

```
http://localhost:8090/configuration/deployments/DeploymentService
```

For more details on the settings in the **Open Connection** dialog, see [Connection Details](#).

Installing the Enterprise Gateway on Solaris

Prerequisites

This document describes how to install Oracle components on Solaris platforms. Please see the [System Requirements](#) to ensure that the target machine is of a suitable specification. For details on Enterprise Gateway components and concepts, see the [Oracle Enterprise Gateway Architecture](#).

Install the Enterprise Gateway

To install the Enterprise Gateway, perform the following steps:

1. Download the Enterprise Gateway `tar.gz` file.
2. Use the appropriate tools on your platform to untar and unzip the archive distribution to a clean directory. The relevant files are installed to the directory from which you run these commands.

Start the Enterprise Gateway

To start the Enterprise Gateway from a shell prompt:

1. Start a shell prompt.
2. Change directory to the `/posix/bin` directory of your Enterprise Gateway installation.
3. Start `./enterprisegateway`.

```
prompt# cd /usr/home/enterprisegateway/posix/bin
prompt# ./enterprisegateway -d
```

Important Note:

You can encrypt all sensitive Enterprise Gateway configuration data with an encryption passphrase. For example, you can specify this passphrase in your Enterprise Gateway configuration file, or on the command line when the Enterprise Gateway is starting up. For more details on configuring this passphrase, see [Setting the Encryption Passphrase](#).

Start the Policy Studio

To start the Policy Studio from a shell prompt:

1. Start a shell prompt.
2. Change to your Policy Studio installation directory.
3. Start `./policystudio`.

```
prompt# cd /usr/home/policystudio
prompt# ./policystudio
```

Making a Server Connection

When the Policy Studio starts up, click a link to a server session to display the **Open Connection** dialog. You can use this dialog to specify **Connection Details** (for example, host, port, user name, and password), or to specify **Saved Sessions**.

If you wish to connect to the server using a non-default URL, click **Advanced**, and enter the **URL**. The default Enterprise Gateway server URL is:

```
http://localhost:8090/configuration/deployments/DeploymentService
```

For more details on the settings in the **Open Connection** dialog, see [Connection Details](#).

Installing Policy Studio on Windows

Prerequisites

This document describes how to install Oracle Policy Studio on Windows platforms. Please see the [System Requirements](#) to ensure that the target machine is of a suitable specification. For details on Enterprise Gateway components and concepts, see the [Oracle Enterprise Gateway Architecture](#).

Download the Policy Studio

Download the Policy Studio install executable for Windows, and unzip in your desired location.

Start the Enterprise Gateway

You can start the Enterprise Gateway in the following ways:

- Console Mode - MS-DOS prompt is visible
- Windows Services - run the Enterprise Gateway as a Windows Service

Console Mode

Select the following menu options:

Start > Programs > Oracle Enterprise Gateway > Start (console mode)

Windows Services

Select the following menu options:

Start > Programs > Oracle Enterprise Gateway > Service

Choose from the following:

- **Install** - install the Enterprise Gateway as a service
- **Start** - start the Enterprise Gateway as a service (requires install option to be run first)
- **Stop** - stop the Enterprise Gateway as a service
- **Remove** - remove the Enterprise Gateway as a service

Windows 7/Vista

On Windows 7 and Windows Vista, when logged in as administrator, you do not have administrator privileges for an application by default, due to Windows User Access Control. To run an application in administrator mode, you must right click the application icon, and select **Run as an administrator**.

To workaroud this issue, you can open a command prompt as an administrator by right clicking the Command Prompt shortcut, and selecting **Run as Administrator**. If successful, the command prompt title is **Administrator: Windows Command Processor**. You can then install the service by running the `mdce install` command.

Important Note:

You can encrypt all sensitive Enterprise Gateway configuration data with an encryption passphrase. For example, you can specify this passphrase in your Enterprise Gateway configuration file, or on the command line when the Enterprise Gateway is starting up. For more details on configuring this passphrase, see [Setting the Encryption Passphrase](#).

Start the Policy Studio

To start the Policy Studio, perform the following steps:

1. Open a command prompt.

2. Change to your Policy Studio installation directory.
3. Start `polycystudio`.

Making a Server Connection

When the Policy Studio starts up, click a link to a server session to display the **Open Connection** dialog. You can use this dialog to specify **Connection Details** (for example, host, port, user name, and password), or to specify **Saved Sessions**.

If you wish to connect to the server using a non-default URL, click **Advanced**, and enter the **URL**. The default Enterprise Gateway server URL is:

```
http://localhost:8090/configuration/deployments/DeploymentService
```

For more details on the settings in the **Open Connection** dialog, see [Connection Details](#).

Installing Policy Studio on Linux

Prerequisites

This document describes how to install Oracle Policy Studio on Linux platforms. Please see the [System Requirements](#) to ensure that the target machine is of a suitable specification. For details on Enterprise Gateway components and concepts, see the [Oracle Enterprise Gateway Architecture](#).

Download the Policy Studio

Download the Policy Studio `tar.gz` archive, and use the appropriate tools on your platform to untar and unzip the archive distribution to a clean directory. The relevant files are installed to the directory from which you run these commands.

Start the Enterprise Gateway

To start the Enterprise Gateway from a shell prompt:

1. Start a shell prompt.
2. Change directory to the `/posix/bin` directory of your Enterprise Gateway installation.
3. Start `./enterprisegateway`.

```
prompt# cd /usr/home/enterprisegateway/posix/bin
prompt# ./enterprisegateway -d
```

Important Note:

You can encrypt all sensitive Enterprise Gateway configuration data with an encryption passphrase. For example, you can specify this passphrase in your Enterprise Gateway configuration file, or on the command line when the Enterprise Gateway is starting up. For more details on configuring this passphrase, see [Setting the Encryption Passphrase](#).

Start the Policy Studio

Follow these steps to start the Policy Studio from a shell prompt:

1. Start a shell prompt.
2. Change to your Policy Studio installation directory.
3. Start `./policystudio`.

```
prompt# cd /usr/home/policystudio
prompt# ./policystudio
```

Making a Server Connection

When the Policy Studio starts up, click a link to a server session to display the **Open Connection** dialog. You can use this dialog to specify **Connection Details** (for example, host, port, user name, and password), or to specify **Saved Sessions**.

If you wish to connect to the server using a non-default URL, click **Advanced**, and enter the **URL**. The default Enterprise Gateway server URL is:

```
http://localhost:8090/configuration/deployments/DeploymentService
```

For more details on the settings in the **Open Connection** dialog, see [Connection Details](#).

Installing Policy Studio on Solaris

Prerequisites

This document describes how to install Oracle Policy Studio on Solaris platforms. Please see the [System Requirements](#) to ensure that the target machine is of a suitable specification. For details on Enterprise Gateway components and concepts, see the [Oracle Enterprise Gateway Architecture](#).

Download the Policy Studio

Download the Policy Studio `tar.gz` file, and use the appropriate tools on your platform to untar and unzip the archive distribution to a clean directory. The relevant files are installed to the directory from which you ran these commands.

Start the Enterprise Gateway

To start the Enterprise Gateway from a shell prompt:

1. Start a shell prompt.
2. Change directory to the `/posix/bin` directory of your Enterprise Gateway installation.
3. Start `./enterprisegateway`.

```
prompt# cd /usr/home/enterprisegateway/posix/bin
prompt# ./enterprisegateway -d
```

Important Note:

You can encrypt all sensitive Enterprise Gateway configuration data with an encryption passphrase. For example, you can specify this passphrase in your Enterprise Gateway configuration file, or on the command line when the Enterprise Gateway is starting up. For more details on configuring this passphrase, see [Setting the Encryption Passphrase](#).

Start the Policy Studio

To start the Policy Studio from a shell prompt:

1. Start a shell prompt.
2. Change to your Policy Studio installation directory.
3. Start `./policystudio`.

```
prompt# cd /usr/home/policystudio
prompt# ./policystudio
```

Making a Server Connection

When the Policy Studio starts up, click a link to a server session to display the **Open Connection** dialog. You can use this dialog to specify **Connection Details** (for example, host, port, user name, and password), or to specify **Saved Sessions**.

If you wish to connect to the server using a non-default URL, click **Advanced**, and enter the **URL**. The default Enterprise Gateway server URL is:

```
http://localhost:8090/configuration/deployments/DeploymentService
```

For more details on the settings in the **Open Connection** dialog, see [Connection Details](#).

Upgrading a Previous Enterprise Gateway Installation

Overview

This topic describes the steps involved in upgrading the configuration data (policies, filters, certificate store, and so on) from a previous version of the Enterprise Gateway to version 11.1.1.5.0. This enables you to migrate the policies that you configured in a pre-11.1.1.5.0 version of the Enterprise Gateway to version 11.1.1.5.0.

Prerequisites

These steps assume that you have installed Enterprise Gateway 11.1.1.5.0 on the same machine but in a different directory from an earlier (pre-11.1.1.5.0) version of the Enterprise Gateway, and that you wish to upgrade the previous version to version 11.1.1.5.0.

Important Note:

These steps describe only how to upgrade the configuration data from a previous version to version 11.1.1.5.0. The supplied upgrade scripts do *not* upgrade the version of the Enterprise Gateway software installed on the machine. Please contact the Oracle [Support Team](#) to obtain a copy of Enterprise Gateway 11.1.1.5.0 if you have not already done so.

Warning:

You must ensure that Enterprise Gateway 11.1.1.5.0 is installed into a different directory from the earlier (pre-11.1.1.5.0) version of the Enterprise Gateway.

Running the Upgrade Script

Complete the following steps when upgrading a pre-11.1.1.5.0 installation of the Enterprise Gateway to version 11.1.1.5.0:

Step 1: Start the upgrade script

In your Enterprise Gateway 11.1.1.5.0 installation, run the following command:

Windows	IN-STALL_DIR\tools\upgrade\win32\upgradeconfig
UNIX/Linux	IN-STALL_DIR/tools/upgrade/posix/upgradeconfig

where `INSTALL_DIR` points to the root of your Enterprise Gateway 11.1.1.5.0 installation.

Step 2: Enter location of installation to upgrade

Enter the location of the previous Enterprise Gateway installation that you want to upgrade, for example:

```
c:\oracleenterprisegateway
```

Step 3: Review configuration

The upgrade script obtains the configuration information from the installation selected in the previous step, and displays it at the prompt. For example, this includes the `entities.xml` file for the installation:

```
URL: [file:///INSTALL_DIR/conf/entities.xml]
Passphrase: []
Process Name: [Oracle Enterprise Gateway]
```

where `INSTALL_DIR` points to the root of your previous Enterprise Gateway installation.

Step 4: Specify upgrade directory

The upgrade script asks you to specify the directory in which the newly upgraded configuration files are generated. The default directory is as follows:

Windows	IN- STALL_DIR\tools\upgrade\win32\fed-upgraded
UNIX/Linux	IN- STALL_DIR/tools/upgrade/posix/fed-upgraded

where `INSTALL_DIR` points to the root of your Enterprise Gateway 11.1.1.5.0 installation.

Press the return key to use the default, or specify a different directory name. You can also specify an absolute path to different directory, for example:

```
/home/user/upgrades/from_<previous_version>
```

Step 5: Confirm upgrade

Before upgrading, you are asked to confirm the upgrade of the specified configuration file, for example:

```
file:///INSTALL_DIR/conf/entities.xml
```

To confirm the upgrade, enter **y**. Otherwise, enter **n** to cancel.

Step 6: Upgrade complete

A result message is printed on the screen that reflects the outcome of running the upgrade script.

Updating your Enterprise Gateway Configuration

After you run the upgrade script, you must update your Enterprise Gateway configuration with the new configuration files generated by the script. The examples in this section assume that you ran the upgrade script with the default settings.

Updating your configuration files

To update your Enterprise Gateway configuration files, perform the following steps in your Enterprise Gateway version 11.1.1.5.0 installation:

1. Rename the `INSTALL_DIR/conf/fed` directory to `conf/fed-orig`.
2. Copy the following directory:
`INSTALL_DIR/tools/upgrade/win32/fed-upgraded`
to:
`INSTALL_DIR/conf`
3. Rename the new directory from `fed-upgraded` to `fed`.

Deploying upgraded configuration

To deploy your upgraded configuration in Policy Studio, perform the following steps:

1. Start the Enterprise Gateway using the `enterprisegateway` command in the following directory:

Windows	INSTALL_DIR\Win32\bin
UNIX/Linux	INSTALL_DIR/posix/bin

2. Start the Policy Studio using the `polycystudio` command in your Policy Studio installation directory.
3. Connect to the Enterprise Gateway from the Policy Studio welcome page.

4. In the **Oracle Enterprise Gateway Dashboard**, the Enterprise Gateway should be displayed as out of sync (as expected).
5. Click the **Manage Processes** button at the bottom right to display the **Process List** dialog.
6. Select the existing Enterprise Gateway process, click **Remove Process**, and click **Yes**.
7. Click **Add Process**, and specify the process details (URL, user name, and password).
8. When prompted to download the configuration from the process, click **Yes**, and enter a comment .
9. Click **OK**. The configuration is now versioned, and is no longer displayed as out of sync.

If you encounter any problems with upgrading from a previous version, please contact the Oracle [Support Team](#) with your queries.

License Acknowledgments

Overview

Enterprise Gateway uses several third-party toolkits to perform specific types of processing. In accordance with the Licensing Agreements for these toolkits, the relevant acknowledgments are listed below.

Acknowledgments

Apache Software Foundation:

This product includes software developed by the [Apache Software Foundation](http://www.apache.org/) [http://www.apache.org/]

OpenSSL Project:

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>) [http://www.openssl.org/].

Eric Young:

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)

James Cooper:

This product includes software developed by James Cooper.

Oracle Support

Contact Details

For online technical assistance, and a complete list of locations, primary service hours, and telephone numbers, contact Oracle Technical Support at the following address:
<https://support.oracle.com>

Oracle Enterprise Gateway Overview

Overview

Oracle Enterprise Gateway provides governance, acceleration, integration, and security for SOA systems. Oracle Enterprise Gateway is available on Windows, Linux, and Solaris (for more details, see [System Requirements](#)). The following sections describe the high-level functionality available in the Oracle Enterprise Gateway.

Performance

Oracle Enterprise Gateway accelerates performance as follows:

Processing Offload

The Enterprise Gateway can be used to offload the heavy lifting of XML from application servers, and on to the network. This frees up resources on application servers and enables applications to run faster. The patented high-performance core XML Acceleration engine (VXA), coupled with hardware acceleration ensures wirespeed network performance.

VXA Platform

The core VXA engine is integrated into the Enterprise Gateway to accelerate the essential XML security primitives. This engine provides XML processing at faster levels than those performed by common JAXP implementations in application servers and other applications that sit downstream from the Enterprise Gateway. The VXA engine performs Document Object Model (DOM) processing, XPath, XSLT conversion, and XML validation.

XML Data Enrichment

The Enterprise Gateway can automatically populate content in XML documents from sources such as databases. By putting this functionality on to the XML network infrastructure, data is automatically populated in XML messages before they reach the consuming Web Services. This simplifies and accelerates applications in ESBs and application servers.

Governance

Oracle Enterprise Gateway provides the following governance features:

Ease of Deployment

The Enterprise Gateway includes many features that speed up deployment. For example, certificates and private keys, necessary for many XML security functions, can be issued on board. The Enterprise Gateway has a *deny-by-default* defense posture, to detect and block any unauthorized deployments of Web Services. Policies can be re-applied across multiple application endpoints using simple drop-down menus. Policies can also be imported and exported as XML files. This minimizes the time needed to replicate policies across multiple Enterprise Gateways, or to move from a staging system to production environment.

Centralized Management

A policy management console enables administrators to add security and management policies to the Enterprise Gateway. You can manage policy versions across multiple Enterprise Gateways using the Oracle Policy Studio enterprise policy management tool. This enables enterprise policy management to be brought under centralized control, rather than be managed separately on each Enterprise Gateway.

Web-based system administration tools are also provided to simplify Enterprise Gateway management tasks. Oracle Service Manager provides quick and easy access to enable you to manage your services and policies. You can use the Traffic Monitor and Real-time Monitoring tools to monitor the messages sent through the Enterprise Gateway.

Traffic Throttling

The Oracle Enterprise Gateway protects Web Services from unanticipated traffic spikes by smoothing out the traffic. It also limits clients to agreed Web Service consumption levels in accordance with service usage agreements. This enables Oracle customers to charge their clients for different levels of Web Services usage.

Integration

Oracle Enterprise Gateway provides the following integration features:

Identity Management

Oracle Enterprise Gateway can use an existing Identity Management (IM) infrastructure to perform authentication and authorization of message traffic. For example, integration is provided with LDAP, Microsoft Active Directory, Oracle Access Manager, CA SiteMinder, Entrust GetAccess, IBM Tivoli Access Manager, RSA Access Manager, and other IM products. The Enterprise Gateway also interoperates with leading XML products and platforms, including Microsoft .NET, Oracle WebLogic, IBM WebSphere, and SAP NetWeaver.

Pluggable Pipeline

The Enterprise Gateway's internal message-handling pipeline is extensible, enabling extra access control and content-filtering rules to be added with ease. Customers do not have to wait for a full product release before receiving updates of support for emerging standards and for additional adapters.

Scalable Architecture

The Enterprise Gateway is designed to offer a highly flexible and scalable solution. Network administrators can deploy new Enterprise Gateway instances as needed, and deploy the same or different policies as required. This enables administrators to apply policies at any point in their SOA system. Policy enforcement points can be distributed around the network, anywhere traffic is being passed.

Security

Oracle Enterprise Gateway includes the following security features:

Identity Mediation

Through its support for a wide range of security standards, Oracle Enterprise Gateway enables identity mediation between different identity schemes. For example, the Enterprise Gateway can authenticate external Web Services clients using passwords, but then issue SAML tokens that are used for identity propagation to application servers.

Application-level Networking

The Enterprise Gateway routes data based on sender identity, content, and content type. This enables XML messages to be sent to the appropriate application in a secure manner. This also enables *service virtualization* to be performed, whereby Web Services are exposed to clients with virtual addresses to mask their actual addresses for security and application-delivery reasons. In this way, the Enterprise Gateway serves as an important control point for traffic on the network by shielding endpoint Web Services from direct access.

Audit Trail

The Enterprise Gateway satisfies audit requirements by enabling Web Services transactions to be archived in a tamper-proof store for subsequent audit. Oracle also facilitates privacy compliance support by allowing sensitive information, such as customer names, to be encrypted or stripped out of message traffic.

Oracle Enterprise Gateway Architecture

Overview

This topic provides a high-level overview of the Oracle Enterprise Gateway product architecture, and describes its main components and concepts. For an introduction to the product features and benefits, see the [Oracle Enterprise Gateway Overview](#).

Product Architecture

This section provides a high-level overview of the Oracle Enterprise Gateway architecture. The following simple diagram shows the main components:



This diagram shows a simplified view, which includes a single Enterprise Gateway, Web Service, and client (for example, Oracle Service Explorer). However, you can deploy multiple components to suit the needs of your environment. In addition, this diagram shows communication between the client and the Enterprise Gateway using REST and XML. However, a wide range of transports and protocols is supported (for example, HTTP, JMS, TIBCO, FTP, SMTP, and POP).

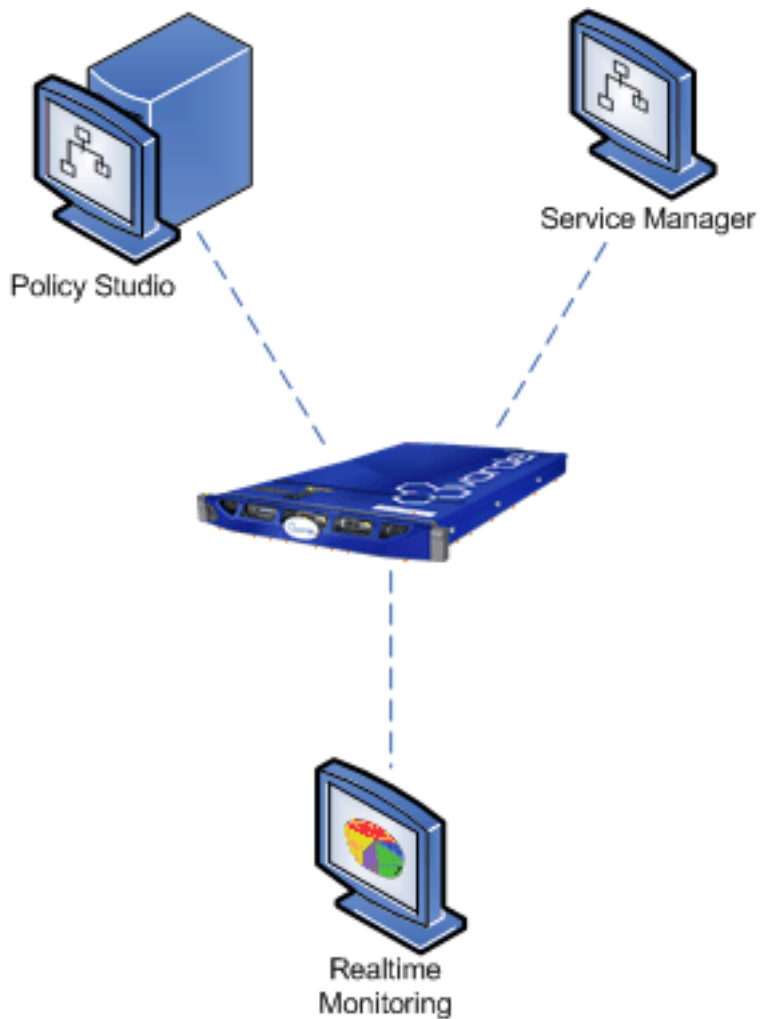
Enterprise Gateway

The Enterprise Gateway provides governance, acceleration, integration, and security for SOA systems. It performs application networking by routing traffic based on content and sender, and by performing XML content screening. The Enterprise Gateway applies policies to incoming messages by running message filters on requests. For more details on Enterprise Gateway features, see [Oracle Enterprise Gateway Overview](#).

Service Explorer

Service Explorer is a Web Services test client, which is used to generate test messages that are sent to the Enterprise Gateway and back to Service Explorer. Service Explorer supports both SOAP-based and REST-based invocations.

The following diagram shows a simplified view of the design-time tools used to manage the Enterprise Gateway:

**Policy Studio**

The Policy Studio is a policy management tool that enables an administrator to easily configure policies and Enterprise Gateway settings to control and protect all deployed Web Services. For example, the Policy Studio enables you to create and assign policies, configure the full range of Enterprise Gateway configuration settings, and manage your Enterprise Gateway deployments. The Policy Studio is typically installed on a separate machine from the Enterprise Gateway to enable remote administration.

Service Manager

Service Manager is a web-based system administration tool that simplifies Enterprise Gateway management tasks. It provides quick and easy access to enable you to manage your services and policies. For example, you can register Web services and assign policies to them.

Real-time Monitoring Console

The Real-time Monitoring Console provides web-based real-time monitoring of HTTP and HTTPS traffic processed by the Enterprise Gateway. This web-based console enables administrators to detect malicious activity in real time, and to take precautionary actions if they feel a service is under attack.

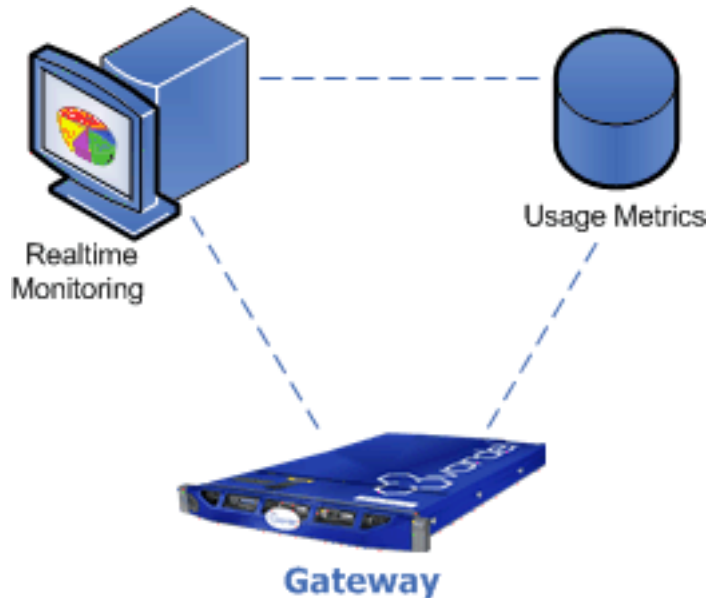
Traffic Monitor

The Traffic Monitor tool provides a web-based message log of the HTTP and HTTPS traffic processed by the Enterprise

Gateway. You can filter messages on a range of criteria (for example, transaction ID, service name, or remote host), drill down to view and save message contents, and change logging settings on-the-fly.

Service Monitor

Service Monitor is a separately installed component that generates reports and charts based on the usage metrics of all the Enterprise Gateways in your network. Service Monitor provides integration with databases such as MySQL Server, MS SQL Server, and Oracle. Service Monitor also includes a Real-time Monitoring Console. For example, you can generate and store reports that monitor which authenticated clients are calling which Web Services.



In a typical deployment scenario, Oracle Enterprise Gateway components are deployed in the demilitarized zone (DMZ). The connection between the client and the Enterprise Gateway is protected by a perimeter firewall, and the connection between the Enterprise Gateway and the Web Service by a Network Address Translation (NAT) firewall.

Product Concepts

This section explains the main concepts in the Oracle Enterprise Gateway product architecture.

Policy

A policy is a network of message filters in which each filter is a modular unit that processes a message. A message can traverse different paths through the network, depending on which filters succeed or fail. For example, this enables you to configure policies that route messages that pass a **Schema Validation** filter to a back-end system, and route messages that pass a different **Schema Validation** filter to a different system.

Filter

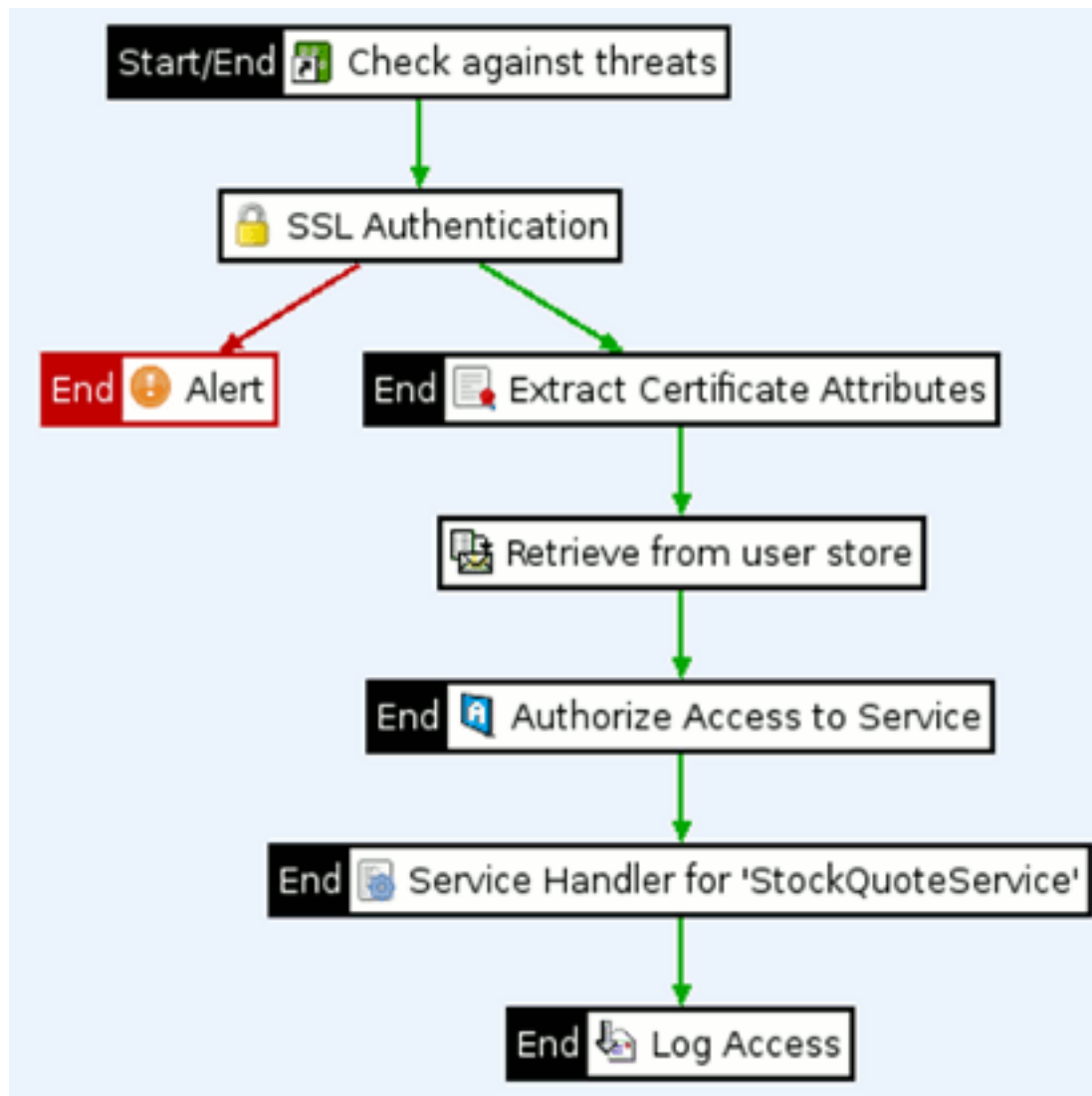
A filter is an executable rule that performs a specific type of processing on a message. For example, the **Message Size** filter rejects messages that are greater or less than a specified size. There are many categories of message filters available with the Enterprise Gateway, including authentication, authorization, content filtering, signing, and conversion. In the Policy Studio, a filter is displayed as a block of business logic that forms part of an execution flow known as a circuit.

Circuit

A circuit is a series of filters through which a message passes. A circuit can contain other circuits, which enables you to build modular reusable policies. In the Policy Studio, a circuit is displayed as a path through a set of filters. A filter can have only one **Success Path** and one **Failure Path**. You can use these success paths and failure paths to create sophisticated rules. For example, if the incoming data matches schema A, scan for attachments, and route to service A, otherwise route to service B. You can configure the colors used to display success paths and failure paths in the Policy Studio.

Preferences menu. You can also specify to **Show Link Labels (S or F)**.

The following example screen shot shows an example circuit with success paths and a failure path:



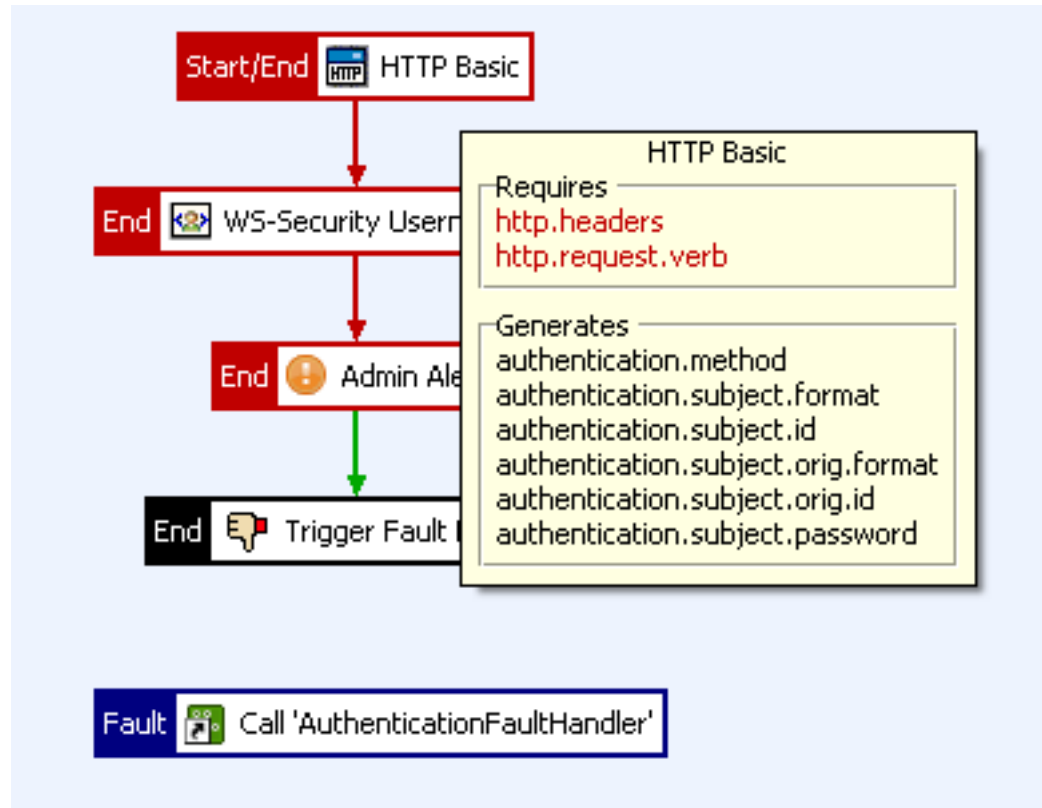
A circuit must have a **Start** filter (in this example, **Check against threats**). Filters labeled **End** stop the execution of the policy if the filter execution fails. Filters labeled **Start/End** indicate that the circuit execution starts there, and that the circuit stops executing if this filter fails. A circuit with a single filter labeled **Start/End** is also valid.

Message Attributes

Each filter requires input data and produces output data. This data is stored in message attributes, and you can access their values using syntax such as `${attribute.name}`. You can also use specific filters to create your own message attributes, and to set their values. The full list of message attributes flowing through a policy is displayed when you right-click the Policy Studio canvas, and select **Show All Attributes**. You can also hover your mouse over a filter to see its inputs and outputs. The **Trace** filter enables you to trace message attribute values at execution time.

The following example screen shot shows the attributes displayed when hovering over an **HTTP Basic** authentication fil-

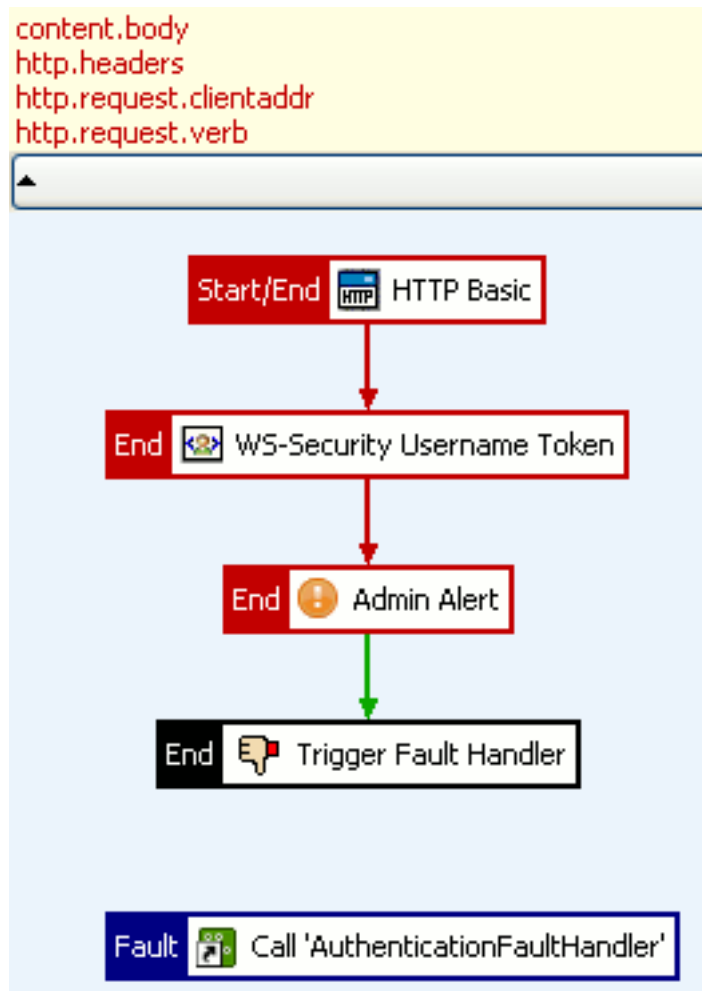
ter:



If a filter requires an attribute as input that has not been generated in the previous execution steps, the filter is displayed in a different color in the Policy Studio (default is red). You can configure the color used to display missing attributes in the Policy Studio **Preferences** menu. Alternatively, you can also view all required attributes by right-clicking the canvas, and selecting **Show All Attributes**.

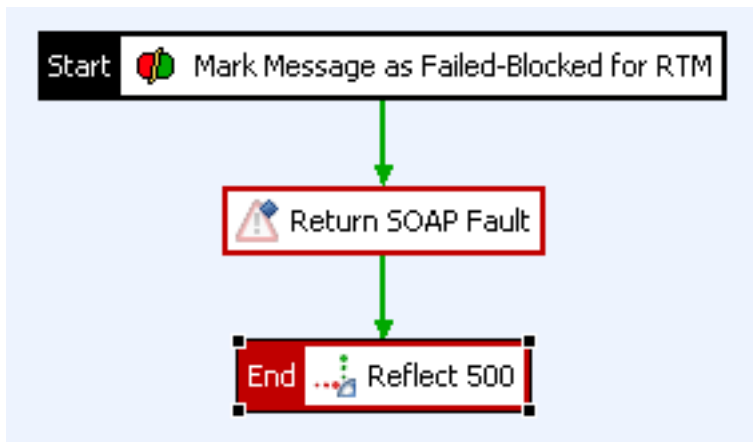
A missing attribute may indicate a problem that you need to investigate (for example, in the chaining of filters or policies, or that the policy can not run on its own). This is often the case for reusable filters, such as those displayed in the previous example.

At the policy level, you also can click the horizontal bar at the top of the Policy Studio canvas to show the list of all attributes required as input to the entire policy. If any attributes are generated by the policy, you can click a corresponding bar at the bottom to see a list of generated attributes. The following example screen shot shows the attributes required by an **Authenticate** policy:



Faults

When a SOAP transaction fails, you can use a SOAP fault to return error information to the SOAP client. By default, the Enterprise Gateway returns a basic SOAP fault to the client when a message filter fails. You can add a **SOAP Fault** filter to a policy to return more detailed error information to the client. For example, the previous example screen shot shows an **AuthenticationFaultHandler**, which is a policy shortcut to the following fault handler policy:



Policy Shortcut

A policy shortcut enables you to create a link from one policy to another policy. For example, you could create a policy that inserts security tokens into a message, and another that adds HTTP headers. You can then create a third policy that calls the other two policies using **Policy Shortcut** filters.

A policy shortcut chain enables you to run a series of policies in sequence without needing to create a policy containing policy shortcuts. In this way, you can create modular policies to perform certain specific tasks, such as authentication, content filtering, returning faults, or logging, and then link these policies together in a sequence using a policy shortcut chain. You can also use Service Manager to automate the creation of a policy shortcut chain simply by dragging and dropping existing policies into a composite policy.

Alerts

The Enterprise Gateway can send alert messages for specified events to various alerting destinations. System alerts are usually sent when a filter fails, but they can also be used for notification purposes. The Enterprise Gateway can send system alerts to the following destinations:

- Windows Event Log
- UNIX/Linux syslog
- SNMP Network Management System
- Check Point Firewall-1
- Email recipient

You can configure alert destinations, and then add an **Alert** filter to a policy, specifying the appropriate alert destination.

Policy Container

A policy container is used to group similar policies together (for example, all authentication or logging policies), or policies that relate to a particular service. A number of useful policies are provided in the **Policy Library** container (for example, policies that return faults, and policies that block threatening content). You can add your own policies to this container, and add your own policy containers to suit your requirements.

Policy Context

Policies can execute in a specified context. You can set a context by associating a relative execution path or listener with a policy. When a policy is called from another policy, the context is set to the calling policy name (for example, **Authenticate**). In the Policy Studio, you can select a context from the drop-down list at the bottom of the canvas, and the Policy Studio displays whether the attributes required for execution are available in that context.

Listeners

You can define different types of listeners and associate them with specific policies. For example, listeners include the following types:

- HTTP/HTTPS
- Directory Scanner
- POP mail server connection
- JMS connection
- TIBCO RV/EMS connection

The Enterprise Gateway can be used to provide protocol mediation (for example, receiving a SOAP request over JMS, and transforming it into a SOAP/HTTP request to a back-end service). For HTTP/HTTPS listeners, policies are linked to a relative path. Otherwise, policies are linked to the listener itself. You can associate a single policy with multiple listeners.

Remote Hosts

You can define a remote host when you need more control of the connection settings to a particular Web Service server. The available connection settings include the following:

- HTTP version
- IP addresses
- Timeouts
- Buffers
- Caches

For example, by default, the Enterprise Gateway uses HTTP 1.1. You can force it to use HTTP 1.0 using Remote Host settings. You must also define a Remote Host if you want to track real-time metrics for a particular host.

Servlet Applications

The Enterprise Gateway provides a Web server and servlet application server that can be used to host static content (for example, documentation for your project), or servlets providing internal services. Static content or servlets can be accessed from a policy using the **Call Internal Service** filter. This feature is not meant to replace an enterprise J2EE server, but rather to enable you to write functionality using technology such as servlets.

Configuration Profile

A Configuration Profile contains the configuration information required to run the Enterprise Gateway. For example, a specific Configuration Profile instance can store certificates, users, core policies and Web Services, external connections, or listeners. A Configuration Profile can have a number of versions, which are created by users. You can use the Policy Studio to deploy Configuration Profile versions to Enterprise Gateway Processes, and to copy existing versions to create new Configuration Profiles.

Process

A Process is a running instance of the Enterprise Gateway. You can use Policy Studio to configure and deploy Enterprise Gateway Processes.

Service Virtualization

When you register a Web Service, and deploy it to the Enterprise Gateway, the Enterprise Gateway virtualizes the Web Service. Instead of connecting to the Web service directly, clients connect through the Enterprise Gateway. The Enterprise Gateway can then apply policies to messages sent to the destination Web Service (for example, to enable security, monitoring, and acceleration).

Starting the Enterprise Gateway

Overview

This topic shows the preliminary steps before you begin. It explains how to start the Enterprise Gateway, the Enterprise Gateway tools (for example, Service Manager, Traffic Monitor, and Real-time Monitoring Console), and Policy Studio. It also lists the main steps in the Getting Started tutorial.

Before you Begin

Check System Requirements

You should ensure that your target machine and platform are supported. For details, see [System Requirements](#).

Install the Enterprise Gateway and Policy Studio

If you have not already done so, see the [Installation Guides](#). For details on upgrading from previous versions, see [Upgrading a Previous Enterprise Gateway Installation](#).

Install Service Explorer

If you wish to use Oracle Service Explorer to test the sample Web Service in the Getting Started tutorial, you must have Service Explorer installed. This step is optional.

Starting the Enterprise Gateway

To start the Enterprise Gateway on the command line, perform the following steps:

1. Change to the following directory in your Enterprise Gateway installation:

Windows	Win32\bin
UNIX/Linux	posix/bin

2. Run the `enterprisegateway` command.

For more information on starting the Enterprise Gateway, see the [Startup Instructions](#).

Accessing the Enterprise Gateway Tools

To access web-based Enterprise Gateway tools such as the Service Manager, Traffic Monitor, and Real-time Monitoring Console, perform the following steps:

1. Enter the following URL to access the **Welcome to Oracle Enterprise Gateway** page:

`http://HOST:8090/`

`HOST` refers to the hostname or IP address of the machine on which the Enterprise Gateway is running (for example, `http://localhost:8090/`). The welcome page contains links to Enterprise Gateway tools and information.

2. Enter your user name and password. The default values are as follows:

User Name	admin
------------------	-------

Password	changeme
-----------------	----------

3. Click the **Service Manager** link to launch this web-based system administration tool for the Enterprise Gateway.
4. Enter your user name and password. The default values are as follows:

User Name	admin
Password	changeme

Service Manager enables you to use your browser to perform a subset of the tasks performed in Policy Studio. This includes tasks such as register Web services, assign policies to them, and create policy shortcut chains. For more information, see [Getting Started with Service Manager](#).

Starting Policy Studio

To start Policy Studio, perform the following steps:

1. In your Policy Studio installation directory, enter the `policystudio` command.
2. In the **Policy Studio**, click a link to a server session to connect to the Enterprise Gateway.
3. In the **Open Connection** dialog, click **OK** to accept the default settings. For more details, see [Connection Details](#).
4. The **Oracle Enterprise Gateway** process is displayed on the **Oracle Enterprise Gateway Dashboard**.

Policy Studio enables you to perform the full range of Enterprise Gateway configuration and management tasks. This includes tasks such as create and assign policies, import Web Services, optimize Enterprise Gateway configuration settings, and manage Enterprise Gateway deployments. For more details on using the Policy Studio to manage Enterprise Gateway processes and configurations, see [Getting Started with Managing Deployments](#).

Getting Started Tutorial

When you have completed the preliminary steps, and started the Enterprise Gateway components, you can then start working through the Getting Started Tutorial. This tutorial is based on a simple Apache Axis Web Service. It includes the following main steps:

1. [Registering a Web Service](#)
2. [Monitoring a Web Service](#)
3. [Securing a Web Service](#)
4. [Monitoring Traffic](#)
5. [Troubleshooting](#)

Registering a Web Service

Overview

This topic explains how to register a simple Apache Axis Web Service using Oracle Service Manager. It uses the example `SimpleAxisServer`, which is installed with the Enterprise Gateway. This example server is provided for illustration only, and is not intended for production use. For more details, see the `samples/stock` directory in an Apache Axis 1.4 installation.

This topic assumes that you have already performed the steps described in [Starting the Enterprise Gateway](#).

Accessing the Axis Web Services

The `SimpleAxisServer` installed with the Enterprise Gateway provides access to the example Axis Web Services. To access these Web services, perform the following steps:

1. In your `INSTALL_DIR\ext\lib` directory, create a `wsdl4j` directory.
2. Download the following file:
<http://sourceforge.net/projects/wsdl4j/files/WSDL4J/1.6.2/wsdl4j-bin-1.6.2.zip> [ht-
tp://sourceforge.net/projects/wsdl4j/files/WSDL4J/1.6.2/wsdl4j-bin-1.6.2.zip]
3. Unzip and place the `wsdl4j.jar` and `qname.jar` files into the `ext\lib\wsdl4j` directory.
4. In your Enterprise Gateway `bin` directory, enter the `axisserver` command. By default, this server runs on port 7070. You can specify a different port number by changing the value of the `-p` option in `INSTALL_DIR/system/conf/axisserver.xml`.
5. Enter the following URL in your browser to view the example Web Services:

```
http://localhost:7070/
```

6. Click a **wsdl** link on this web page to access the WSDL for an example Web Service. This tutorial uses the WSDL for the `ComInfoService`, which is available from the first link on this page.

Testing the ComInfoService

You can test the `ComInfoService` using a simple test client provided with the Enterprise Gateway. In your Enterprise Gateway `bin` directory, enter the `axisstockclient` command. This simple client uses the `ComInfoService` to retrieve the address details of Cisco Systems Inc. When you run the `axistestclient`, you should see the following output:

```
CSCO: San Jose, CA
```

Registering the Example Web Service

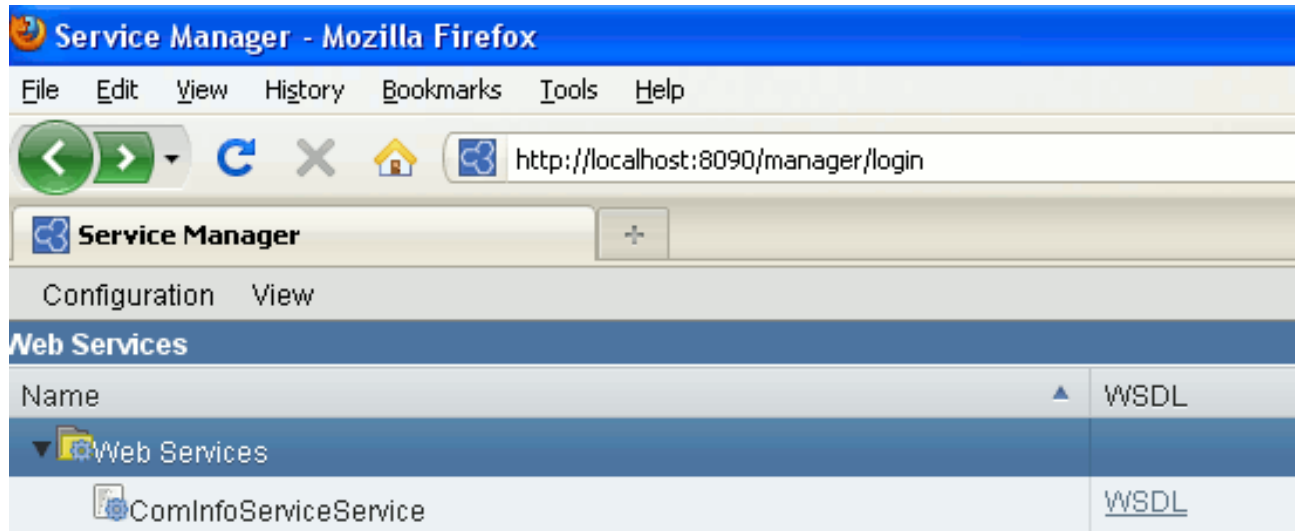
To register the `ComInfoService` example Web Service using Service Manager, perform the following steps:

1. In the **Web Services** window on the left, right-click the **Web Services** group, and select **Register Web Service**.
2. In the **Register Web Service** dialog, specify the following URL for the `ComInfoService` Web Service in the **WSDL URL** field:

```
http://localhost:7070/axis/services/urn:cominfo?wsdl
```

3. Click the **Register** button.
4. Click the **Deploy** icon at the top right of the screen to deploy the Web Service to the Enterprise Gateway.
5. In the **Deploy** dialog, click **Yes**.

The newly registered Web Service is displayed in the **Web Services** tree. For more information, see [Managing Web Services](#).



Web Services registered using Service Manager and deployed to the Enterprise Gateway can also be viewed in the Policy Studio in the **Policies** view. Similarly, Web Services imported using the Policy Studio and deployed to the Enterprise Gateway can be viewed in Service Manager in the **Web Services** window. For details on using Policy Studio to import Web Services, see [Web Service Repository](#).

Accessing the Protected Web Service

When you register a Web Service with the default settings, and deploy it to the Enterprise Gateway, the Enterprise Gateway protects the Web Service. This means that instead of connecting to the Web service directly, clients connect through the Enterprise Gateway. The Enterprise Gateway can then apply policies to messages sent to the destination Web Service (for example, to enable security, monitoring, and acceleration).

To view the WSDL for the protected Web Service, click the link in the **WSDL** column on the right of the **Web Services** window. The published WSDL now uses the host and port of the Enterprise Gateway in its URL. For example, in the case of the example `ComInfoService`, the new URL on which the Web service is available to clients is as follows:

```
http://HOST:8080/axis/services/urn:cominfo?WSDL
```

where `HOST` is the machine on which the Enterprise Gateway is running.

Testing the Protected Web Service

If you have Service Explorer installed, you can test the protected Web Service by sending a message through the Enter-

prise Gateway (this step is optional). To do this, perform the following steps:

1. In your Service Explorer installation directory, run the `serviceexplorer` command.
2. Click the **Import WSDL** button in the Service Explorer toolbar.
3. In the **Load WSDL** screen, select **WSDL URL**, and paste in the URL for the protected WSDL (in this case, `http://HOST:8080/axis/services/urn:cominfo?WSDL`).
4. Click **Next**.
5. In the **WSDL Operations** screen, select the `getInfo` operation.
6. Click **Finish**.
7. In the **Request** tab on the left, paste in the following message:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <ns1:getInfo xmlns:ns1="http://stock.samples">
      <symbol xsi:type="xsd:string">CSCO</symbol>
      <info xsi:type="xsd:string">address</info>
    </ns1:getInfo>
  </soap:Body>
</soap:Envelope>
```

8. Click the triangular green send button in the toolbar to send the message to the protected Web Service through the Enterprise Gateway. The following message should be displayed in the **Response** tab on the right:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:getInfoResponse
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:ns1="http://stock.samples">
      <getInfoReturn xsi:type="xsd:string">San Jose, CA </getInfoReturn>
    </ns1:getInfoResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

9. To format the message, right-click the **Request** pane, and select **Format**.

The following example shows the **Request** displayed in Service Explorer:



Monitoring the Web Service

When you have registered the example Web Service, the next step is to monitor the Web Service using the real-time monitoring tools provided with the Enterprise Gateway. For details, see [Monitoring a Web Service](#).

Monitoring a Web Service

Overview

This topic explains how to monitor the `ComInfoService` example Web Service using the real-time monitoring tools provided with the Enterprise Gateway. It assumes that you have already performed the steps in the following topics:

1. [Starting the Enterprise Gateway](#)
2. [Registering a Web Service](#)

Enabling Monitoring

You must first enable traffic monitoring and real-time monitoring in the Policy Studio. To enable these settings, perform the following steps:

1. In the **Oracle Enterprise Gateway Dashboard**, double-click the **Oracle Enterprise Gateway** process to load the configuration for the active Enterprise Gateway.
2. If a passphrase has been set, enter it in the **Enter Passphrase** dialog, and click **OK**. Alternatively, if no passphrase has been set, click **OK**.
3. In the **Services** tree on the left, right-click the **Oracle Enterprise Gateway** process, and select **Monitoring -> Traffic**.
4. In the **Traffic Monitor Settings** dialog, ensure **Enable Traffic Monitor** is selected.
5. Click **OK**.
6. Right-click the Oracle Enterprise Gateway process, and select **Monitoring -> Metrics**.
7. In the **Real time monitoring settings** dialog, ensure **Enable real time monitoring** is selected.
8. Click **OK**.
9. Click the **Deploy** icon in the Policy Studio toolbar to deploy these settings to the Enterprise Gateway. Alternatively, press F6.

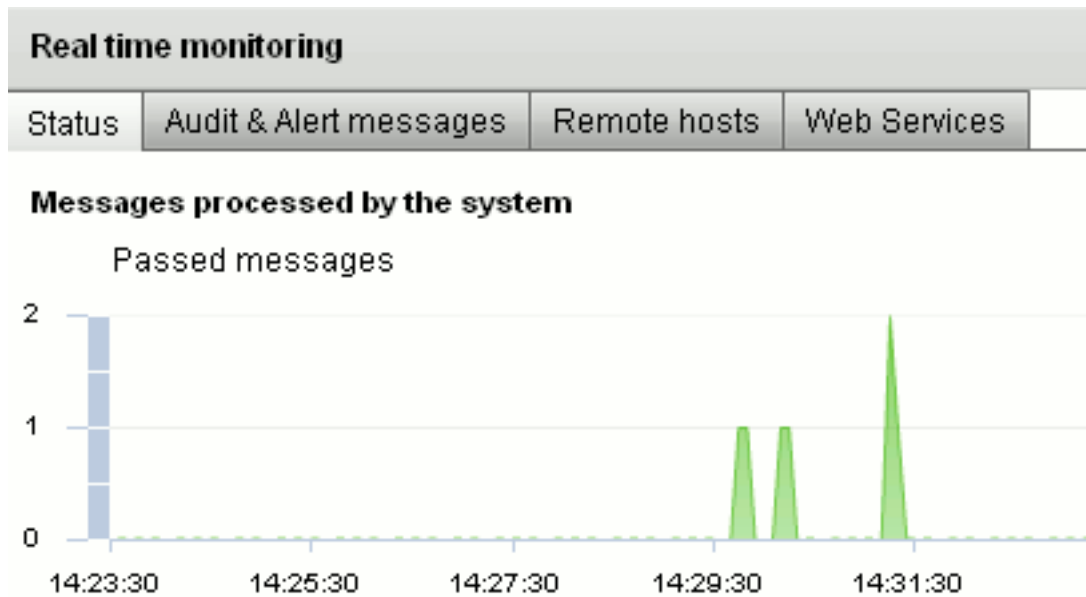
Important Note:

Enabling traffic monitoring settings may have a negative impact on performance. If you wish to maximize performance, do not enable these settings. For more details, see [Configuring Traffic Monitoring](#).

Viewing Real-time Monitoring

You can view a range of monitoring statistics in the **Real-time monitoring** console (for example, status, audit messages, remote hosts, and Web Services). To launch the **Real-time monitoring** console from Service Manager, in the toolbar on the top right, click the **Real-time Monitoring** icon.

The following example shows the number of messages that have been passed by the Enterprise Gateway to the Web Service:



In Service Manager, each time you click the **WSDL** link in the **Web Services** window, the message sent to the example Web Service through the Enterprise Gateway is displayed in the **Real-time monitoring** console. Similarly, in Service Explorer, each time you send a test message to the example Web Service through the Enterprise Gateway, the message is displayed in the console.

Viewing Message Monitoring

When monitoring is enabled, the **Real-time monitoring** console displays summary information for each message. The following example shows the details displayed for messages sent to the ComInfoService through the Enterprise Gateway:

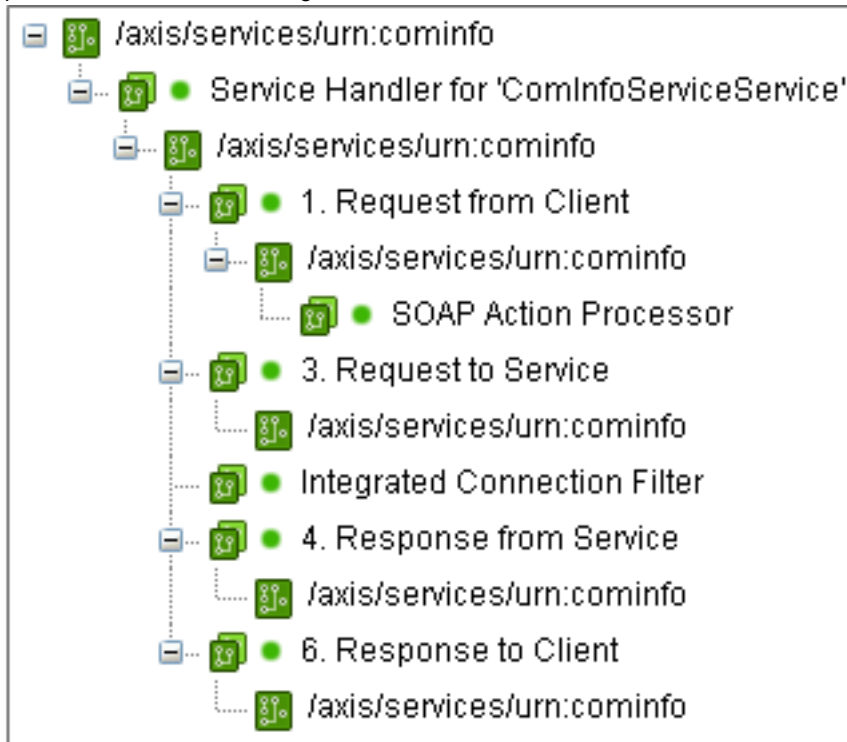
Latest HTTP messages

	Time	Status	Method	Local address	URI	Remote name
●	2011/05/27 11:13:05	200	POST	192.168.148.1	/axis/services/urn:cominfo	192.168.148.1
●	2011/05/27 11:12:48	200	GET	192.168.0.132	/axis/services/urn:cominfo?WSDL	192.168.0.132
●	2011/05/27 11:12:43	200	POST	192.168.24.1	/axis/services/urn:cominfo	192.168.24.1
●	2011/05/27 11:12:42	200	POST	192.168.0.132	/axis/services/urn:cominfo	192.168.0.132
●	2011/05/27 11:12:42	200	POST	192.168.148.1	/axis/services/urn:cominfo	192.168.148.1
●	2011/05/27 11:10:12	200	POST	192.168.24.1	/axis/services/urn:cominfo	192.168.24.1
●	2011/05/27 11:10:06	200	POST	192.168.0.132	/axis/services/urn:cominfo	192.168.0.132

Viewing Message Content

The **Real-time monitoring** console also displays the message path and the contents of each request message and response message. For example, to view the message path and the content of the response message sent by the ComInfoService through the Enterprise Gateway, perform the following steps:

1. In the **Latest HTTP Messages** list, select the message whose contents you wish to view. This displays the message path for the selected message:



2. Select the outbound message from the Enterprise Gateway, which is second in the list on the right of the message path.
3. Click the **View** button for the **Response** message to view its contents.
4. Alternatively, click the **Download** button for the **Response** message to download its contents and save them to a file.

Detecting Malformed Messages

Messages with malformed content are blocked by the Enterprise Gateway and displayed in the **Real-time monitoring** console. To detect a malformed message sent to the `ComInfoService` and blocked by the Enterprise Gateway, perform the following steps:

1. In Service Explorer, on the **Request** tab, change the operation name from `ns1:getInfo` to `ns1:getDetails`.
2. Click the triangular green send button to send the message to the virtualized Web Service through the Enterprise Gateway. The **Response** tab displays an HTTP 500 error and a `MessageBlocked` SOAP fault.
3. In the **Real-time monitoring** console, on the **Status** tab, the message is displayed as blocked.



4. Click the **Audit and Alert Messages** tab to view the audit message for the blocked message.

Monitoring a Remote Host

The **Remote hosts** tab enables you to monitor statistics such as the response time of the Web Service host, and the number of bytes sent and received. To monitor a remote host, perform the following steps:

1. Click the **Remote hosts** tab.
2. In the list of **Configured remote hosts**, select the host to display its statistics, for example:

Remote host: flyer:7070

Display ☒ Totals or data for last ... ☐ 5 seconds ☐ 5 minutes ☐ 1 hour

Message details:	Remote host response codes:	Remote host response times:
Number of requests: 5	1xx: 0	< 10 ms: 1
Requests / second: 0	2xx: 5	< 100 ms: 4
Bytes transfered (in & out): 5894	3xx: 0	< 500 ms: 0
Bytes transfered / sec. (in & out): 0	4xx: 0	< 1000 ms: 0
Bytes sent: 3099	5xx: 0	< 3s: 0
Bytes received: 2795		< 6s: 0
		< 10s: 0
		< 30s: 0
		< 60s: 0
		> 60s: 0

Using Service Monitor

This tutorial explains how to monitor the example Web Service using the real-time monitoring tools provided with the Enterprise Gateway. Service Monitor is a separately installed component that enables you to monitor Web Services and to generate reports on the traffic history of multiple Enterprise Gateways. For details on using Service Monitor, see the following topics:

- [Service Monitor Install Guide](#)
- [Service Monitor User Guide](#)

Securing the Web Service

When you have performed the steps required to monitor the example Web Service, the next steps are to secure the Web Service using Policy Studio. For details, see [Securing a Web Service](#).

Securing a Web Service

Overview

This topic explains how to use Policy Studio to create a simple security policy, how to assign this policy to the `ComIn-foService` Web Service, and how to add a new user. It then shows how to test this security policy using Service Explorer. Finally, this topic shows how to chain policies together and assign them using Service Manager.

Policy Studio enables you to perform the full range of Enterprise Gateway configuration and management tasks (for example, create and assign policies, import Web Services, optimize configuration settings, and manage deployments). Service Manager enables you to use your browser to perform a subset of the tasks performed in Policy Studio (for example, reuse policies created in Policy Studio, and assign them to imported Web Services).

This topic assumes that you have already performed the steps described in the following topics:

1. [Starting the Enterprise Gateway](#)
2. [Registering a Web Service](#)
3. [Monitoring a Web Service](#)

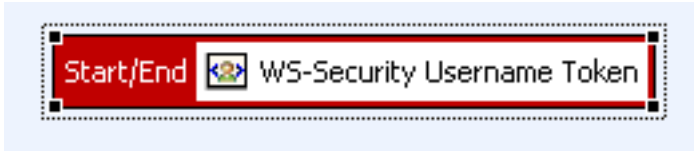
Creating a Security Policy

To create a simple WS-Security authentication policy using Policy Studio, perform the following steps:

1. Click the **Policies** button on the left to view the **Policies** tab.
2. In the **Policies** tree, right-click the **Policy Library** node, and select **Add Policy**.
3. In the **Policy** dialog, enter **WS-Security UsernameToken AuthN** in the **Name** field.
4. Select **Security** from the **Category** drop-down list.
5. Click **OK**. The new policy is added to the tree and displayed as empty on the blank Policy Studio canvas.
6. Click the **Authentication** category on the right of the canvas, and scroll down to select the **WS-Security UsernameToken** filter.
7. Drag and drop the **WS-Security Username Token** filter on to the canvas.
8. Enter the following details in the **Configure a new WS-Security Username Token filter** dialog:

Actor	Select Current actor/role only from the drop-down box.
Drift	Specify a value of 5 seconds drift time to allow for a difference between the clock on the machine hosting the Enterprise Gateway and the machine hosting the Web Service.
Validity Period	Specify a value of 5 mins.
Repository Name	Select Local User Store from the drop-down box.

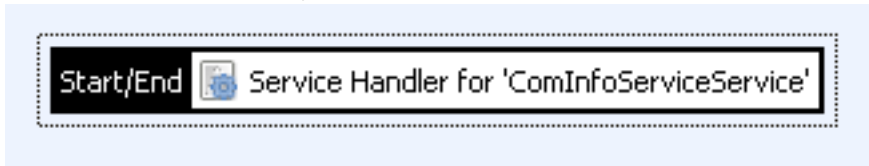
9. Click **Finish**. The new filter is added to the policy on the canvas.
10. Right-click the filter, and select **Set as Start**. This sets the **WS-Security Username Token** filter as the start of this simple policy circuit.



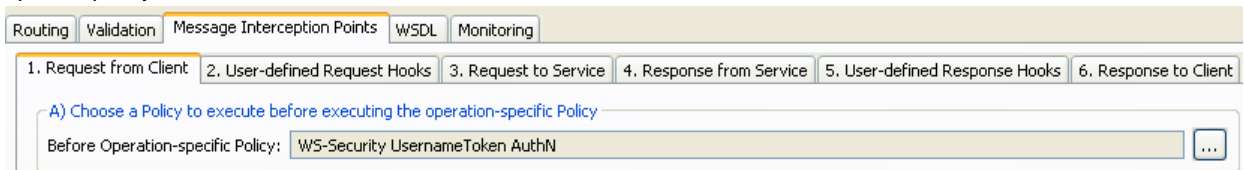
Applying a Security Policy

To apply the newly created WS-Security authentication policy to the `ComInfoService` Web Service using Policy Studio, perform the following steps:

1. In the **Policies** tree on the left, expand the **Generated Circuits** node, and select the **Service Handler for 'ComInfoServiceService'** to display the service handler on the canvas.



2. Double-click the service handler to open it.
3. Click the **Message Interception Points** tab.
4. On the **1. Request from Client** tab, click the button on the right to choose a policy to execute before the operation-specific policy.



5. In the dialog, select the **WS-Security UsernameToken AuthN** policy.
6. Click **OK**.
7. Click **Finish**.

You can also use Service Manager to assign policies to Web Services. For details, see [Managing Web Services](#).

Adding a User

To add a sample user to the local Enterprise Gateway user store to test the **WS-Security UsernameToken AuthN** policy, perform the following steps:

1. Click the **Users** button at the bottom left to view the **Users** tab.
2. Select the **Users** tree node to display the **Users** tab on the right.
3. In the **Users** tab, click the **Add** button.
4. In the **Add User** dialog, enter an example **User Name** and **User's Password**, and confirm the password. You need to remember this password for the next step.
5. Click **OK**.
6. Click the **Deploy** button in the toolbar to deploy these updates to the Enterprise Gateway. Alternatively, press F6.

Testing a Security Policy

If you have Service Explorer installed, you can test the **WS-Security UsernameToken AuthN** policy as follows:

1. Click the triangular green send button to send the message to the virtualized Web Service through the Enterprise Gateway. The **Response** tab displays an HTTP 500 ERROR and a `MessageBlocked` SOAP fault. The **Real-time monitoring** console also displays the message as blocked. This is because the request message now requires a WS-Security UsernameToken, and without this token the message is blocked by the Enterprise Gateway.
2. Select **Security -> Insert WS-Security UserName**.
3. In the **Insert WS Security UserName** dialog, specify the following settings in the **Credential details**:

User name	Enter the user name that you added in Policy Studio.
Include Password	Select this checkbox.
Password	Select this radio button. In the text box, enter the password that you specified in Policy Studio.
Clear	Select this radio button.

4. Click **Finish** to insert the WS Security UserName token into the request message.
5. Click the send button to send the message to the virtualized Web Service through the Enterprise Gateway. The **Response** tab should display an HTTP 200 OK message and the desired response message. The **Real-time monitoring** console also displays the message as passed.

The following example shows the SOAP header with the WS-Security UsernameToken that is inserted into the request message in Service Explorer:

```
<soap:Header>
  <wsse:Security
    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/
      oasis-200401-wss-wssecurity-secext-1.0.xsd">
    <wsse:UsernameToken
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/
        oasis-200401-wss-wssecurity-utility-1.0.xsd"
      wsu:Id="Id-000001289118bc76-0000000000a2f435-2">
      <wsse:Username>joeuser</wsse:Username>
      <wsse:Nonce EncodingType="utf-8">
        pp69a0hHBz0msnYiZ5rTAQ==
      </wsse:Nonce>
      <wsse:Password
        Type="http://docs.oasis-open.org/wss/2004/01/
          oasis-200401-wss-username-token-profile-1.0#PasswordText">
        joepwd
      </wsse:Password>
      <wsu:Created>2011-05-12T09:57:17Z</wsu:Created>
    </wsse:UsernameToken>
  </wsse:Security>
</soap:Header>
```

The following example shows the **WS Security UserName Token** filter displayed in the message path in the **Real-time monitoring** console:



Chaining Policies Together

You can use Service Manager to chain policies together into a composite policy. This is equivalent to a policy shortcut chain in Policy Studio. To create a composite **ComInfoSecurity** policy using Service Manager, perform the following steps:

1. Enter your Service Manager login details to reconnect to the Enterprise Gateway server.
2. Click **Discard** to obtain the latest Enterprise Gateway configuration.
3. In the **Policies** window on the right, right-click the **Policy Library** node, and select **Add Policy**.
4. In the **Policy Details** below, on the **Settings** tab, enter **ComInfoSecurity** in the **Name** field.
5. Select **Security** from the **Category** drop-down list.
6. Click the **Sub-Policies** tab, and drag and drop the **XML Threat Policy** and **WS-Security UsernameToken AuthN** policies from the **Policy** tree.
7. Click **Deploy** to deploy this configuration to the Enterprise Gateway.
8. Click **Yes**.

Edit Policy: ComInfoSecurity

Settings

Sub-Policies

Name

XML Threat Policy

WS-Security UsernameToken AuthN

Alternatively, you can also use Policy Studio to create a policy shortcut chain. For details, see [Policy Shortcut Chain](#).

Assigning the Policy in Service Manager

To assign the composite **ComInfoSecurity** policy to the `ComInfoService` Web Service using Service Manager, perform the following steps:

1. In **Web Services** window on the left, double-click the **ComInfoServiceService** node.
2. In the **Edit Web Service** section below, click the **Policies** tab.
3. Drag and drop the **ComInfoSecurity** policy from the tree in the **Policies** screen on to the **Request** interception point in the diagram or in the table below. This replaces the **WS-Security UsernameToken AuthN** policy previously assigned in Policy Studio.
4. Click **Deploy** to deploy this configuration to the Enterprise Gateway.
5. Click **Yes**.

Edit Web Service: ComInfoServiceService

Settings

Policies

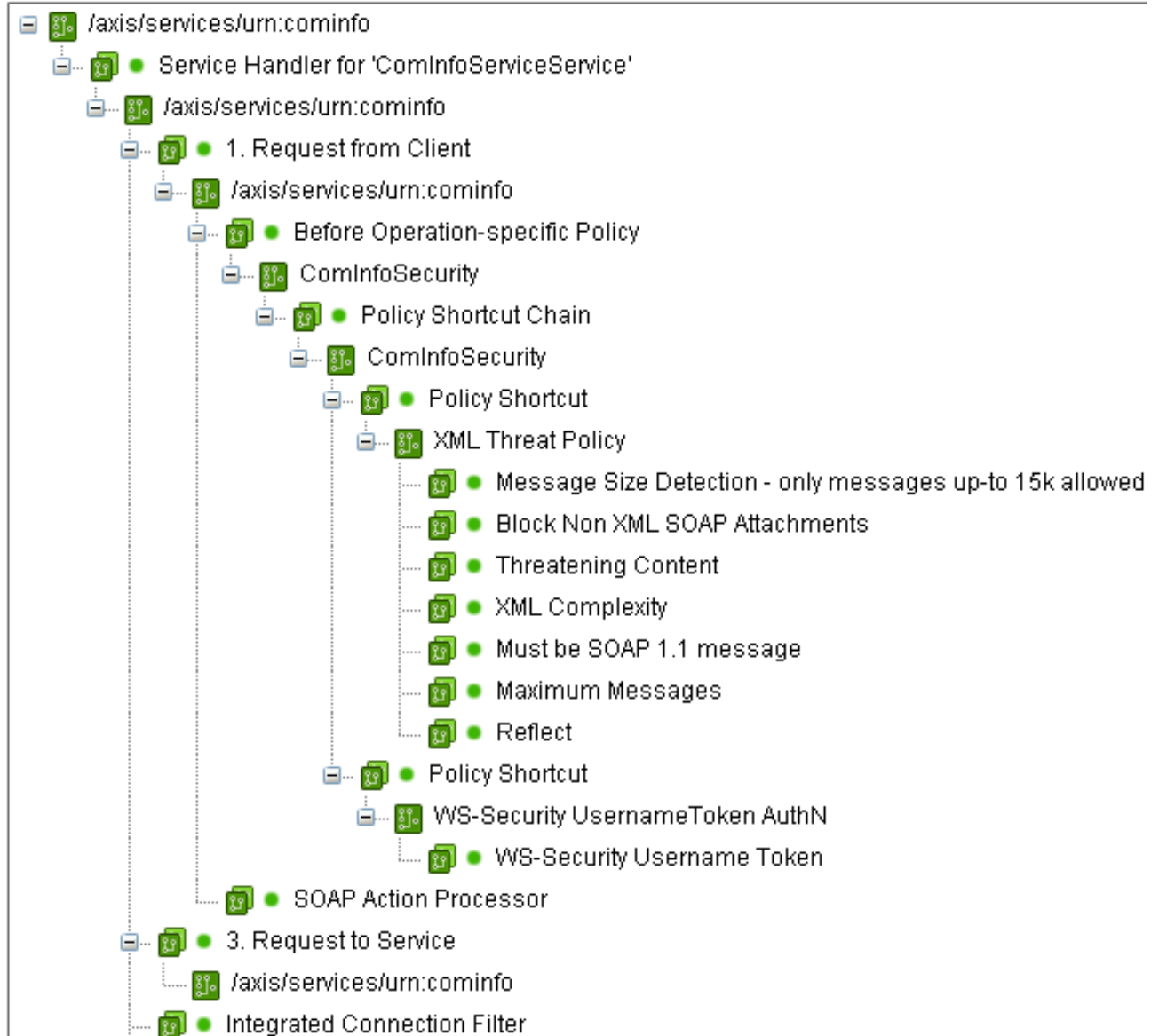
Interception Point	Policy Name	Policy Category
Request	ComInfoSecurity	security
Route		
Response		
Fault		

The **ComInfoSecurity** policy is now assigned to the `ComInfoService` Web Service, and is run on the request received by the Enterprise Gateway from the client. For more information on using Service Manager to assign policies to Web Services, see [Managing Web Services](#).

Testing the Policy in Service Explorer

If you have Service Explorer installed, you can also test the composite **ComInfoSecurity** policy using the steps outlined in [Testing a Security Policy](#). In the Service Explorer **Request** tab, delete the existing `wsse:UsernameToken` before inserting the new WS-Security UsernameToken.

The following example shows the composite **ComInfoSecurity** policy displayed in the message path in the **Real-time monitoring** console:



Switching between Service Manager and Policy Studio

When you deploy updates to the Enterprise Gateway in Service Manager, you must reload the active Enterprise Gateway configuration in Policy Studio to view updates made in Service Manager. For example, when you close and reload the active configuration in Policy Studio, the **ComInfoSecurity** policy should be displayed under the **Policy Library** node in the **Policies** tree in Policy Studio. It should also be displayed under the **Generated Circuits** node in the **Service**

Handler for 'ComInfoServiceService'. When you double-click the Service Handler, the **ComInfoSecurity** policy should be displayed on the **Message Interception Points** tab as the **Request from Client** policy.

Similarly, when you deploy updates to the Enterprise Gateway in Policy Studio, you must reconnect and discard your configuration in Service Manager to view these updates made in Policy Studio. This ensures that Service Manager is synchronized with the active Enterprise Gateway configuration.

Monitoring Traffic

When you have completed these steps to secure the example Web Service, the next steps are to learn how to monitor historical message traffic using the web-based Traffic Monitor tool. For more details, see [Monitoring Traffic](#).

Monitoring Traffic

Overview

This topic introduces the Traffic Monitor tool, which is used for operational diagnostics. Traffic Monitor provides a web-based message log of the HTTP and HTTPS traffic processed by the Enterprise Gateway. You can filter messages on a range of criteria (for example, transaction ID, service name, or remote host), drill down to the message contents, and change logging settings on-the-fly.

Enabling Traffic Monitor

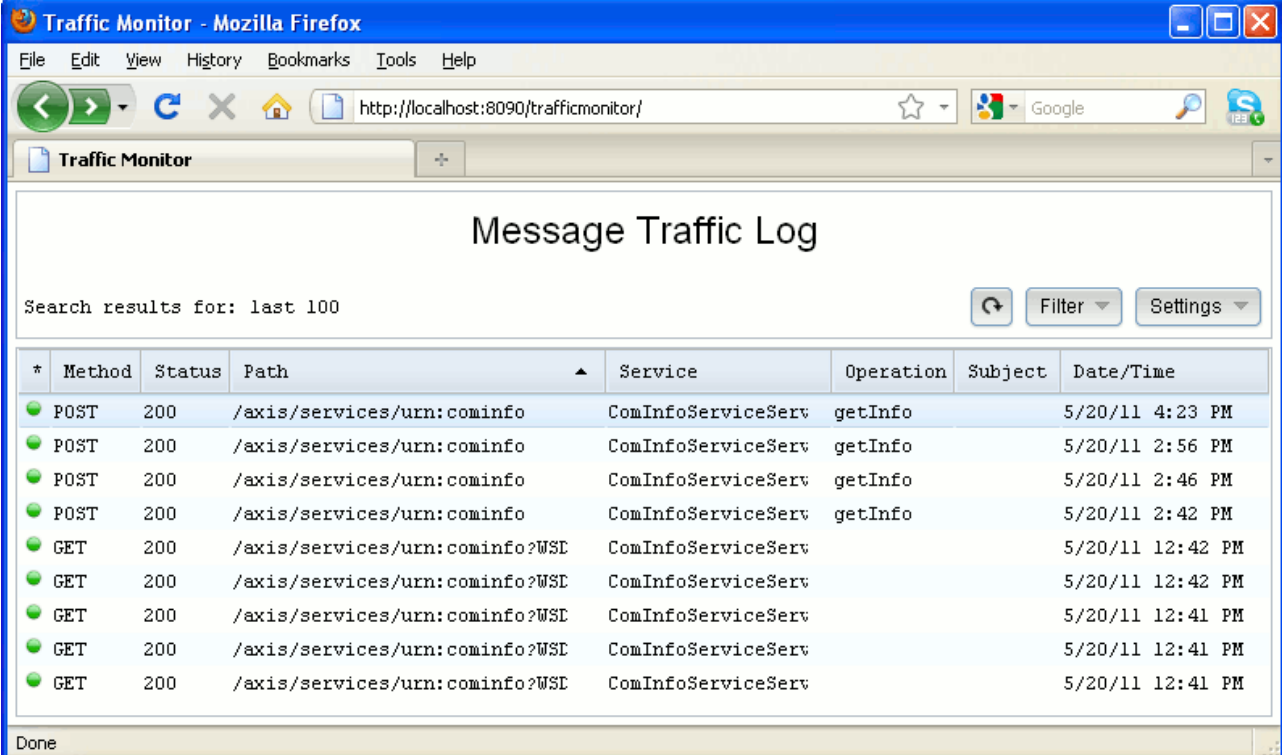
Before you can start viewing messages the Traffic Monitor tool, you must first ensure that Traffic Monitor has been enabled for the Enterprise Gateway process. To enable traffic monitoring in the Policy Studio, perform the following steps:

1. In the **Services** tab on the left, right-click the **Oracle Enterprise Gateway** process, and select **Monitoring -> Traffic**.
2. In the **Traffic Monitor Settings** dialog, ensure **Enable Traffic Monitor** is selected.
3. Click **OK**.

Viewing the Message Traffic Log

The Message Traffic Log provides an historical log of the HTTP and HTTPS traffic processed by the Enterprise Gateway. To view the Message Traffic Log, click the Traffic Monitor link on the **Welcome to Oracle Enterprise Gateway** page, which is available from <http://localhost:8090/index.html>.

For example, the **Message Traffic Log** page is displayed as follows:



The screenshot shows a Mozilla Firefox browser window titled "Traffic Monitor - Mozilla Firefox". The address bar shows the URL <http://localhost:8090/trafficmonitor/>. The page content is titled "Message Traffic Log". Below the title, it says "Search results for: last 100". There are buttons for "Filter" and "Settings". The log is presented as a table with the following columns: *, Method, Status, Path, Service, Operation, Subject, and Date/Time. The table contains 10 rows of data, all with a status of 200. The first two rows are POST requests to `/axis/services/urn:cominfo` with the operation `getInfo`. The remaining eight rows are GET requests to `/axis/services/urn:cominfo?WSL`.

*	Method	Status	Path	Service	Operation	Subject	Date/Time
●	POST	200	/axis/services/urn:cominfo	ComInfoServiceServ	getInfo		5/20/11 4:23 PM
●	POST	200	/axis/services/urn:cominfo	ComInfoServiceServ	getInfo		5/20/11 2:56 PM
●	POST	200	/axis/services/urn:cominfo	ComInfoServiceServ	getInfo		5/20/11 2:46 PM
●	POST	200	/axis/services/urn:cominfo	ComInfoServiceServ	getInfo		5/20/11 2:42 PM
●	GET	200	/axis/services/urn:cominfo?WSL	ComInfoServiceServ			5/20/11 12:42 PM
●	GET	200	/axis/services/urn:cominfo?WSL	ComInfoServiceServ			5/20/11 12:42 PM
●	GET	200	/axis/services/urn:cominfo?WSL	ComInfoServiceServ			5/20/11 12:41 PM
●	GET	200	/axis/services/urn:cominfo?WSL	ComInfoServiceServ			5/20/11 12:41 PM
●	GET	200	/axis/services/urn:cominfo?WSL	ComInfoServiceServ			5/20/11 12:41 PM

Filtering the Message Traffic Log

You can click the **Filter** button on the top right of the screen to filter the messages displayed based on a range of criteria. The default filters include transaction ID, message from (client or gateway), maximum results, and time. The Enterprise Gateway inserts a transaction ID in all HTTP and HTTPS traffic, which has a header named `X-CorrelationID`.

Click the **Add Search Criteria** to search on different criteria (for example, service name, remote host, authentication subject, and operation). When you have selected your search criteria, click the **Search** button.

Configuring Settings

You can click the **Settings** button on the top right of the screen to set logging settings on-the-fly. You do not need to re-fresh or deploy to the Enterprise Gateway. The **System Settings** enable you to specify whether inbound and outbound transactions, the circuit path, and message trace are recorded. You can also select the HTTP interface in the tree on the left to override these settings and configure the interface trace level.







Select the Enterprise Gateway process on the left to configure the process trace level. Finally, you can also select a Relative Path or Web Service on the left, and select **Logging levels** and **Message Payload Logging** on the right.

Viewing Transaction Details

You can click a message entry in the Message Traffic Log to view transactions details for that message. The transaction details include the filter execution path, the contents of the outbound and inbound HTTP request and response messages, and the message trace. You can click to view one of these sections in the screen. Alternatively, click the **Expand All** button at the top right of the screen.







Filter Execution Path

The filter execution path shows the filters that the message passes through in the policy circuit, for example:

Filter execution path				
Filter	Status	Audit Trail Message	Execution Time	Time
[-]  /axis/services/urn:cominfo				
[-] Service Handler for 'ComInfoServiceService'	✓		672	May 20, 2011 4:23:57 PM
[-]  /axis/services/urn:cominfo				
[-] 1. Request from Client	✓		0	May 20, 2011 4:23:56 PM
[-]  /axis/services/urn:cominfo				
SOAP Action Processor	✓		0	May 20, 2011 4:23:56 PM
[-] 3. Request to Service	✓		0	May 20, 2011 4:23:56 PM
 /axis/services/urn:cominfo				
Integrated Connection Filter	✓		656	May 20, 2011 4:23:57 PM
[-] 4. Response from Service	✓		0	May 20, 2011 4:23:57 PM
 /axis/services/urn:cominfo				
[-] 6. Response to Client	✓		0	May 20, 2011 4:23:57 PM
 /axis/services/urn:cominfo				







Request from Client and Response from Gateway

The contents of an example request message from the client and the response message from the Enterprise Gateway are displayed as follows:

▼ Filter execution path				
Filter	Status	Audit Trail Message	Execution Time	Time
[-]  /axis/services/urn:cominfo				
[-] Service Handler for 'ComInfoServiceService'	✓		672	May 20, 2011 4:23:57 PM
[-]  /axis/services/urn:cominfo				
[-] 1. Request from Client	✓		0	May 20, 2011 4:23:56 PM
[-]  /axis/services/urn:cominfo				
SOAP Action Processor	✓		0	May 20, 2011 4:23:56 PM
[-] 3. Request to Service	✓		0	May 20, 2011 4:23:56 PM
 /axis/services/urn:cominfo				
Integrated Connection Filter	✓		656	May 20, 2011 4:23:57 PM
[-] 4. Response from Service	✓		0	May 20, 2011 4:23:57 PM
 /axis/services/urn:cominfo				
[-] 6. Response to Client	✓		0	May 20, 2011 4:23:57 PM
 /axis/services/urn:cominfo				

Request from Gateway and Response from Web Service

The contents of an example request message from the Enterprise Gateway and the response message from the Web Service are displayed as follows:

Filter execution path				
Filter	Status	Audit Trail Message	Execution Time	Time
[-]  /axis/services/urn:cominfo				
[-] Service Handler for 'ComInfoServiceService'	✓		672	May 20, 2011 4:23:57 PM
[-]  /axis/services/urn:cominfo				
[-] 1. Request from Client	✓		0	May 20, 2011 4:23:56 PM
[-]  /axis/services/urn:cominfo				
SOAP Action Processor	✓		0	May 20, 2011 4:23:56 PM
[-] 3. Request to Service	✓		0	May 20, 2011 4:23:56 PM
 /axis/services/urn:cominfo				
Integrated Connection Filter	✓		656	May 20, 2011 4:23:57 PM
[-] 4. Response from Service	✓		0	May 20, 2011 4:23:57 PM
 /axis/services/urn:cominfo				
[-] 6. Response to Client	✓		0	May 20, 2011 4:23:57 PM
 /axis/services/urn:cominfo				

When viewing message contents, you can click the **Save** link to download the contents of the request or response message body. You can also click the **Download** button on the top right of the screen to download the entire transaction in binary format. This provides diagnostic information for the Oracle Support team.

Trace

The trace output for the message is displayed at the currently set trace level (for example, `DEBUG`). You can configure trace levels for the Enterprise Gateway process or HTTP(S) interface using the **Settings** button on the right.

Troubleshooting

When you have viewed the message traffic and message contents, you can learn more about how to configure tracing for the Enterprise Gateway, and how to configure logging for specific message filters. For details, see [Troubleshooting](#).

Troubleshooting

Overview

This topic explains how to configure tracing for the Enterprise Gateway, and how to configure logging for message filters to provide an audit trail of message transactions.

Viewing Enterprise Gateway Trace Files

Each time the Enterprise Gateway starts up, by default, it outputs a trace file to the `INSTALL_DIR\trace` directory. The following example shows an extract from a default Enterprise Gateway trace file:

```
INFO      13:58:29:687 [0804] rolling trace to file trace/Oracle Enterprise Gateway.trc started
INFO      13:58:31:078 [0804] loaded netservice library
INFO      13:58:31:406 [0804] attempting to connect to entity store at
federated:file:///C:/enterprisegateway/conf/fed/configs.xml for process Enterprise Gateway
INFO      13:58:31:984 [0804] Registered EntityTypeFactory: com.vordel.es.EntityTypeFactoryImpl
...
```

The trace file output takes the following format:

```
TraceLevel   Timestamp [thread-id] TraceMessage
```

For example, the first line in the default trace file is described as follows:

Trace Level	INFO
Timestamp	13:58:29:687 (hours:minutes:seconds:milliseconds)
Thread-id	0804
Trace Message	rolling trace to file trace/Oracle Enterprise Gateway.trc started

Setting Enterprise Gateway Trace Levels

The possible trace levels in order of least to most verbose output are as follows:

- FATAL
- ERROR
- INFO
- DEBUG

- DATA

where FATAL is the least verbose and DATA is the most verbose. The default trace level is INFO.

Setting Trace Levels

You can set the trace level in the following different ways:

Startup trace	When the Enterprise Gateway is starting up, it gets its trace level from the <code>tracelevel</code> attribute of the <code>SystemSettings</code> element in <code>/system/conf/enterprisegateway.xml</code> . You can set the trace level in this file if you need to diagnose boot up issues.
System Settings trace	When the Enterprise Gateway has started, it reads its trace level from the System Settings for the Enterprise Gateway process. To set this trace level in the Policy Studio, click the Settings button in the toolbar, select a Trace level from the drop-down list, and click OK .
Interface level trace	You can configure HTTP/HTTPS interfaces with a different trace level from System Settings. For example, the Management Services port (8090) has a default trace level set to ERROR to ensure that it is not too verbose at runtime. To set an interface trace level in the Policy Studio, in the Processes tree, right click an HTTP/HTTPS interface port, and select Edit . Select a Trace level from the drop-down list, and click OK .

Configuring Enterprise Gateway Trace Files

By default, the most recent trace file is named `Oracle Enterprise Gateway.trc`. Older trace files are versioned with the highest version number as oldest (for example, `Oracle Enterprise Gateway0.trc`, `Oracle Enterprise Gateway1.trc`, `Oracle Enterprise Gateway2.trc`, and so on).

You can configure the settings for trace file output in the following file: `INSTALL_DIR/system/conf/enterprisegateway.xml`. By default, this file contains the following trace file setting:

```
<FileRolloverTrace maxfiles="500" />
```

This setting means that the Enterprise Gateway writes trace output to `Oracle Enterprise Gateway.trc` in the `trace` directory of the Enterprise Gateway installation. And the maximum number of files that the `trace` directory can contain is 500.

Note:

Alternatively, you can configure these trace file settings in `INSTALL_DIR/conf/trace.xml`, which is included by `INSTALL_DIR/system/conf/enterprisegateway.xml`.

FileRolloverTrace Attributes

The `FileRolloverTrace` element can contain the following attributes:

<code>filename</code>	File name used for trace output. Defaults to the <code>trace-component</code> attribute read from the <code>SystemSettings</code> element.
<code>directory</code>	Directory where the trace file is written. Defaults to <code>INSTALL_DIR/trace</code> when not specified.
<code>maxlen</code>	Maximum size of the trace file before it rolls over to a new file. Defaults to 16 MB.
<code>maxfiles</code>	Maximum number of files that the trace directory contains for this filename. Defaults to the maximum integer value (2147483647).
<code>rollDaily</code>	Whether the trace file is rolled at the start of the day. Defaults to <code>true</code> .

Writing Trace Output to Syslog

On UNIX and Linux, you can send Enterprise Gateway trace output to syslog. In your `INSTALL_DIR/conf/trace.xml` file, add a `SyslogTrace` element, and specify a facility. For example:

```
<SyslogTrace facility="local0"/>
```

Running Trace at DEBUG level

When troubleshooting, it can be useful to set to the trace level to `DEBUG` for more verbose output. When running a trace at `DEBUG` level, the Enterprise Gateway outputs the status of every circuit and filter that it processes into the trace file.

Debugging a Filter

The trace output for a specific filter takes the following format:

```
Filter name {
    Trace for the filter is indented
    to the following point to make it clear
    to identify output from the filter
} status, in x milliseconds
```

The status is 0, 1, or 2, depending if the filter failed, succeeded, or aborted. For example, the result of an **WS-Security Username Token** filter running successfully is as follows:

```
DEBUG 12:43:59:093 [11a4] run filter [WS-Security Username Token] {
DEBUG 12:43:59:093 [11a4]   WsUsernameTokenFilter.invoke: Verify the username and password
DEBUG 12:43:59:093 [11a4]   WsAuthN.getWSUsernameTokenDetails:
                        Get token from actor=current actor
```



```

DEBUG 12:43:59:093 [11a4] Version handler - creating a new ws block
DEBUG 12:43:59:108 [11a4] Version handler - adding the ws element to the wsodelist
DEBUG 12:43:59:108 [11a4] Version handler - number of ws blocks found:1
DEBUG 12:43:59:124 [11a4] No timestamp passed in WS block so no need to check timestamp
DEBUG 12:43:59:139 [11a4] WsAuthN.getWSUsernameTokenDetails: Check <Created> element
                        in token. Value=2010-08-06T11:43:43Z
DEBUG 12:43:59:139 [11a4] The WS Nonce TimeStamp Max Size is 1000 and wsNonces cache is 4
DEBUG 12:43:59:139 [11a4] Add WS nonce for key [joe:2010-08-06T11:43:43Z].
                        New cache size [5].
DEBUG 12:43:59:155 [11a4] WsBasicAuthN.getUsername: Getting username
DEBUG 12:43:59:171 [11a4] WS-Security UsernameToken authN via CLEAr password
DEBUG 12:43:59:171 [11a4] VordelRepository.checkCredentials: username=joe
DEBUG 12:43:59:186 [11a4] } = 1, in 62 milliseconds

```

Debugging a Circuit

The trace output for a circuit shows the circuit running with all its contained filters, and takes the following format:

```

Circuit name {
  Filter 1{
    Trace for the filter
  } status, in x milliseconds
  Filter 2{
    Trace for the filter
  } status, in x milliseconds
}

```

For example, the following extract shows a circuit called when running the Getting Started demo:

```

DEBUG ... run circuit "/axis/services/urn:cominfo"...
DEBUG ... run filter [Service Handler for 'ComInfoServiceService'] {
DEBUG ...   Set the service name to be ComInfoServiceService
DEBUG ...   Web Service context already set to ComInfoServiceService
DEBUG ...   close content stream
DEBUG ...   Calling the Operation Processor Chain [1. Request from Client]...
DEBUG ...   run filter [1. Request from Client] {
DEBUG ...     run filter [Before Operation-specific Policy] {
DEBUG ...       run circuit "WS-Security UsernameToken AuthN"...
DEBUG ...       run filter [WS-Security Username Token] {
DEBUG ...         ...
DEBUG ...         } = 1, in 62 milliseconds
DEBUG ...       ... "WS-Security UsernameToken AuthN" complete.
DEBUG ...     } = 1, in 74 milliseconds
DEBUG ...   }
...

```

Debugging at Startup

When running a startup trace with a DEBUG level set in the SystemSettings, the Enterprise Gateway outputs the con-

figuration that it is loading. This can often help to debug any incorrectly configured items at start up, for example:

```
DEBUG 14:38:54:206 [1ee0] configure loadable module type RemoteHost, load order = 500000
DEBUG 14:38:54:206 [1ee0] RemoteHost {
DEBUG 14:38:54:206 [1ee0]     ESPK: 1035
DEBUG 14:38:54:206 [1ee0]     ParentPK: 113
DEBUG 14:38:54:206 [1ee0]     Key Fields:
DEBUG 14:38:54:206 [1ee0]         name: {csdwmp3308.wellsfargo.com}
DEBUG 14:38:54:206 [1ee0]         port: {7010}
DEBUG 14:38:54:221 [1ee0]     Fields:
DEBUG 14:38:54:221 [1ee0]         maxConnections: {128}
DEBUG 14:38:54:268 [1ee0]         turnMode: {off}
DEBUG 14:38:54:268 [1ee0]         inputBufSize: {8192}
DEBUG 14:38:54:268 [1ee0]         includeContentLengthRequest: {0}
DEBUG 14:38:54:268 [1ee0]         idletimeout: {15000}
DEBUG 14:38:54:268 [1ee0]         activetimeout: {30000}
DEBUG 14:38:54:268 [1ee0]         forceHTTP10: {0}
DEBUG 14:38:54:268 [1ee0]         turnProtocol: {http}
DEBUG 14:38:54:268 [1ee0]         includeContentLengthResponse: {0}
DEBUG 14:38:54:268 [1ee0]         addressCacheTime: {300000}
DEBUG 14:38:54:268 [1ee0]         outputBufSize: {8192}
DEBUG 14:38:54:268 [1ee0]         sessionCacheSize: {32}
DEBUG 14:38:54:268 [1ee0]         _version: {1}
DEBUG 14:38:54:268 [1ee0]         loaderorder: {500000}
DEBUG 14:38:54:268 [1ee0]         class: {com.vordel.dwe.NativeModule}
DEBUG 14:38:54:268 [1ee0] }
```

For details on setting trace levels and running a startup trace, see [Setting Enterprise Gateway Trace Levels](#).

Running Trace at DATA level

When the trace level is set to DATA, the Enterprise Gateway writes the contents of the messages that it receives and sends to the trace file. This enables you to see what messages the Enterprise Gateway has received and sent (for example, to reassemble a received or sent message).

Note:

On Windows, you can not rely on the console output because it truncates large messages.

Every HTTP request handled by the Enterprise Gateway is processed in its own thread, and threads can be reused when an HTTP transaction is complete. You can see what has happened to a message in the Enterprise Gateway by following the trace by thread ID. Because multiple messages can be processed by the Enterprise Gateway at the same time, it is useful to eliminate threads that you are not interested in by searching for items that only match the thread you want.

For example, you can do this using vi by specifying the thread ID as a pattern to search for in the trace file. In this case, the thread ID is 145c:

```
:g!/145c/d
```

The following example shows the DATA trace when a message is sent by the Enterprise Gateway (message starts with snd):

```

DATA 17:45:35:718 [145c] snd 1495: <POST /axis/services/urn:cominfo HTTP/
1.1Connection: closeContent-Length: 1295User-Agent: VordelSOAPAction:
"Via: 1.0 devsupport2 (Vordel)Host: devsupport2:7070Content-Type:
text/xml<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soap:Header>
  <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-secext-1.0.xsd">
    <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-utility-1.0.xsd"
      wsu:Id="Id-00000128d05aca81-00000000009d04dc-10">
      <wsse:Username>joeuser</wsse:Username>
      <wsse:Nonce EncodingType="utf-8">
        gmP9GCjoe+YIuKleinlENA==
      </wsse:Nonce>
      <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-username-token-profile-1.0#PasswordText">
        joepwd
      </wsse:Password>
      <wsu:Created>2010-05-25T16:45:30Z</wsu:Created>
    </wsse:UsernameToken>
  </wsse:Security>
</soap:Header>
<soap:Body>
  <ns1:getInfo xmlns:ns1="http://stock.samples">
    <symbol xsi:type="xsd:string">CSCO</symbol>
    <info xsi:type="xsd:string">address</info>
  </ns1:getInfo>
</soap:Body>
</soap:Envelope>
>

```

The following example shows the DATA trace when a message is received by the Enterprise Gateway (message starts with rcv):

```

DATA 17:45:35:734 [145c] rcv 557: <HTTP/1.0 200 OKSet-Cookie: 8Set-Cookie2: 8Content-Type:
text/xml; charset=utf-8<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
/XMLSchema-instance">
<soapenv:Body>
  <ns1:getInfoResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns1="http://stock.samples">
    <getInfoReturn xsi:type="xsd:string">San Jose, CA</getInfoReturn>
  </ns1:getInfoResponse>
</soapenv:Body>
</soapenv:Envelope>
>

```

If you want to see what has been recieved by the Enterprise Gateway on this thread, run the following command:

```
:g!/145c] rcv/d
```

All `snd` and `rcv` trace statements start and end with `<` and `>` respectively. If you are assembling a message by hand from the `DATA` trace, remember to remove characters. In addition, the sending and/or receiving of a single message may span multiple trace statements.

Integrating Trace Output with Apache log4J

Apache log4j is included on the Enterprise Gateway classpath. This is because some third-party products that the Enterprise Gateway interoperates with require log4j. The configuration for log4j is found in the Enterprise Gateway `INSTALL_DIR/system/lib` directory in the `log4j.properties` file.

For example, to specify that the log4j appender sends output to the Enterprise Gateway trace file, add the following setting to your `log4j.properties` file:

```
log4j.rootLogger=DEBUG, A1, Vordel
log4j.appender.Vordel=com.vordel.trace.VordelTraceAppender
```

Configuring Logging Output

The Enterprise Gateway provides detailed logging for specific message filters (for example, the request, the time of the request, where the request was routed to, and the response returned to the client). You can configure logging to a number of different locations:

- Text file
- XML file
- Database
- Local syslog
- Remote syslog
- System console

To configure where logging information is sent, perform the following steps:

1. In the Policy Studio tree, right-click your process (for example, the **Oracle Enterprise Gateway** process), and select **Logging -> Custom**.
2. Specify the required settings on the appropriate tab (for example, **Text File**, **Database**, or **XML File**).
3. Click **OK**.
4. Click the **Deploy** button in the toolbar to deploy your settings to the Enterprise Gateway.

For details on configuring all the available options, see the [Logging Configuration](#) topic.

Configuring Log Level and Message

You can configure the log level and log message for a specific filter as follows:

1. In the Policy Studio tree, click a policy to display it in the canvas on the right (for example, **WS-Security Username-Token AuthN** created in the Getting Started tutorial).
2. Double-click the **WS-Security UsernameToken** filter on the canvas to edit it.
3. Click **Next** to display the **Log Level and Message** screen.
4. Select the **Fatal** and **Failure** log levels for troubleshooting.
5. Specify any non-default log messages if required.
6. Click **Finish**.
7. Click the **Deploy** button in the toolbar to deploy your settings to the Enterprise Gateway.

For more details, see the [Log Level and Message](#) topic.

Further Details

For details on logging the message payload at any point in a circuit, see the [Log Payload Filter](#) topic. For details on logging the access details of a message (for example, remote hostname, user login name, and authenticated user name), see the [Log Access Filter](#) topic.

Getting Help

Context-sensitive help is available from the Policy Studio screens. Simply click the **Help** button on any screen to display the relevant help page for that screen. If you require further information or assistance, please contact the Oracle [Support Team](#).

Contacting Support

It is important to include as much information as possible when sending support emails to the Oracle Support team. This helps to diagnose and solve the problem in a more efficient manner. The following information should be included with any support query:

- Name and version of the product (for example, Oracle Enterprise Gateway 11.1.1.5.0).
- Details of patches that were applied to the product, if any.
- Platform on which the product is running.
- A clear (step-by-step) description of the problem or query.
- If you have encountered an error, the error message should be included in the email. It is also useful to include any relevant trace files from the `/trace` directory of your product installation, preferably with the trace level set to `DE-BUG`.

Downloading Diagnostics

The Enterprise Gateway welcome page available on `http://HOST:8090/` includes a **Download Diagnostic Information** link. If you need assistance debugging a problem, you can click this link to zip up all the relevant trace and configuration files and send them to the Oracle Support team.

Getting Started with Service Manager

Introduction

Service Manager is a web-based system administration tool that simplifies Enterprise Gateway management tasks. It provides quick and easy access to enable you to manage your Web Services and policies online. For example, you can perform tasks such as the following:

- [Register Web Services and assign policies](#)
- [Add policies composed of existing policies](#)
- [Deploy configuration to the Enterprise Gateway](#)
- [Access real-time monitoring of Web Services and messages](#)

Logging in to Service Manager

To log in to Service Manager in your browser, perform the following steps:

1. Ensure that the Enterprise Gateway is running (for example, use the `enterprisegateway` command to start).
2. Enter the following URL in your browser:

```
http://localhost:8090/manager/
```

3. Enter your user name and password. The default values are as follows:
 - **User Name:** `admin`
 - **Password:** `changeme`

Note	You can log in to Service Manager from multiple browsers. However, only the most recent login is a valid session from which you can make updates. You can not make updates from multiple browsers concurrently. By default, Service Manager sessions expire after 30 minutes of inactivity.
-------------	---

Logging out of Service Manager

To log out of Service Manager, perform the following steps:

1. Click the Logout icon in the top right corner of the Service Manager screen. The tool tip displays the current user (for example, Logout user `admin`).
2. In the **Confirm Logout** dialog, click **Yes**.

When you log out, your session is expired. However, any updates that you have not yet deployed remain unchanged until you log in again and make more updates, or discard the updates not yet deployed.

To log in again as the same user, click the **Log in** button in the Service Manager Login web page. To log in again as a different user, specify the user name and password, and click the **Log in** button. Alternatively, open a new browser, and perform the steps described in [Logging in to Service Manager](#).

Note	When using Firefox, before opening a new browser to log in as a different user, close all existing Service Manager browser sessions. This is due to how sessions are managed in Firefox.
-------------	--

Deploying to the Enterprise Gateway

You can deploy configuration updates to the Enterprise Gateway by clicking the **Deploy** icon at the top right of the Service Manager screen. Alternatively, select **Configuration -> Deploy** from the Service Manager main menu at the top left.

The **Deploy** menu item is enabled only if the configuration specified in Service Manager differs from the active configuration deployed in the Enterprise Gateway. These configurations are compared when Service Manager is launched, and after each time you make an update in Service Manager. If the configuration specified in Service Manager differs from the active configuration deployed in the Enterprise Gateway, a `Configuration needs to be Synchronized` message is displayed, and **Deploy** is enabled.

After you deploy your updates, **Deploy** is disabled because your configuration is now the same as the current active Enterprise Gateway configuration. Each time you access Service Manager, it checks that your local configuration is in-sync with the current active Enterprise Gateway configuration. It checks that no other user deployed changes after you last took a copy of the current Enterprise Gateway configuration and started editing. If another user deployed changes, you can re-sync your deployment in the **Out-of-Sync** dialog.

Discarding from Service Manager

You can discard configuration updates from Service Manager, and obtain updates from the Enterprise Gateway, by clicking the **Discard** icon at the top right of the Service Manager screen. Alternatively, select **Configuration -> Discard** from the Service Manager main menu at the top left.

Discarding updates enables you to reset the Service Manager copy of the configuration back to the current active Enterprise Gateway configuration. **Discard** is enabled only if the configuration in Service Manager differs from the active configuration deployed in the Enterprise Gateway. If the configuration specified in Service Manager differs from the active configuration deployed in the Enterprise Gateway, a `Configuration needs to be Synchronized` message is displayed, and **Discard** is enabled.

After you discard your updates, **Discard** is disabled because your configuration is now the same as the current active Enterprise Gateway configuration. **Discard** is managed in the same way as **Deploy**.

Managing Web Services

Overview

You can use the **Web Services** screen in Service Manager to register WSDL files in the Web Services repository. This stores Web Service definitions and their related XML schemas. Web Service clients can query the repository for the WSDL file, and use it to send messages to the Web Service through the Enterprise Gateway.

When you register a WSDL file with the repository, the Enterprise Gateway exposes a *virtualized* version of the Web Service. The host and port for the Web Service are changed dynamically to point to the machine running the Enterprise Gateway. The client can then retrieve the WSDL for the virtualized Web Service from the Enterprise Gateway, without knowing its real location. Some policies are also automatically generated for the registered Web Service. These include resolvers and connection filters, and are hidden by default in Service Manager.

You can also use the **Web Services** screen to perform tasks such as assigning policies to a Web Service by dragging and dropping from the **Policies** screen on to the Web Service.

Web Service Groups

WSDL files are registered in *Web Service groups*, which provide a convenient way of grouping related Web Service definitions. You can register a WSDL file by right-clicking the default **Web Services** group in the **Web Service** screen, and selecting **Register Web Service**.

Alternatively, you can add a new Web Services group by right-clicking an existing group, and selecting **Add Web Services Group**. You can also select **Add Web Services Root Group** to add a top-level group.

To edit the Web Service group name (for example, the default **New Web Services Group** name), double-click the group name in the **Web Services** tree, and edit the name on the **Settings** tab below.

Registering a Web Service

To register a Web Service, perform the following steps:

1. Select a Web services group (for example, the default **Web Services** group).
2. Right-click, and select **Register Web Service**.
3. In the **Register Web Service** dialog, specify the following fields:

WSDL URL	The URL for the Web Service (for example, <code>http://www.example.com/services/myService.svc?WSDL</code>).
WSDL Connectivity Timeout	The timeout when the Service Manager waits for a response, and the network connection does not return (defaults to 60 seconds).
Web Service Deployment	Where the Web Service is deployed (under the Default Services relative path).

4. Click **Register**.
5. Click **Deploy** if you wish to deploy the Web Service to the Enterprise Gateway now.

The newly registered Web Service is displayed in the **Edit Web Service** section below. There are no policies assigned

yet in the **Policies** tab.

Accessing a Virtualized Web Service

When you register a Web Service using the default settings, and deploy it to the Enterprise Gateway, you can view its virtualized WSDL by clicking the link in the **WSDL** column on the right. The published WSDL now uses the host and port of the Enterprise Gateway in its URL. For example, if the Enterprise Gateway is running on a machine named `services`, the new URL on which the Web service is available to clients is: `http://services:8080/services/myService.svc?WSDL`.

Note:

You must first deploy the newly registered Web Service before you can view its virtualized WSDL.

Assigning Policies to a Web Service

You can assign policies to a Web Service by dragging and dropping from the **Policies** screen to the **Web Services** screen. All policies are optional, and run at specified points during message processing.

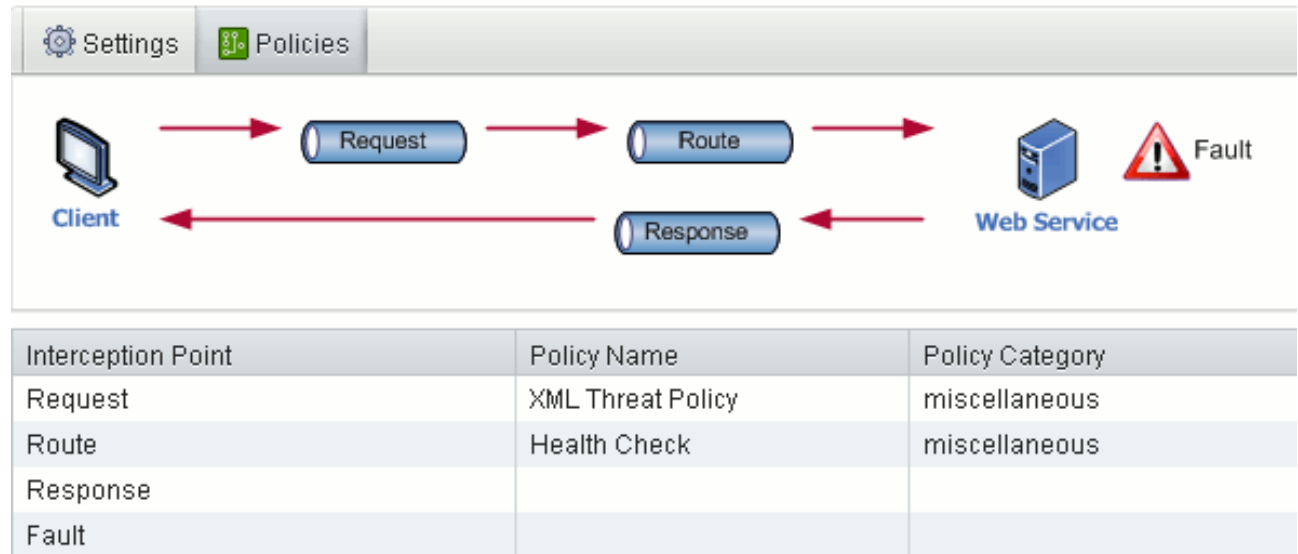
For example, if your Web Service requires complex routing for security, you can assign a routing policy to the **Route** interception point. If no policy is assigned to the **Route** interception point, the message is forwarded to the Web Service using the URL defined in the WSDL. You can assign policies to the following interception points:

Request	Policy is run on the request received from the client.
Route	Policy for routing the request to the Web Service.
Response	Policy is run on the response from the Web Service before it is sent back to the client.
Fault	Policy for handling faults.

To assign policies to a Web Service, perform the following steps:

1. In the **Edit Web Service** section, click the **Policies** tab.
2. Open the **Policies** screen.
3. Drag and drop a policy (for example, **XML Threat Policy**) from the **Policies** tree on to the appropriate icon in the diagram or in the table (for example, **Request**).
4. Repeat these steps to assign policies to other interception points.
5. Click **Deploy** if you wish to deploy this configuration to the Enterprise Gateway.

Edit Web Service: HelloWorldService



Removing an Assigned Policy

You can remove an assigned policy as follows:

1. In the **Policies** tab, select an assigned policy in the table.
2. Right-click, and select **Remove policy from interception point**.

Alternatively, you can drag and drop a different policy to overwrite an assigned policy.

Editing a Web Service

When you have registered a Web Service, you can edit its details as follows:

1. In the **Web Services** tree, select the Web Service to be edited.
2. Right-click, and select **Edit Web Service**.
3. In the **Edit Web Service** section, make your updates on the appropriate tab.

Deleting a Web Service or Group

You can delete a Web Service or Web Service group as follows:

1. In the **Web Services** tree, select the Web Service or group to be deleted.
2. Right-click, and select **Delete Web Service**.

Managing Policies

Overview

A *policy* is a sequence of modular, reusable message filters, each of which processes a message in a particular way. For example, a typical policy might contain an authentication filter (WS-Security UserNameToken), followed by several content-based filters (Schema Validation, Threatening Content, and Message Size). If all configured filters run successfully, the message is routed on to the configured destination.

A policy can be seen as a network of message filters. A message can traverse different paths through the network depending on which filters succeed or fail. For example, you can configure policies that route messages that pass a Schema Validation filter to a back-end system, and route messages that pass a different Schema Validation filter to a different system.

You can use the **Policies** screen in Service Manager to perform tasks such as adding a policy composed of other policies, and editing policy details.

Note	You can use Service Manager to add composite policies only. You must use Policy Studio to create new policies. For more details, see the Policy Configuration topic.
-------------	--

Policy Containers

A *policy container* is used to group together similar policies (for example, all authentication policies), or policies that relate to a particular service (for example, all policies for a *StockQuote* Web Service).

A number of useful policies are provided in the **Policy Library** container (for example, policies that return faults to the client, and policies that block threatening content). You can add your own policies to this container, and add your own policy containers to suit your requirements.

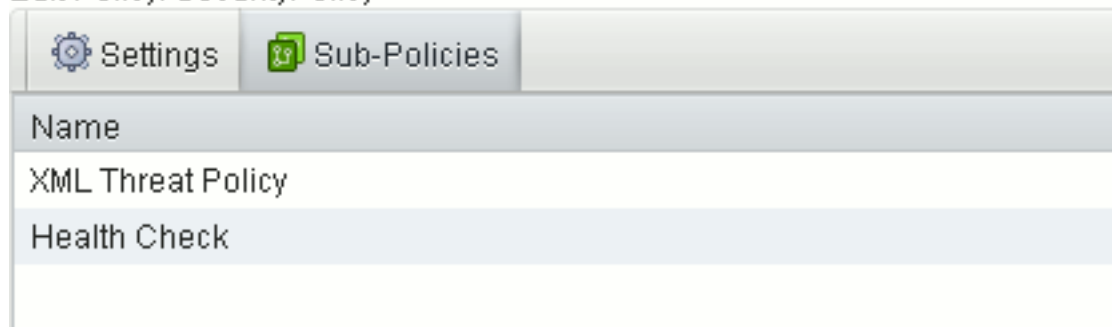
To add a new policy container, right-click an existing container, and select **Add Policy Container**. You can also select **Add Root Policy Container** to add a top-level container.

Adding a Composite Policy

To add a policy composed of existing policies, perform the following steps:

1. Select a policy container in the **Policies** screen, right-click, and select **Add Policy**. Alternatively, to add a new root policy, select **Add Root Policy**.
2. In the **Edit Policy** section, on the **Settings** tab, enter the following:
 - **Name**: the name of the policy.
 - **Category**: the category of the policy from the drop-down list.
 - **Description**: the description of the policy.
3. Click the **Sub-Policies** tab.
4. Drag and drop a policy (for example, **Health Check**) from the **Policies** tree into the table on the **Sub-Policies** tab.
5. Repeat the previous step to add more policies.
6. If wish to deploy the newly added policy to the Enterprise Gateway, click **Deploy**.

Edit Policy: SecurityPolicy



Adding a composite policy in Service Manager is equivalent to creating a *policy shortcut chain* in Policy Studio. A policy shortcut chain is a series of configured policies that run in sequence. For more details, see the [Policy Shortcut Chain](#) topic.

Removing a Sub-Policy

You can remove a sub-policy as follows:

1. In the **Sub-Policies** tab, select a policy in the table.
2. Right-click, and select **Remove Subpolicy**.

Editing a Policy

You can edit policy details as follows:

1. In the **Policies** tree, select the policy to be edited.
2. Right-click, and select **Edit Policy**.
3. In the **Edit Policy** section, make your updates on the appropriate tab.

Alternatively, you can edit a policy name or category by double-clicking the policy name in the policy tree.

Deleting a Policy or Container

You can delete a policy as follows:

1. In the **Policies** tree, select the policy or container to be deleted.
2. Right-click, and select **Delete Policy** or **Delete Policy Container**.

Configuring Security Policies from WSDL Files

Overview

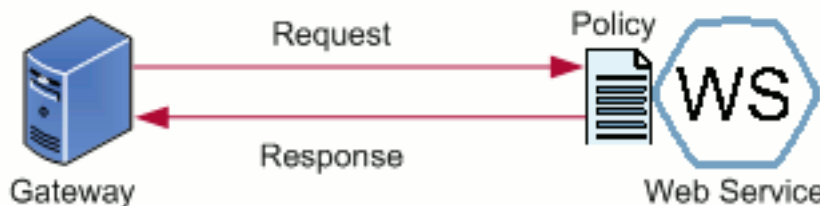
When you import a WSDL file into the Web Services Repository to virtualize and secure a protected Web Service, the Policy Studio automatically generates policy circuits. For example, a **Service Handler** is created to control and validate requests to the Web Service and responses from the Web Service. The information used to configure the Service Handler is automatically taken from the operation definitions in the WSDL.

When clients must use message-level or transport-level security mechanisms to communicate with the Web Service, you can include WS-Policy assertions in the WSDL. These policy assertions can then be referenced at the operation or endpoint level in the WSDL. For example, a given Web Service may require the client to sign sensitive parts of the message and include a WS-Security UsernameToken to authenticate to the Web Service. You can include these policy requirements in the WSDL. Whenever a client retrieves the WSDL, it can automatically sign the relevant parts of the message and insert a valid UsernameToken.

When you import a WSDL file containing WS-Policy assertions into the Web Services Repository, you can select the operations that you want to protect as normal in the **Import WSDL** wizard. The **Secure Virtual Service** dialog enables you to specify the policy that the Enterprise Gateway enforces on the messages that it receives from a client. For more details, see [Securing a Virtual Service using Policies](#).

In the **Policy Configuration Settings** wizard, you can configure specific filters to fulfill the security requirements specified by the policy assertions in the WSDL file. Most of these requirements are met without the need for human intervention. However, a small number of filters require the administrator to configure specific fields. For example, when signing or encrypting a message, you must specify the signing or encrypting key. When configuring the duration of a WS-Security Timestamp, you may need to specify longer or shorter than the default of one hour. However, most of the information required to configure these filters is set automatically based on the policy assertions in the WSDL file.

Using WS-Policy assertions, the Policy Studio can automatically generate complicated circuits that can then be used to talk to the Web Services defined in the WSDL. The Enterprise Gateway then becomes the client or *initiator* of the Web Service, and is responsible for making sure the requests it sends to the service adhere to the security constraints specified in the policy:



In this way, administrators can configure complex circuits to talk to secure Web Services with only a few clicks and minimal intervention. In addition, the Enterprise Gateway uses cryptographic acceleration to reduce the overhead associated with running the cryptographic operations required to secure the message.

Importing a WSDL file

Complete the following steps to import the WSDL file into the Web Service Repository:

1. In the Policy Studio tree view, expand the **Web Services Repository** node, and select the **Web Services** node.
2. Right-click the **Web Services** node, and select **Register Web Service**.
3. In the **Load WSDL** screen, browse to the location of the WSDL in the file system, enter the URL of the WSDL, or retrieve it from a UDDI (Universal Description, Discovery, and Integration) repository. Select as appropriate, and click

Next.

4. The operations defined in the WSDL and exposed by the Web Service are listed on the **WSDL Operations** screen. Select the operations that you want to secure, and click **Next**.
5. When you import the WSDL for this Web Service into the Enterprise Gateway's Web Services Repository, the Enterprise Gateway exposes a *virtualized* version of this service. This involves changing the host and port where the Web Service is available to point to the machine running the Enterprise Gateway. In this way, a client can retrieve the WSDL for the virtualized Web Service from the Enterprise Gateway without knowing its real location. You can also expose only the operations selected on the **WSDL Operations** screen in a slimmed down version of the Web Service. To do this, select the checkbox on the **WSDL Import Settings** screen to remove operations that you do not want to secure from the virtualized service that is exposed to clients. Click **Next** to continue.
6. Select the Relative Path from the list where you want this service to be deployed (for example, **Default Services**). Click **Finish**.

When you have completed the steps in the **Import WSDL** wizard, the **Secure Virtual Service** dialog is displayed. For details, see [Securing a Virtual Service using Policies](#).

Configuring Policy Settings

Depending on the type and number of WS-Policy assertions in the WSDL, the **Policy Configuration Settings** wizard contains configuration screens for the filters used to implement the rules required by the assertions. The exact sequence of screens differs depending on the assertions specified in the WSDL.

For example, if an `sp:Sam1Token` assertion is specified, the wizard contains a screen for the **Insert SAML Authentication Assertion** filter. Only certain fields must be specified by the administrator on this filter screen, while the rest are automatically populated based on the assertions and properties defined in the WSDL.

In the case of the **Sign Message** filter, the decision to use asymmetric or symmetric signatures is based on whether the policy uses an asymmetric or symmetric binding. The layout rules are determined by the `sp:Layout` assertion. The digest method, signature method, and key wrap algorithm (for symmetric signatures) are all populated automatically based on the contents of the `sp:AlgorithmSuite` assertion. The `KeyInfo` section of the XML Signature can be taken from various properties set in the WSDL. The parts of the message to be signed can be inferred from assertions such as `sp:SignedParts`, `sp:SignedElements`, and `SignedSupportingTokens`.

The same is true for the **XML Encryption Settings** filter where the encryption algorithms and key types can all be taken from the assertions in the WSDL. The `ConfirmationMethod` for SAML assertions can be inferred from the context of the `Sam1Token` assertion. For example, an `sp:Sam1Token` that appears as a child of the `sp:SignedSupportingTokens` assertion uses a `sender-vouches` confirmation method, whereas if it appears as a child of an `sp:EndorsingSupportingTokens` assertion, the `holder-of-key` confirmation method can be assumed.

In the **Policy Configuration Settings** wizard, the **Configure Initiator Security Settings** screen enables you to specify the required filter settings when the Enterprise Gateway is configured as the initiator for the Web Service. If the Enterprise Gateway has also been configured as the *recipient* for the client, the **Configure Recipient Security Settings** screen is then displayed. For details, see [Securing a Virtual Service using Policies](#).

Configuring Policy Filters

The following tables list the types of filters that are created, and which fields must be completed by the administrator in the **Configure Initiator Security Settings** screen. For simplicity, the tables below list only the filters that require manual input from the administrator.

Insert WS-Security Timestamp

Field Name	Description
Expires In	You may want to specify a more appropriate lifetime for the assertion (instead of the default one hour) by configuring the various time period fields.

Sign Message

<i>Field Name</i>	<i>Description</i>
Signing Key	If the policy uses an asymmetric binding, on the Asymmetric tab, click the Signing Key button, and select a key from the Certificate Store to sign the message parts with. Alternatively, if the policy specifies a symmetric binding, on the Symmetric tab, click the Signing Key button, and select a key to wrap the symmetric signing key with.

Insert WS-Security Username

<i>Field Name</i>	<i>Description</i>
Username	Enter the username inserted into the WS-Security UsernameToken block. By default, the name of the authenticated user is used, which is stored in the authentication.subject.id message attribute. However, any user-specified value can be entered in this field.
Password	If the policy requires a password, the password for the user entered above must be specified here. You can use the default authenticated user password by selecting the authentication.subject.password message attribute. Alternatively, you can enter any suitable password manually entered if necessary. The decision to use a Clear or Digest password is taken from the corresponding policy assertions.

Insert SAML Authentication Assertion

<i>Field Name</i>	<i>Description</i>
Expire In	Specify a suitable lifetime for the SAML assertion by configuring the various time period fields.
Drift Time	You may need to specify a drift time value to allow for a time differential between the clock on the machine hosting the Enterprise Gateway and the machine hosting your Web Service.
Issuer Name	Select the alias of the certificate from the Certificate Store that you want to use to identify the issuer of the assertion. The alias name is used as the value of the <code>Issuer</code> attribute of the <code>saml:Assertion</code> element.
Holder of Key: Signing Key	In cases where the <code>sp:SamlToken</code> appears as a child of <code>EndorsingSupportingTokens</code> or an <code>InitiatorToken</code> , the holder-of-key SAML confirmation method is inferred. In this case, if an asymmetric binding is used, on the Asymmetric tab, specify a key from the Certificate Store by clicking the Signing Key button. Alternatively, if a symmetric binding is used in the policy, on the Symmetric tab, specify a key to use to encrypt the

	symmetric key with by clicking the Signing Key button.
--	---

Find Recipient Certificate for Encryption

<i>Field Name</i>	<i>Description</i>
Certificate Store	Select this option, and click the Select button to choose the recipient's certificate from the Certificate Store. The public key contained in this certificate is used to encrypt the message parts so that only the recipient is able to decrypt them using the corresponding private key.

Connect to URL

<i>Field Name</i>	<i>Description</i>
Trusted Certificates	To connect to an external Web Service over SSL, you need to trust that Web Service's SSL certificate. You can do this on the Trusted Certificates tab of the Connect to URL filter. Assuming you have already imported this certificate into the Trusted Certificate Store, simply select it from the list.
Client SSL Authentication	If the Web Service requires the client to present an SSL certificate to it during the SSL handshake, you must select that certificate from the list on the Client SSL Authentication tab. Note that this certificate must have a private key associated with it that is also stored in the Certificate Store.

Extract MTOM Content

<i>Field Name</i>	<i>Description</i>
XPath Location	When the <code>wsoma:OptimizedMimeSerialization</code> WS-MTOMPolicy assertion is specified in a policy, you must configure an Extract MTOM Content filter. You need only configure an XPath expression to point to the base64-encoded element content that you want to extract and create an MTOM type attachment for.

Editing a Policy

You may wish to edit a previously configured WS-Policy (for example, to change the signing key in the auto-generated circuit). You can do this by right-clicking the Web Service in the Policy Studio tree, and selecting **Configure Initiator WS-Policy** or **Configure Recipient WS-Policy**. These menu options are described as follows:

Configure Initiator WS-Policy:

If you have already configured an initiator WS-Policy, you can edit its filters using this menu option. However, if there was no WS-Policy in the imported WSDL file, you can not use this option. You can not add a WS-Policy to the Web Service because that would break the contract between the Enterprise Gateway and the back-end Web Service. If the contract for the Web Service changes (for example, a WS-Policy is applied to it at the back-end), you need to re-import the modified WSDL to reflect the changes.

Configure Recipient WS-Policy:

If a recipient WS-Policy was configured when the WSDL file was imported into the Web Service Repository, you can edit its filters using this option. If you did not configure a WS-Policy when importing the WSDL file (using the **Secure Virtual Service** dialog), when you select this option, the **Secure Virtual Service** dialog is displayed. This enables you to select a WS-policy to secure the service. The next time that you select the **Configure Recipient WS-Policy** option, you will edit this policy.

Removing Security Tokens

When you import a WSDL file containing a WS-Policy into the Web Service Repository, the **Remove All Security Tokens** filter is enabled in the **Service Handler** for the imported Web Service. You can view the configured policy by double clicking the **Service Handler**, and selecting the **Message Interception Points -> 2. User-defined Request Hooks** tab.

The **Remove All Security Tokens** policy ensures that the following security contexts are kept separate:

- *Recipient security context:* This is between the client and the Enterprise Gateway, and is determined by the WS-Policy selected in the **Secure Virtual Service** dialog.
- *Initiator security context:* This is between the Enterprise Gateway and the back-end Web Service, and is determined by the WS-Policy contained in the imported WSDL for the back-end Web Service.

The **Remove All Security Tokens** policy prevents tokens from one context passing over into the other context, which could breach the security contract governing that context. This ensures that each security context receives a clean SOAP message, on which it can then act to enforce the security requirements of the relevant WS-Policy. The following diagram shows both security contexts and the **Remove All Security Tokens** policy:



Initiator-side WS-Policy only

In this case, the WS-Policy is contained in the imported WSDL file. The WS-Policy defines the security contract between the Enterprise Gateway and the back-end Web Service defined in the WSDL. On the request side, any security tokens sent by the client to the Enterprise Gateway, which are out of scope of the initiator WS-Policy between the Enterprise Gateway and Web Service, are removed before the Enterprise Gateway starts enforcing the initiator WS-Policy on the request, and before it sends the request to the Web Service.

For example, if the client sends a `wsu:Timestamp` in the request message and the initiator policy stipulates that a `wsu:Timestamp` must be sent by the Enterprise Gateway to the Web Service, two timestamps could be sent in the request, which is invalid. This means that the timestamp and any other security tokens sent by the client to the Enterprise Gateway, which may contradict the rules in the initiator contract (between the Enterprise Gateway and Web Service).

must be stripped out before the Enterprise Gateway starts adding security tokens to the message. This ensures that the message adheres to the initiator WS-Policy.

Similarly, any security tokens returned by the Web Service are only present because the Web Service complies with the contract between the Web Service and the Enterprise Gateway. Therefore, any tokens returned by the Web Service are only intended for use by the Enterprise Gateway. They are *not* intended for consumption by the client. In other words, the security context is only between the Enterprise Gateway and the Web Service. If the Web Service returns a `UsernameToken`, it is consumed by the Enterprise Gateway.

If a token must be returned to the client, this is a user-enforced rule, which is out of scope of the WS-Policy configuration in the WSDL. If necessary, you can override the default behavior by removing the **Remove All Security Tokens** filter from the **Service Handler** to allow the `UsernameToken` to be propagated to the client.

Initiator-side and Recipient-side WS-Policy

This occurs when you import a WSDL file that includes a WS-Policy (initiator case), and you also select a WS-Policy in the **Secure Virtual Service** dialog (recipient case). This scenario includes both the *recipient security context* between the client and the Enterprise Gateway, and the *initiator security context* between the Enterprise Gateway and the Web Service.

It is vital that these security contexts are kept separate because if tokens from one context pass over into the other context, it is highly likely that the security contract for that context will be breached. For example, if the recipient contract between the client and the Enterprise Gateway requires a `UsernameToken`, but the initiator contract between the Enterprise Gateway and the Web Service requires a SAML token, the `UsernameToken` must not pass over into the initiator context and be sent to the Web Service.

For more details on the **Secure Virtual Service** dialog (recipient case), see [Securing a Virtual Service using Policies](#).

Important Note

The **Remove All Security Tokens** policy only applies when a WS-Policy is configured, and is *not* enabled when a WS-Policy is not configured. In addition, any non-standard behavior that requires a security token to be propagated over to another security context can be handled by disabling the **Remove All Security Tokens** policy in the **Service Handler** for the imported WSDL.

Further Information

For more details on configuring policies to protect your Web Services, see the following:

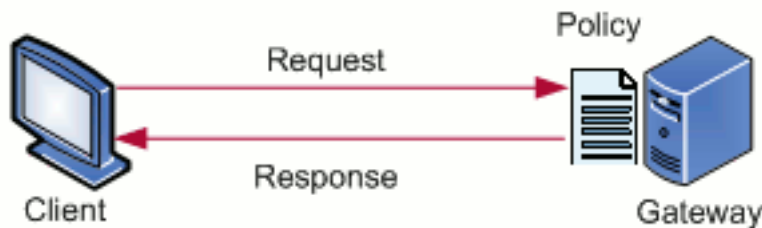
- [Securing a Virtual Service using Policies](#)
- [Configuring Policies Manually](#)
- [Web Services Filter](#)

The **Web Services** filter is the main filter generated when a WSDL file is imported into the Web Services Repository. It contains all the routing information and links all the policies that are to be run on the request and response messages into a logical flow.

Securing a Virtual Service using Policies

Overview

You can specify a WS-Policy to enforce security between a client and the Enterprise Gateway. In this deployment scenario, the Enterprise Gateway exhibits recipient-side WS-Policy behavior, where the Enterprise Gateway is the *recipient*, and the client is the *initiator*. The following architecture diagram shows where the recipient WS-Policy applies in a typical message flow between a client and the Enterprise Gateway:



When you import a WSDL file into the Web Services Repository, you can select the operations that you want to secure in the **Import WSDL** wizard. The **Secure Virtual Service** dialog then enables you to specify the policy that the Enterprise Gateway enforces on the messages that it receives from the client.

In the **Policy Configuration Settings** wizard, you can configure specific fields in the filters that are necessary to fulfill the security requirements specified in the **Secure Virtual Service** dialog. Most of these requirements are met without the need for human intervention. However, a small number of filters require the administrator to configure specific fields. For example, when signing or encrypting a message, you must specify the signing or encrypting key. When configuring the duration of a WS-Security Timestamp, you may need to specify longer or shorter than the default of one hour. However, most of the information required to configure these filters is set automatically based on the selected WS-Policy.

Using policies in this way, the Policy Studio automatically generates the complicated circuits that the Enterprise Gateway uses to talk to the client. The Enterprise Gateway then becomes the recipient of the client, and is responsible for enforcing the selected policies on the messages that it receives from the client. The main advantage is that administrators can configure complex circuits to talk to clients in a secure manner with only a few clicks and minimal intervention.

Importing a WSDL File

Complete the following steps to import the WSDL file into the Web Service Repository:

1. In the Policy Studio tree view, expand the **Web Services Repository** node, and select the **Web Services** node.
2. Right-click the **Web Services** node, and select **Register Web Service**.
3. In the **Load WSDL** screen, browse to the location of the WSDL in the file system, enter the URL of the WSDL, or retrieve it from a UDDI (Universal Description, Discovery, and Integration) repository. Select as appropriate, and click **Next**.
4. The operations defined in the WSDL and exposed by the Web Service are listed on the **WSDL Operations** screen. Select the operations that you want to secure, and click **Next**.
5. When you import the WSDL for this Web Service into the Enterprise Gateway's Web Services Repository, the Enterprise Gateway exposes a *virtualized* version of this service. This involves changing the host and port where the Web Service is available to point to the machine running the Enterprise Gateway. In this way, a client can retrieve the WSDL for the virtualized Web Service from the Enterprise Gateway without knowing its real location. You can also expose only the operations selected on the **WSDL Operations** screen in a slimmed down version of the Web Service. To do this, select the checkbox on the **WSDL Import Settings** screen to remove operations that you do not want to secure from the virtualized service that is exposed to clients. Click **Next** to continue.

6. Select the Relative Path where you want this service to be deployed (for example, **Default Services**).
7. Click **Finish**.

When you have completed the steps in the **Import WSDL** wizard, the **Secure Virtual Service** dialog is displayed.

Configuring a Security Policy

The **Secure Virtual Service** dialog enables you to specify the policy that the Enterprise Gateway enforces on the messages that it receives from the client. To specify a policy, perform the following steps:

1. In the **Secure Virtual Service** dialog, select the **Secure Virtual Service** checkbox.
2. In the **Security Policy** panel, select a **Gateway Policy** from the drop-down list. The following table shows some example policies. The full list of available policies and descriptions are displayed in the dialog.

<i>Policy</i>	<i>Description</i>
AsymmetricBinding with Encrypted UsernameToken	Service exposes an AsymmetricBinding where the client and server use their respective X.509v3 tokens to sign and encrypt the message. An encrypted UsernameToken with hash password must be included in all messages from the client to the server.
Username SupportingToken Plaintext Password	Client must authenticate with a WS-Security UsernameToken with plaintext password.
SAML 1.1 Bearer	Client must include a SAML 1.1 Assertion (bearer) representing the Requestor in all messages from the client to the service.
SymmetricBinding with Signed and Encrypted UsernameToken	Service uses a SymmetricBinding where the client and service use the same X.509v3 token to sign and encrypt the message. A signed and encrypted UsernameToken with plaintext password must be included in all messages from the client to the service. The Policy uses WSS SOAP Message Security 1.1 options.
SSL Transport Binding	Service is secured by SSL (HTTPS).
SAML 2.0 Sender-Vouches over SSL	Client includes a SAML 2.0 Assertion (sender vouches) on behalf of the Requestor to all messages from the client to the service. The service uses a TransportBinding to ensure that all messages are signed and encrypted.
WCF MutualCertificate Service	Service exposes an AsymmetricBinding protected by mutual X.509 certificates.

3. In the **Message-Level Policy** panel, select a **Request Policy** from the drop-down list. The available policies are as follows:
 - Encrypt SOAP Body
 - Sign SOAP Body
 - Sign and Encrypt SOAP Body
4. Select a **Response Policy** from the drop-down list. The available policies are the same as for **Request Policy**.
5. Click **OK**.

The **Policy Configuration Settings** wizard is displayed. This enables you to set some of the fields in the filters that require human intervention (for example, the signing and encrypting key).

Configuring Policy Settings

Depending on the policy configured in the **Secure Virtual Service** dialog, the **Policy Configuration Settings** wizard displays configuration screens for the filters that implement the rules required by the configured policy. The exact sequence of screens differs depending on the policy that is selected.

For example, if a policy with a SAML token is selected, the **Validate SAML Authentication Assertion** filter is displayed instead of the **Validate WS-Security UsernameToken** filter. The effort in configuring these screens is minimal because the information is taken automatically from the WS-Policy assertions. For example, the layout, signing, encryption, and key wrapping algorithms, key referencing method, Username digest, and clear password are all automatically taken from the WS-Policy assertions. This means that the administrator has only to configure a small number of settings. For example, the signing key, encryption certificate, and Timestamp validity period).

If your WSDL file includes WS-Policy assertions, the **Configure Initiator Security Settings** screen is first displayed in the wizard. This screen enables you to specify the required settings when the Enterprise Gateway is deployed as the initiator for the Web Service. For details, see [Configuring Security Policies from WSDL Files](#). The **Configure Recipient Security Settings** screen then enables you to specify the required settings when the Enterprise Gateway is deployed as the recipient for the client.

Configuring Policy Filters

The following tables show examples of the types of filters that are created, and which fields must be completed by the administrator in the **Configure Recipient Security Settings** screen. For simplicity, these tables list only filters that require manual input from the administrator.

Insert Timestamp Filter

<i>Field Name</i>	<i>Description</i>
Expires In	You may want to specify a more appropriate lifetime for the assertion (instead of the default one hour) by configuring the various time period fields.

Signed Parts Outbound Filter

<i>Field Name</i>	<i>Description</i>
Signing Key	If the policy uses an asymmetric binding, on the Asymmetric tab, click the Signing Key button, and select a key from the Certificate Store to sign the message parts with. Alternatively, if the policy specifies a symmetric binding, on the Symmetric tab, click the Signing Key button, and select a key to wrap the symmetric signing key with.

Find Recipient Certificate for Encryption

<i>Field Name</i>	<i>Description</i>
Certificate Store	Click the Select button to choose the recipient's certificate from the Certificate Store. The public key contained in this certificate is used to encrypt the message parts so that only the recipient is able to decrypt them using the corresponding private key.

Validate SAML Authentication Assertion

Field Name	Description
Drift Time	You may need to specify a drift time value to allow for a time differential between the clock on the machine hosting the Enterprise Gateway and the machine hosting your Web Service.
Trusted Issuers	On the Trusted Issuers tab, click Add to specify the Distinguished Name of a SAML Authority whose certificate has been added to the Certificate Store, and click OK . Repeat this step to add more SAML Authorities to the list of trusted issuers.

Configure SSL Certificate

Field Name	Description
X.509 Certificate	On the Network tab, click the X.509 Certificate button to create or import an SSL certificate.
SSL Server Name Identifier (SNI)	On the Network tab, click the Add button to configure a server name in the SSL Server Name Identifier (SNI) dialog. You can specify the server name in the Client requests server name field. Click the Server assumes identity button to import a Certificate Authority certificate into the Trusted Certificate Store.
Mutual Authentication	On the Mutual Authentication tab, select root Certificate Authorities trusted for mutual authentication.

Insert MTOM Content

Field Name	Description
XPath Location	When the <code>wsoma:OptimizedMimeSerialization</code> WS-MTOMPolicy assertion is specified in a policy, you must configure an Insert MTOM Content filter. You need only configure an XPath expression to point to the base64-encoded element content that you want to insert and create an MTOM type attachment for.

Editing a Security Policy

You may wish to edit a previously configured WS-Policy (for example, to change the signing key in the auto-generated circuit). You can do this by right-clicking the Web Service in the Policy Studio tree, and selecting **Configure Initiator WS-Policy** or **Configure Recipient WS-Policy**. These menu options are described as follows:

Configure Initiator WS-Policy:

If you have already configured an initiator WS-Policy, you can edit its filters using this menu option. However, if there was

no WS-Policy in the imported WSDL file, you can not use this option. You can not add a WS-Policy to the Web Service because that would break the contract between the Enterprise Gateway and the back-end Web Service. If the contract for the Web Service changes (for example, a WS-Policy is applied to it at the back-end), you need to re-import the modified WSDL to reflect the changes.

Configure Recipient WS-Policy:

If a recipient WS-Policy was configured when the WSDL file was imported into the Web Service Repository, you can edit its filters using this option. If you did not configure a WS-Policy when importing the WSDL file (using the **Secure Virtual Service** dialog), when you select this option, the **Secure Virtual Service** dialog is displayed. This enables you to select a WS-policy to secure the service. The next time that you select the **Configure Recipient WS-Policy** option, you will edit this policy.

Using WCF WS-Policies

The Enterprise Gateway provides four WS-Policies that are identical to those exposed by WCF (Windows Communication Foundation) Web Services. When one of these policies is exposed by a virtual service in the Enterprise Gateway, the `svcutil` .NET Web Services utility can consume the WS-Policy and auto-generate clients that communicate securely with the Enterprise Gateway.

The security settings for a WCF Web Service are configured in its `web.config` file, in which the `security` element determines the WS-Policy applied to the service. For example, the following extract from a WCF Web Service `web.config` file shows the configuration:

```
<customBinding>
  binding name="MyCustomBinding">
    <textMessageEncoding messageVersion="Soap11" />
    <security defaultAlgorithmSuite="Basic256"
      allowSerializedSigningTokenOnReply="true"
      authenticationMode="MutualCertificate" requireDerivedKeys="false"
      includeTimestamp="true" messageProtectionOrder="SignBeforeEncrypt"
      messageSecurityVersion="WSSecurity10..."
      requireSecurityContextCancellation="false">
    </security>
  </binding>
</customBinding>
```

In this example, the `authenticationMode` for a `customBinding` is set to `MutualCertificate`, which means that messages sent to and from the Web Service must be signed and encrypted with mutual certificates. The following example shows an example of the WCF `wsHttpBinding` configuration, which is less verbose:

```
<wsHttpBinding>
  <binding name="MyWsHttpBinding">
    <security mode="Message">
      <message clientCredentialType="Certificate" />
    </security>
  </binding>
</wsHttpBinding>
```

The following table shows how the WCF WS-policies provided with the Enterprise Gateway correspond to a particular configuration of the `security` element in the WCF Web Service `web.config` file. As shown in the preceding ex-

amples, the configuration settings are slightly different, depending on the WCF binding (`customBinding` or `wsHttpBinding`). The following table shows the available settings:

<i>WS-Policy Name</i>	<i>WCF Binding</i>	<i>Authentication Mode</i>	<i>Security Mode</i>	<i>Client Type</i>	<i>Credential</i>
WCF MutualCertificate Service	<code>customBinding</code>	<code>MutualCertificate</code>			
WCF UsernameForCertificate Service	<code>customBinding</code>	<code>UserNameForCertificate</code>			
WCF UsernameOverTransport Service	<code>customBinding</code>	<code>UsernameForTransport</code>			
WCF BrokeredX509Authentication Service	<code>wsHttpBinding</code>		<code>Message</code>	<code>Certificate</code>	

Important Note:

If you intend to consume the WS-Policy exposed by the Enterprise Gateway with a WCF client, you should use one of the WCF WS-Policies. All of these policies can be consumed seamlessly by the WCF `svcutil` utility to auto-generate secure clients. While the other WS-Policies exposed by the Enterprise Gateway can be consumed by `svcutil`, you need to make additional configuration changes to the auto-generated WCF client to communicate securely with the Enterprise Gateway. For more details on making any necessary configuration changes, see your WCF documentation.

Removing Security Tokens

When you configure a recipient WS-Policy in the **Secure Virtual Service** dialog, the **Remove All Security Tokens** policy is enabled in the **Service Handler** for the imported Web Service. You can view the configured policy by double clicking the **Service Handler**, and selecting the **Message Interception Points -> 2. User-defined Request Hooks** tab.

The **Remove All Security Tokens** policy ensures that the following security contexts are kept separate:

- *Recipient security context*: This is between the client and the Enterprise Gateway, and is determined by the WS-Policy selected in the **Secure Virtual Service** dialog.
- *Initiator security context*: This is between the Enterprise Gateway and the back-end Web Service, and is determined by the WS-Policy contained in the imported WSDL for the back-end Web Service.

The **Remove All Security Tokens** policy prevents tokens from one context passing over into the other context, which could breach the security contract governing that context. This ensures that each security context receives a clean SOAP message, on which it can then act to enforce the security requirements of the relevant WS-Policy. The following diagram shows both security contexts and the **Remove All Security Tokens** policy:



Recipient-side WS-Policy only

In this case, a recipient WS-Policy is configured in the **Secure Virtual Service** dialog to protect a virtual service exposed by the Enterprise Gateway. The recipient WS-Policy defines the security contract between the client and the Enterprise Gateway. Any security tokens sent by the client are intended for consumption by the Enterprise Gateway. They are *not* intended for the back-end Web Service.

For example, the Web Service may not understand SAML, WS-Security, XML Signature, and so on, which may result in a serialization error if these tokens are propagated to it. In addition, it would add unnecessary overhead to the message to propagate security tokens to it. On the response side, the response that the Enterprise Gateway returns to the client must adhere to the selected recipient WS-Policy. For example, if the Web Service has returned a SAML Token (out of scope of any WS-Policy requirements), this must not be returned to the client because it would breach the recipient WS-Policy.

Initiator-side and Recipient-side WS-Policy

This occurs when you import a WSDL file that includes a WS-Policy (initiator case), and you also select a WS-Policy in the **Secure Virtual Service** dialog (recipient case). This scenario includes both the *recipient security context* between the client and the Enterprise Gateway, and the *initiator security context* between the Enterprise Gateway and the Web Service.

It is vital that these security contexts are kept separate because if tokens from one context pass over into the other context, it is highly likely that the security contract for that context will be breached. For example, if the recipient contract between the client and the Enterprise Gateway requires a `UsernameToken`, but the initiator contract between the Enterprise Gateway and the Web Service requires a SAML token, the `UsernameToken` must not pass over into the initiator context and be sent to the Web Service.

For more details on importing WSDL files that include WS-Policies (initiator case), see [Configuring Security Policies from WSDL Files](#).

Important Note

The **Remove All Security Tokens** policy only applies when a WS-Policy is configured, and is *not* configured when a WS-Policy is not used. In addition, any non-standard behavior that requires a security token to be propagated over to another security context can be handled by disabling the **Remove All Security Tokens** policy in the **Service Handler** for the imported WSDL.

Further Information

For more details on configuring policies to protect your Web Services, see the following:

- [Configuring Security Policies from WSDL Files](#)
- [Configuring Policies Manually](#)
- [Web Services Filter](#)

The **Web Services** filter is the main filter generated when a WSDL file is imported into the Web Services Repository. It contains all the routing information and links all the policies that are to be run on the request and response messages into a logical flow.

Configuring Policies Manually

Overview

In cases where a Web Services definition file is not available in a WSDL file (Web Services Description Language), the policy used to protect a Web Service must be configured manually. The steps outlined in this tutorial describe how to do this.

However, that the recommended way to configure a policy to protect a Web Service is to import the WSDL file for that service. If WS-Policy information is contained in the WSDL file, the policy assertions can also be used to produce a complex policy with minimum effort for administrators.

If your Web Service has WSDL-based definitions, see the following:

- [Securing a Virtual Service using Policies](#)
- [Configuring Security Policies from WSDL Files](#)

Configuration

The following steps outline how to manually create a policy to protect a Web Service and then test it.

Step 1: Create the Policy

To create a policy manually, complete the following simple steps:

1. Right-click the **Policies** node in the tree view of the Policy Studio, and select the **Add Policy** menu option.
2. Enter a suitable name (e.g. "New") for the new policy in the **Name** field and click the **OK** button. The new policy will now be visible in the tree view.
3. Click the new policy in the tree view to start configuring the filters for the new policy. You can easily configure the policy by dragging the required filters from the filter palette on the right-hand side of the Policy Studio and dropping them onto the circuit canvas.
4. Most policies will attempt to check characteristics of the message, such as message size and format, as well as attempting to authenticate and/or authorize the sender of the message. Once the message successfully passes all configured filters it is usually routed on to the protected Web Service.
5. For demonstration purposes we will create a very simple policy consisting of 2 filters. The first filter will check the size of the message and the second will echo the request message back to the client if it is below a certain size.
6. Expand the **Content Filtering** category of filters from the filter palette and drag and drop the **Message Size** filter on to the canvas.
7. Enter "10" in the **At least** field and "1000" in the **At most** field to make sure that only messages between 10 bytes and 1000 bytes will be reflected back to the client. Select all other defaults and click the **Finish** button.
8. Right-click on the newly added filter and select the **Set as Start** menu option to indicate that this is the 1st filter to be executed in this policy. The icon for the filter will change to indicate that it is the "Start" of the policy.
9. Now open the **Utilities** category of filters and drag the **Reflect** filter onto the canvas. This time drop it onto the previously configured **Message Size** filter. Select the defaults for the **Reflect** filter and click the **Finish** button.
10. Because you dropped the **Reflect** filter on to the **Message Size** filter, both filters are automatically linked with a *success path*. This means that if the first filter runs successfully, the next filter on the success path is executed. To link in more filters, add the filters to the canvas, and click the **Success Path** button at the top of the palette. Click the first filter followed by the second filter in the success path to link both filters.
11. You can also configure *failure paths* for filters in the same way. Failure paths are followed when the checks configured in the filter fail.

This completes the configuration of the simple policy.

Step 2: Create a New Relative Path

You must now create a **Relative Path** on the Oracle Enterprise Gateway Process, which will map incoming requests on a particular URI to the new policy. Complete the following steps to do this:

1. Right-click the "Default Services" node in the tree view of the Policy Studio, which can be found under the "Oracle Enterprise Gateway" node under the "Processes" node. Select the **Add Relative Path** menu option.
2. On the **Configure Relative Path** dialog, enter a suitable URI (for example, "/New") on which you want to receive requests that are to be processed by the new policy.
3. To map requests received on this URI to our new policy, you simply need to select the "New" policy from the list of policies in the tree. Click the **OK** button when you have done this.

Step 3: Deploy to the Enterprise Gateway

Before the new configuration changes can take effect, you must deploy them to the Enterprise Gateway. You can do this by simply by selecting the F6 button. It is important to note that you *must* deploy to the server after making configuration changes.

Important Note:

When deploying to the Enterprise Gateway, the Policy Studio sends a deployment request to the server. If necessary, you can configure the socket timeout value for this connection in the Policy Studio **Preferences** dialog. Enter the timeout value in milliseconds in the **Server Socket Connection Timeout** field. For more details, see [Policy Studio Preferences](#).

Step 4: Test the Policy

You can use the tool of your choice (e.g. Oracle Service Explorer) to send SOAP requests to the new policy. You should send requests of different sizes to the following URL, assuming a default installation of the Enterprise Gateway running on the local machine:

`http://localhost:8080/New`

You will notice that requests message falling between the configured size will be reflected to the client, but that those that fall outside of the configured will be "blocked" and a SOAP Fault will be returned to the client.

Step 5: Next Steps

Try running more complicated checks on request messages by adding new filters to the "New" policy. Try also adding "failure paths" to the original **Message Size** filter to handle messages that fall outside of the 10-1000 byte range.

Use the **Help** button on each filter screen to find out more about the configuration fields that are available on each screen.

Configuring Global Policies

Overview

Global policies enable you to label policies with specific roles in the Enterprise Gateway configuration. For example, you can label a specific policy such as **XML Threat Policy** as a **Global Request policy**. This policy can be executed globally on the request path for all messages passing through the Enterprise Gateway. Using a global policy in this way enables you to use the same policy on all requests, and for multiple services. It also means that you can change the labeled global policy to a different policy without needing to rewire any existing policy circuits.

For example, using a **Policy Shortcut Chain** filter in a policy enables you to delegate to one or more policies to perform specific tasks, before continuing execution of the remaining filters in the current policy. Using this approach to encapsulate specific functionality in a policy facilitates modularity and reusability when designing Enterprise Gateway circuits. This enables you to build up a policy library of reusable routines over time.

Each shortcut in a **Policy Shortcut Chain** points to a specific policy, which is called at each point in the execution chain. However, consider a policy whose role is to be called first in all message handling contexts before any context-specific policies are run, and call this the run-first role. To realize this, you must create a **Policy Shortcut Chain** with a link to the run-first policy as its first entry, the context-specific policy as its second link, and so on.

One of the shortcomings of this approach is that if you have set up a large number of **Policy Shortcut Chain** filters, each calling the run-first policy, and you need to change the run-first policy globally, you must update each **Policy Shortcut Chain** filter individually to point to the newly designated run-first policy. Similarly, if you wish to ignore the run-first Policy globally, you must remove the first entry in each filter.

Global policies enable you to label a specific policy in terms of its role. You can delegate to the policy using its label instead of a specific link to the policy. This indirection using a label makes it very easy to globally change which policy is delegated to, merely by moving the label from one policy to another. Each filter that refers to the policy using its label now resolves the label to the new policy without needing to change the filter configuration. Similarly, if the label is not applied to a specific policy, nothing is executed for this link.

Global Policy Roles

The following global policy roles have a reserved label and a specific meaning in the Enterprise Gateway policy framework:

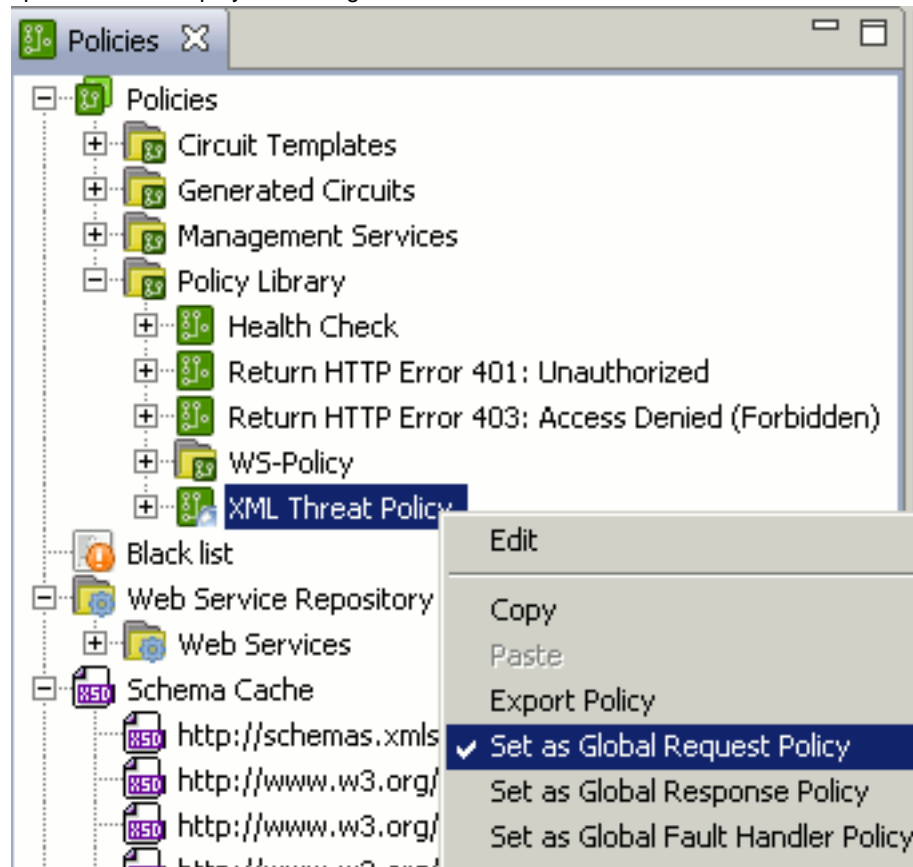
<i>Role</i>	<i>Label</i>	<i>Description</i>
Global Request Policy	<code>system.policy.request</code>	Executed globally for all messages passing through the Enterprise Gateway on the request path.
Global Response Policy	<code>system.policy.response</code>	Executed globally for all messages passing through the Enterprise Gateway on the response path.
Global Fault Handler Policy	<code>system.policy.faulthandler</code>	If any policy aborts during execution, or a top-level policy fails and has not specified a Fault Handler filter, this policy is executed instead of the internal SOAP Fault filter.

You can select specific policies with these roles on the **Policies** tab in the Policy Studio, and then create links to these roles when creating a **Policy Shortcut Chain**. These steps are explained in the next sections.

Selecting a Global Policy

To select a global policy, right-click a policy on the **Policies** tab, and select one or more global policies (for example, **Set as Global Request Policy**, **Set as Global Response Policy**, or **Set as Global Fault Handler Policy**). These policies are executed globally for all messages passing through the Enterprise Gateway.

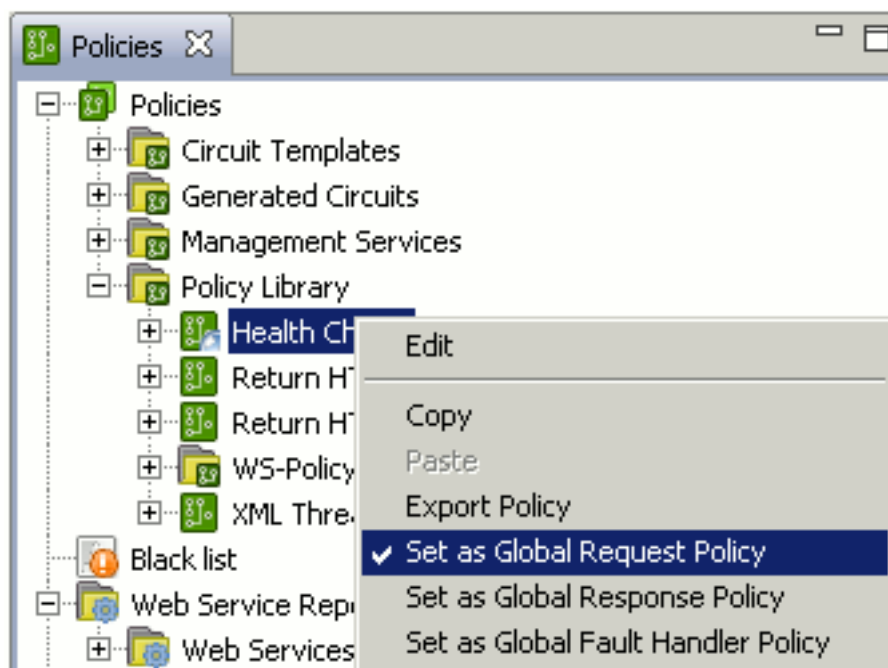
The following example shows the **XML Threat Policy** set as the **Global Request Policy**. The policy node labeled for the specific role is displayed with a globe icon:



When you have selected the policy for a specific role, you can then reference the labeled policy in a **Policy Shortcut Chain** filter, or at the service level in a Relative Path or Web Service Resolver. Referencing a labeled policy is different from referencing a specific policy directly. Referencing a policy directly involves selecting a specific policy to execute in the chain. Referencing a labeled policy means selecting a filter by its label only.

The main advantage of this approach is that you can configure a policy to run in a policy shortcut chain in a specific role, and then select a different policy as the global policy for that role. All references to the global policy label in the various shortcut chain filters are changed to use the newly selected policy, without requiring you to modify each policy shortcut chain filter individually to explicitly point to a different policy.

Selecting another policy in a global role unselects the previously selected policy. The following example shows the **Health Check** set in the global role, and the **XML Threat Policy** policy is no longer selected:

**Important Note:**

You can not select a policy for a specific role if, in doing so, you create a loop in the policies. For example, if a **Policy Shortcut Chain** filter has a reference to a labeled policy, and the filter's parent policy is marked as the labeled policy, the filter would call back to itself in a loop. This error is caught, and a trace line is output to the Policy Studio **Console** view.

Configuring Global Policies in a Policy Shortcut Chain

When adding a policy shortcut in a **Policy Shortcut Chain** filter, you can select to call a labeled policy instead of selecting a specific policy. The following example from the **Add a new Shortcut to a Policy** dialog shows adding a shortcut to the **Global Request Policy (Health Check)** policy label:

Shortcut Label

☒ Evaluate this shortcut when executing the chain

☐ Choose a specific Policy to execute:

Policy

☒ Choose a Policy to execute by label:

Policy Label

Then if you select a different policy as the request policy on the **Policies** tab, when you subsequently view this shortcut in the chain filter, you see that the details for the shortcut have changed. The following example from the **Edit the Shortcut to the Policy** dialog shows the policy label changed to **Global Request Policy (XML Threat Policy)**.

For more details on configuring these screens, see the [Policy Shortcut Chain](#) filter.

Important Note:

If you remove a label from a policy by unselecting it on the **Policies** tab, any reference to that labeled policy is not called when evaluating the shortcut in the chain, irrespective of whether the **Evaluate this shortcut when executing the chain** checkbox is selected (the **Active** status column in the table view). This corresponds with the behavior for a specific policy in the chain. If a link to a policy is not set for a shortcut, the link is not evaluated.

In this example, the table shows that the shortcut is configured to point to the labeled policy, but the label does not resolve to a policy (for example, it is unspecified because there is no policy in the specified role):

Policy Shortcut Chain

Configure a list of Policies to be called in successful sequence.

Name: Policy Shortcut Chain		
Active	Label	Policy to Execute
Yes	Test Shortcut	Gateway request policy (<Unspecified>)

Configuring Global Policies for a Service

On the **Services** tab, you can also configure global policies at the service level to run on a specific Relative Path or Web Service Resolver when messages are received by the Enterprise Gateway. A Relative Path binds a policy to a specific relative path location (for example `/healthcheck`). A Web Service Resolver maps messages destined for a specific Web Service to a **Service Handler** or **Web Service Filter**.

You can configure a global policy at the service level to run as part of a policy chain invoked when incoming messages are received by the Enterprise Gateway. The following example shows the **Global Request Policy** configured to run first on the `/healthcheck` relative path:

☒ Enable this path resolver

Policies | Audit Settings

When a request arrives that matches the path:

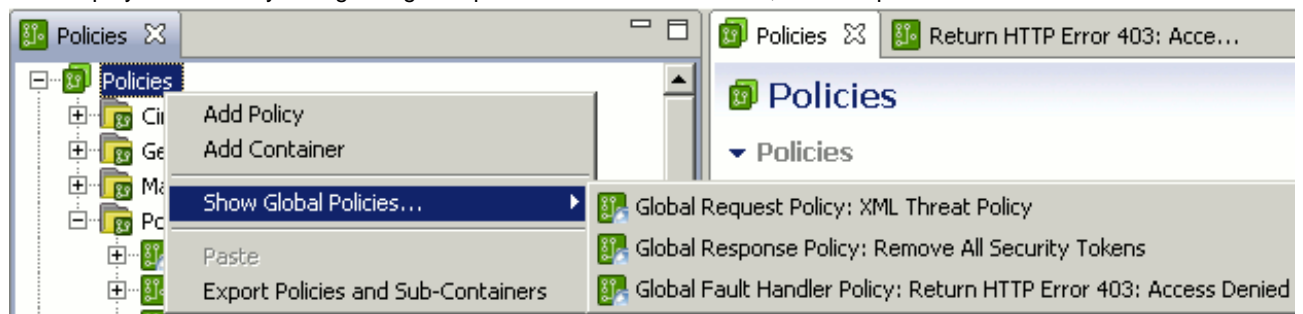
Call the following Policies:

- ☒ Global Request Policy
- ☒ Path Specific Policy: ...
- ☐ Global Response Policy

For more details on how to configure a global policy for a service, see the [Relative Paths](#) and [Web Service Resolvers](#) sections in the [Configuring HTTP Services](#) topic.

Showing Global Policies

To view the currently configured global policies, right-click the **Policies** root tree node, and select **Show Global Policies**. This displays all currently configured global policies in the context menu, for example:



Note:

If there are no global policies configured, the **Show Global Policies** menu item is not available.

Introduction to Policy Management

Overview

The ever escalating increase in message traffic results in multiple instances of Enterprise Gateways across the network. For example, Enterprise Gateways can be deployed inside the DMZ, at the data center for acceleration and application offload to relieve key processing bottlenecks, and in front of key applications to identify service usage and behavior. To overcome potential operational control issues, you need policy management tools for centralized control.

Policy Studio and Policy Center address the need for centralized policy management across a distributed network of Enterprise Gateways. These tools enable enterprises to control a distributed network of Enterprise Gateways from a single location. By creating modular reusable policies using Policy Studio, organizations can cluster, version, reuse, and migrate policies to meet high-level security, offloading, and monitoring requirements across their entire SOA. Using a Policy Center server, you can then deploy modified configuration to any Process managed by Policy Studio.

Policy Management Features

This section provides a high-level overview of the policy management features available in Policy Studio and Policy Center:

Avoid the Island Mentality

Policy Studio ensures that Oracle customers do not have to manage a group of isolated policy islands, each in an individual Enterprise Gateway. Instead, when connected to a Policy Center server, you can use Policy Studio's policy management features to centrally manage policies deployed on different Enterprise Gateways.

Centralized SOA Policy Creation

Policies contain specific assertions about operational attributes such as authentication and authorization, encryption and signatures, routing, transformations, and versioning. Policy Studio enables policies to be created, and versioned. When connected to a Policy Center server, you can then push out this configuration to Enterprise Gateways that have been deployed throughout the organization.

Lifecycle Policy Management

Policies managed by Policy Studio are readily transferable across multiple Enterprise Gateways. This enables policies to be developed in a testing environment, and then be proofed in a staging environment designed to replicate production systems, prior to going into production. Using a Policy Center server, policies can be easily migrated between testing Enterprise Gateways, staging machines, and production Enterprise Gateways.

Policy Rollback

Using Policy Studio, policies can be rolled back to revert to previous policy versions. This is a safeguard for policy updates. It also enables systems to be restored to policy versions that were in place at a certain date for testing or auditing purposes.

Role-based Policy Control

Policy Studio administrators are assigned preferential access to policies, including read, write, and update permissions.

Centralized Logging

Metrics on Web Services usage across multiple Enterprise Gateways are centrally stored by Policy Center. This centralized logging allows reports to be generated by Service Monitor. This enables all application networking activity to be viewed together in centralized Web-based reports.

Fast Provisioning of XML Networking Infrastructure

Policy Studio policy management features enable new instances of Enterprise Gateways to be created and deployed quickly. This is particularly relevant for VMWare-based deployments of software-based Enterprise Gateways.

Getting Started with Managing Deployments

Overview

Administrators can use Policy Studio to specify different configuration versions. When connected to a Policy Center server, you can deploy configurations to different Enterprise Gateway servers running in the network. In Policy Studio, the **Oracle Enterprise Gateway Dashboard** tab enables you to edit the configuration of currently running Enterprise Gateway server instances, which are called *Processes*. You can then update the downloaded configuration, and commit it to the server, where it can be reloaded later. When connected to a Policy Center server, you can deploy modified configuration to multiple Processes managed by Policy Studio. Managed Processes can be Enterprise Gateway, Policy Center, and Service Monitor server instances.

In this way, Policy Studio and Policy Center play a key role in an operational Service Oriented Architecture (SOA). They enable administrators to centrally manage the policies that are enforced at all nodes throughout the SOA. In addition, Policy Studio enables administrators to compare and merge differences between versions of the same policy. Policies can be merged, and deployed to any running Process that is managed by Policy Studio.

One of the most powerful uses of this centralized management capability is in transitioning from a staging environment to a production environment. For example, policies can be developed and tested on the staging environment, and when ready, they can be deployed to all Processes deployed in the production environment.

Connecting to a Server

Before starting Policy Studio, you should first ensure that the server process that you wish to connect to has been started (for example, Enterprise Gateway, Policy Center, or Service Monitor). For details on starting Policy Studio and the Enterprise Gateway, see [Startup Instructions](#).

When the Policy Studio starts up, click a link to a server to display the **Open Connection** dialog. You can use this dialog to specify **Connection Details** (for example, host, port, user name, and password) or to specify **Saved Sessions**. If you wish to connect to the server using a non-default URL, click **Advanced**, and enter the **URL**. The default Enterprise Gateway server URL is:

```
http://localhost:8090/configuration/deployments/DeploymentService
```

Alternatively, you can connect to a server configuration file by clicking the **Open File** button. For more details on connecting using a server URL or configuration file, see [Connection Details](#).

Managing Multiple Servers

To manage multiple Enterprise Gateways in your network, you must connect to a Policy Center server. The default Policy Center server URL is:

```
http://localhost:8060/configuration/deployments/DeploymentService
```

When the connection to the server has been made, the **Oracle Enterprise Gateway Dashboard** tab is displayed. This displays the list of processes currently managed by the Policy Studio, and enables you to manage the active configuration for a Process.

Editing an Active Server Configuration

The **Oracle Enterprise Gateway Dashboard** tab lists all available processes on the left, and displays summary information on the right. Double-click a process name in the **Process List** tree to load its active configuration. Alternatively, right-click a process name, and select **Edit Active Configuration**, or click the link on the **Summary** tab. The active server process configuration is loaded and displayed in a tab named in the following format: **ProcessName [MachineName]** (for example, **Oracle Enterprise Gateway [ronaldo]**).

When an active server configuration is loaded, its Processes and Services are displayed in the **Services** tab on the left. Click one of the buttons on the left to display additional details (for example, **Policies**, **External Connections**, **Users** or **Certificates**).

When editing an active server configuration, you can deploy updates using the **Deploy** button in the toolbar (alternatively, press **F6**). You can version deployed updates using the **Version** button in the toolbar (alternatively, press **F7**).

Managing Processes

Each Process displayed in the list in the **Oracle Enterprise Gateway Dashboard** represents a running Enterprise Gateway, Policy Center, or Service Monitor. The tabs on the right display more detailed information about a deployed process. This includes summary details (for example, process name, host, status, and owner), deployed configuration, and connection details.

You can also perform tasks such as adding or removing a process, changing a process owner, updating connection details, and versioning process configuration. For more information, see [Managing Processes](#).

Managing Deployed Configuration

The **Oracle Enterprise Gateway Dashboard** also enables you to manage the component configuration stores that are currently deployed on a selected process. These include the configuration stores for core policies and services, external connections, users, and certificates. Click the **Deployment Details** tab on the right to manage the currently deployed Configuration Profiles.

For example, you can select a process and view the versions of the component configuration stores that are currently running on that process. You can also change the currently deployed versions, and deploy your updates to the Enterprise Gateway server. For more details, see [Managing Configuration Profiles](#).

Managing Configuration Versions

You can manage configuration versions by selecting **Window -> Show View -> Profile Repository** from the main menu. Alternatively, right-click a component configuration store on the **Deployment Details** tab, and select **View in Profile Repository**. The **Profile Repository** tab displays all available configuration versions for particular component stores. For example, the list of available Core Configurations depends on what versions of policies and services have been committed to the server. Similarly, this view depends on the currently logged in user.

The **Profile Repository** tab enables you to perform tasks such as the following:

- Edit a specific configuration version
- Create a new configuration version
- Commit a configuration version to the server
- Rollback a configuration version to a previous version
- Compare and merge specific configuration versions
- Rename a configuration version
- Change the owner of a configuration version

For more details, see [Managing Configuration Profiles](#).

Editing Configuration Profiles

On the **Profile Repository** tab, you can right-click a specific Configuration Profile under the **My Configuration Profiles** tree node, and select **Edit**. The selected configuration version is loaded into the **Profile Editor** tab below. Double-click a node in the **Profile Editor** tab to view all configuration items for the selected component configuration version on the right (for example, Certificate Store or Core Configuration).

You can modify the loaded configuration if necessary, and commit it to the server. For more details, see [Managing Configuration Profiles](#).

Comparing and Merging Configurations

On the **Profile Repository** tab, you can select specific component Configuration versions and compare them. Both versions are then loaded and compared, and the results are displayed below in the **Compare and Merge** tab.

For example, you can view the differences made to particular fields in an Authentication filter that occurs in both versions. When a difference like this is located, you can merge the differences. For example, you can update the fields in the Authentication filter in one version with the fields configured for the same Authentication filter in the other version.

For more information, see [Comparing and Merging Configurations](#).

Managing Policy Studio Users

You can now add users to enable role-based access to the configuration information managed by Policy Studio. **Super Users** can view and modify any Process or Configuration, whereas normal Users can only view and modify Processes and Configurations that they create.

To add or remove users, click the **Users** button at the top left of the **Oracle Enterprise Gateway Dashboard**. For more details, see [Managing Users](#).

Configuring Policies

You can use Policy Studio to manage versions of policy Configurations, which can then be deployed to running instances of Oracle Enterprise Gateways. For more information on configuring specific message filters, see [Policy Configuration](#). For information on more general configuration topics, see [General Configuration](#).

Managing Processes

Overview

You can manage Oracle server processes using the **Oracle Enterprise Gateway Dashboard**. Managed processes can include Enterprise Gateway, Policy Center, and Service Monitor processes. For example, you can view process details, change a process owner, or edit its connection details. In addition, if Policy Studio is connected to a Policy Center server, you can add and remove multiple Enterprise Gateways in the managed **Processes** list.

Important Note:

To display the Policy Center server process in the Policy Studio, you must first enable the **Show Management Services** checkbox in the Policy Studio **Preferences** dialog. In the Policy Studio main menu, select **Windows -> Preferences**. For more details, see [Policy Studio Preferences](#).

Viewing Process Details

In the **Oracle Enterprise Gateway Dashboard**, when you select a process in the list on the left, the process details are displayed in the following tabs on the right:

Summary	Includes the process name, host, process status (for example, Up or Down), configuration status (for example, Deployed or Out of Sync), version, last changed by, and owner.
Deployment Details	Displays the component configuration stores that are deployed on the process (for example, Core Configuration, User Store, and Certificate Store). Details include revision number, owner, last updated by, and log entry.
Connection Details	Includes the URL, user name, and password for the process (for example, the default URL for the Enterprise Gateway process is <code>http://localhost:8090/runtime/management/ManagementAgent</code>).

Adding a New Process

If Policy Studio is connected to the Policy Center server, you can add new processes to the list of deployed processes. You can not add new processes if Policy Studio is connected to the Enterprise Gateway or Service Monitor server.

When connected to a Policy Center server, to add a new process deployment, perform the following steps:

Host	Specify the host to connect to in this field. The default is <code>localhost</code> .
Port	Specify the port to connect on in this field. The default Enterprise Gateway port is 8090. The default Enterprise Gateway port is 8060.
Username	Policy Studio must authenticate to the Process using HTTP basic authentication. Specify the user name for this purpose.
Password	Specify the password for this user.

1. Click the **Add Process** button at the top right of the **Process List**.
2. In the **Add a New Process** dialog, complete the following fields:
3. If you wish to explicitly set the URL on which the process is managed, click **Advanced**, and enter the URL for the process. The default URL for the Enterprise Gateway process is `http://HOST:8090/runtime/management/ManagementAgent`, where `HOST` is the IP address or hostname on which the Process is running. For more details, see [Show Management Services](#).
4. Click **OK**.

Removing a Process

You can remove a process from the list of deployed processes. To remove a process, perform the following steps:

1. Select the process in the **Process List**.
2. Right-click, and select **Remove Process**.
3. Click **Yes**.

This removes the process from the **Process List** permanently. The process is no longer contacted, and continues to run with its most recently deployed configuration.

Changing a Process Owner

You can transfer ownership of a process to another user. To change a process owner, perform the following steps:

1. On the **Summary** tab, beside the **Owner** field, click **Change**.
2. In the **Change Owner** dialog, select a user from the drop-down list.
3. Click **OK**.

Editing Connection Details

You can update the deployment URL and the user credentials used by the Enterprise Gateway to authenticate to the process. To edit the connection details, perform the following steps:

1. On the **Connection** tab, specify your updates in the appropriate fields. For example, you might need to update the process URL if you are using a different management port instead of the default 8090. For details on the fields in the **Connection** tab, see [Adding a New Process](#).
2. Click **Save**.

Versioning Process Configuration

If you have edited an active process configuration, and deployed updates to the server without versioning, the **Configuration Status** for the selected process is displayed as **Out of Sync** on the **Summary** tab.

To version the configuration of a selected process, perform the following steps:

1. Click the **Version Configuration** link on the right of the **Summary** tab. The **Versioning** dialog displays the list of **Modified Components** that are out of sync on the left (for example, Listener Configuration or Core Configuration).
2. Enter a **Comment** to version the specified components in the text area on the right.
3. You can select the relevant option to apply this comment to all modified components (the default), selected modified components, or all components.
4. Click **Yes** to finish.

The **Configuration Status** for the selected process is now displayed as **Deployed** on the **Summary** tab, and the **Version Configuration** link is no longer displayed. On the **Deployment Details** tab, the **Revision** number of the components that were out of sync is updated to the latest version, and the **Comment** entered in the **Versioning** dialog is also displayed.

When editing an active process configuration, you can also version deployed updates directly by clicking the **Version** button in the toolbar (alternatively, press **F7**). This displays the **Versioning** dialog, which enables you to version the selected components.

Deploying Configuration to a Process

For details on how to deploy Configuration Profiles to processes, see [Managing Configuration Profiles](#).

Managing Configuration Profiles

Overview

A Configuration Profile is a store of configuration information required to run a server (Enterprise Gateway, Service Monitor, or Policy Center). For example, a specific Enterprise Gateway Configuration Profile can store certificates, users, core policies and services, external connections, or listeners.

When you load a Configuration Profile to edit it, a copy is loaded into a Policy Studio workspace (for example, when you edit policies, the changes are saved locally). When you commit your updates, the local Configuration Profile is committed to the server, and a new Version of the Configuration Profile is created and labeled with the comment specified when committing.

A Configuration Profile can have a number of Versions, one per user commit. A newly created Configuration Profile has one Version. A Version is read-only in the Policy Studio. When you load a Configuration Profile to edit to it, the latest Version is returned to the Policy Studio. Committed updates to the Configuration Profile are saved in a new Version.

Managing Deployed Configuration Profiles

The **Deployment Details** tab on the **Oracle Enterprise Gateway Dashboard** enables you to manage the currently deployed Configuration Profiles on a selected server process (Enterprise Gateway, Service Monitor, or Policy Center). For example, these include the component configuration stores for core policies and services, external connections, users, and certificates.

You can select a process to view the versions of the component configuration stores that are currently running on that process. You can also change the currently deployed Configuration Profile versions, and deploy updates to a server process. You can specify and deploy Configuration Profile versions on the selected process, on a different process, or on multiple processes.

Deploying Configuration Profile Versions

To change a currently deployed Configuration Profile or Version on the currently selected process, perform the following steps:

1. Select a component configuration store from the table (for example, `Certificate Store`, or `External Connections`). The available Profiles and Versions for this configuration store are displayed in the panel at the bottom of the screen (for example, `Type: Certificate Store`).
2. Select the **Profile** that you wish to deploy in the drop-down list (for example, `Oracle Enterprise Gateway(version) - Default Certificate Store`). This enables you to deploy Configuration Profile versions currently deployed on other processes. The list of Versions available for this Profile is displayed in the table below. The configuration store to be updated is highlighted and marked with an asterisk in the table above.
3. Select the **Active** checkbox for the **Revision** or version that you wish to deploy (for example, 1).
4. Click the **Deploy** button at the bottom right. Alternatively, to cancel at any time before deploying, right-click, and select **Refresh**.

Deploying Configuration Profiles to Multiple Processes

When connected to a Policy Center server, you can use the **Deployment Wizard** to deploy Configuration Profiles to multiple processes. To deploy to one or more processes, perform the following steps:

1. Click the **Deployment Wizard** button at the top of the **Process List** on the left to display the **Configuration Profiles** screen.
2. In the **Type** column, select one or more Configuration Profile types that you wish to deploy (for example `Certificate Store`, `External Connections`, or `Core Configuration`).
3. In the **Profile** column, select the Configuration Profile that you wish to deploy from the list (for example `OracleEn-`

- terprise Gatewayversion) - Default Certificate Store).
4. In the **Revision** column, select the Configuration Profile version that you wish to deploy from the list (for example, 1).
 5. Click **Next** to display the **Process List** dialog.
 6. Select one or more processes to which to deploy the selected Configuration Profiles.
 7. Click **Next**, and check the deployment details in the **Deployment Summary**.
 8. Click **Finish**.

Managing All Configuration Profiles

You can select **Window -> Show View -> Profile Repository** from the main menu to manage all the Configuration Profile versions available to the currently logged in User. The **Profile Repository** tab displays all Configuration Profiles managed by the Policy Studio in a tree view. This includes a sub-node for each User's set of Configuration Profiles.

A User can only update the Configuration Profiles that they own (listed in **My Configuration Profiles**). Super Users can modify all User Configuration Profiles as if they were the owner. An update means internal configuration information (for example, a policy change) is updated, added, or removed. Each Configuration Profile has only one owner. In addition, a User can copy another User's Configuration Profile Version to create a new Configuration Profile that they can then load into the **Profile Editor** for editing.

Managing My Configuration Profiles

The **My Configuration Profiles** node on the **Profile Repository** tab shows the Configuration Profiles that the current user has ownership of. The following configuration options are available when you right-click the **My Configuration Profiles** node:

Create Baseline Configuration:

Creates a set of component store configurations using a default template. Enter a **Name prefix** for the new Baseline Configuration Profile (for example `Test`), and select a currently deployed **Process** from the drop-down list. Processes are listed in the following format: **MachineName:ProcessName** (for example, `cayote:Oracle Enterprise Gateway`). The new Baseline Configuration includes appropriately named nodes for each component store configuration (for example, Test Core Configurations, Test Users, Test Certificates, Test Listeners, and Test External Connections).

You can also enter an optional **Comment** and **Description**. The new nodes are added under the **My Configuration Profiles** node, indicating that the User that created the new set of Configuration Profiles is its owner.

Create Configuration Profile:

Creates a new component store configuration using a template. Enter a **Name** for the new configuration store, and select a currently deployed **Process** and **Type** from the drop-down lists. Processes are listed in the following format: **Machine-Name:ProcessName** (for example, `cayote:Oracle Enterprise Gateway`). Example Types include Core Configurations, Users, Certificates, Listeners, and External Connections.

You can also enter an optional **Comment** and **Description**. A new node is added under the **My Configuration Profiles** node, indicating that the User that created the new component configuration store is its owner.

Create Configuration via Import:

Imports a full configuration store (for example, Core Configurations, Users, Certificates, and so on). In addition to the fields specified in the **Create Configuration Profile** dialog, you must also select a configuration file to import from. A new node is added to the **My Configuration Profiles** node.

Change Owner:

If you select this option, you are presented with a list of all other Users. When you select a new User, the ownership of the currently selected Configuration Profile is changed to the new User.

Important Note: The Super User can perform all options on any User's Configuration Profile node.

Managing a Component Store

A specific Configuration Profile node corresponds to an instance of a complete component store configuration (for example, **Default Certificate Store**). The set of Versions for that Configuration Profile that have been saved over time are listed under the component store node. You can access the following configuration options by right-clicking a component store node or its most recent Version node in the tree:

Edit:

Opens this component store in the **Profile Editor** for editing so that its policies, certificates, and so on, can be updated. The latest Version is opened when this option is selected on the component store node. All updates applied are saved to a local copy of the configuration file. This option is available if more than one component store is selected in the tree (or table) with the result that all selected component stores are simultaneously loaded into the **Profile Editor**.

Commit Version:

This option is only enabled after the component store configuration is loaded in the **Profile Editor**. When you commit a Configuration Profile, you are asked to enter a Comment. When you commit your changes, the local Configuration Profile is returned to the currently managed server (for example, Enterprise Gateway), and a new Version is created. A new Version node is added under this configuration node. You can also use the **Commit Version** menu option in the **Profile Editor** tab from the top-level node for that Configuration Profile.

If you make local changes that are not committed, you are asked if you wish to load your local version or the server version the next time you load the Configuration Profile. This enables you to retrieve local changes if your session times out before you commit the changes.

Create via Copy:

A copy of the latest Configuration Profile Version is taken and used to create a new Configuration Profile. You must enter a name for the Configuration Profile. A new Configuration Profile node is displayed under **My Configuration Profiles**.

Rename:

You can rename the Configuration Profile by specifying a new name in the **Rename Configuration Profile** dialog.

Change Owner:

You can change the ownership of the Configuration Profile. When ownership changes, the configuration node moves into the other User's **My Configuration Profiles** node. This option is available if more than one Configuration Profile is selected in the tree (or table).

Export:

You are asked to enter a filename to export to. The latest Version of the Configuration Profile is exported to this file, which you can then import into a different Enterprise Gateway configuration. For example, this is useful if you have configured the Enterprise Gateway in a testing environment, and want to move this configuration to a live production environment.

Archive:

Archives the Configuration Profile and all of its related Versions. The configuration node is removed from the tree. This option is only available if none of the Versions related to this Configuration Profile are deployed. Note that you can not delete a Configuration Profile.

When the Policy Studio receives a request to archive a Configuration Profile, the underlying files that hold the Configuration Profile and its Versions are moved into the `INSTALL_DIR/conf/archive` directory on the Policy Studio machine. The underlying configuration data used by the Policy Studio is deleted. The system administrator must determine whether to remove these files from disk and store them elsewhere for backup purposes. If there is a requirement to reload an archived Configuration Profile or one of its Versions, the archived file must be located manually and re-imported to create a new Configuration Profile. This option is also available if more than one Configuration Profile is selected in the tree (or table).

Refresh:

Retrieves the latest list of Versions for the Configuration Profile. This can change if another User commits a version.

Compare:

If you select two Configuration Profiles, the **Compare** option appears on the right-click context menu. The Configuration Profiles are loaded into the **Profile Editor** tab if they are not already loaded. A new **Compare and Merge** tab is opened that shows the differences, if any, between the two selected Configuration Profiles. For more details, see the [Comparing and Merging Configurations](#) topic.

Configuration Profiles owned by another User

When a User selects a Configuration Profile node that they do not own, only the **Edit**, **Create via Copy**, **Export**, **Refresh**, and **Compare** options are enabled.

Important Note: A User can load another User's Configuration Profile, but can not commit a new version of it. The User can make local changes to the Configuration Profile, and then create a new Configuration Profile from the **Profile Editor** tab that includes the local changes. You can do this using the **Create Configuration** menu option on the top-level node. This is similar to the **Create via Copy** option but differs in that it includes any local changes in the newly created Configuration Profile.

Super User Access

The Super User can access all functionality on this node for all Users. Both the Super User and the owner User can modify a Configuration Profile simultaneously when they both make changes locally. However, the commits are synchronized so that if a commit occurs after you load the Configuration Profile, you are warned when attempting to commit a new Version. You can then choose to continue and overwrite the last User's changes.

Alternatively, you can cancel the commit operation, create a new Configuration Profile from your locally modified Configuration Profile, load the new Version created by the other User, compare both Configuration Profiles, and then merge the changes before committing a new Version. You can use the **Refresh** menu option on the Configuration Profile node to retrieve the latest list of Versions.

Managing a Configuration Version

A Configuration Version node relates to a committed Version of the Configuration Profile that was saved at some point in time. After a Version of a Configuration Profile is created, you can deploy this version of the Configuration Profile to a server (for example, Enterprise Gateway) at any stage. The Super User can access all functionality on this node for all Users.

You can access the following configuration options by right-clicking an historical Version node in the tree:

Edit:

You can load an historical Version, but you can not make new Versions (*branching* is not supported). You can make changes locally and create a new Configuration Profile using the **Create Configuration** menu option in the tree on the **Profile Editor** tab.

Create via Copy:

A copy of the Configuration Version is taken and used to create a new Configuration Profile. You must enter a name for the Configuration Profile. A new Configuration Profile node appears under the **My Configuration Profiles** node.

Export:

You are asked to enter a filename to export the Version to. The Version of the Configuration Profile is exported to this file.

Compare:

If you select two Configuration Profiles, or two Versions, or one Configuration Profile and one Version, the **Compare** menu option appears. For more details, see [Comparing and Merging Configurations](#).

Rollback:

The **Rollback** option is only available on historical Versions, and is not available on the most recent Version. The **Rollback** option sets the historical version to be the latest version. This enables you to roll back to a selected point in the Version history, and potentially make more changes. You are asked to enter a comment when rolling back to a previous Version. Only the Configuration Profile owner or Super User can perform this action.

Refresh:

Retrieves the latest list of Versions for the Configuration Profile. This can change if another User commits a version.

Important Note:

You can not archive or delete Configuration Versions individually. If this was allowed, the historical trail of updates to the Configuration Profile would not be maintained correctly. If you wish to tidy up a Configuration Profile and remove old Versions from drop-down lists, do the following:

1. Create a copy of the Configuration Profile. This results in a new Configuration Profile with a single Version.
2. Redeploy any Processes that were using the old Configuration Profile so that they are using the new Version of the Configuration Profile.
3. Archive the old Configuration Profile.

Managing Policy Studio Users

Overview

The Policy Studio has its own user store. This means that when logging into the Policy Studio to access a particular process, you must enter the credentials of a user stored in its user store to enable the Policy Studio to connect to the process.

Policy Studio Users are responsible for managing server Processes, Configuration Profiles, and in the case of Super Users, other Policy Studio Users. You can manage Policy Studio Users by clicking the **Users** button at the top left of the **Oracle Enterprise Gateway Dashboard**.

Note:

Policy Studio Users provide access to the configuration and process management features in Policy Studio. Whereas Enterprise Gateway Users provide access to the messages and services protected by the Enterprise Gateway. For more details, see the [Enterprise Gateway Users](#) topic.

Super User Privileges

After installation, a single Super User is defined in the Policy Studio. This User is named `admin`, and has a default password of `changeme`. The Super User has its own set of Configuration Profiles and Processes like any other User. However, the Super User also has special rights in the system, which include the following:

- Add, delete, and update other Users.
- Reset User passwords.
- Add another Super User.
- Perform actions normally only allowed by a Configuration Profile owner (for example, transfer ownership of a Configuration Profile to another User).
- Perform actions normally only allowed by a Process owner (for example, deploy a new Configuration Profile to a Process owned by another User).

Important Note:

A Super User *cannot* delete itself.

Removing the default Super User

If you wish to remove the default Super User, perform the following steps:

1. Add another Super User.
2. Log in as the new Super User.
3. Delete the default Super User.

The **User List** dialog displays all existing Policy Studio Users. Super Users can use this dialog to add, update, and delete Users. These tasks are explained in the sections that follow.

Adding a New User

Complete the following steps to add a new Policy Studio User to the system:

1. Click the **Users** button at the top left of the **Enterprise Gateway Dashboard** to display the **User List** dialog.
2. Click the **Add** button.
3. In the **Add new user** dialog, enter a name for the User in the **User ID** field.

4. Enter a user password in the **Password** field.
5. Select roles for the user from the list of available roles (for example, Administrators or Deployers).
6. Click **OK**.

Removing a User

To remove a User, select it in the **Username** list, click the **Remove** button, and click **OK** to confirm. The user is removed from the list. When the Super User deletes a User, the following occurs in the Policy Studio:

- All Configuration Profiles owned by that User are assigned to the Super User.
- All Processes owned by that User are assigned to the Super User.
- The underlying configuration that holds the Configuration Profiles and Processes for that User is removed from the Policy Studio configuration.
- The User is removed from the Policy Studio configuration.

Resetting a User Password

You can reset a User password as follows:

1. Select the User in the **Username** list.
2. Click the **Reset Password** button.
3. In the **Reset Password** dialog, specify and confirm the new password.
4. Click **OK**.

Managing User Roles

You can manage the roles that are assigned to specific users as follows:

1. Select the User in the **Username** list.
2. Click the **Edit User Roles** button.
3. Select the user roles that you wish to enable for this user in the dialog (for example, Operators and/or Auditors).
4. Click **OK**.

Editing Roles

You can click the **Edit Roles** button to add or delete specific roles that can be enabled for users. You must also add any new roles to the `acl.policy` file in the `conf` directory of your Enterprise Gateway installation. For full details on managing roles, see the topic on [Configuring Role-Based Access Control](#). For example, this explains how to add, edit, and delete specific roles using the Role-Based Access Control (RBAC) model.

Comparing and Merging Configurations

Overview

In the Policy Studio, the **Profile Repository** tab enables administrators to compare different configuration Profiles and merge differences between them. You can select **Window -> Show View -> Profile Repository** from the main menu. On the **Profile Repository** tab, when you select two Configuration Profiles, two Configuration Profile Versions, or one Configuration Profile and one Version in the tree or table, the **Compare** menu option is available. This option loads the selected Configuration Profiles into the **Profile Editor** tab below, and compares them in the **Compare and Merge** tab on the right.

Compare and Merge Example

The **Compare and Merge** tab is best illustrated with an example. The following screen shot shows the **Compare and Merge** tab:

This simple example compares two configurations. The configuration at the top left of the screen is the **Merge from Configuration (Source)**. The configuration on the right is the **Merge into Configuration (Target)**. Any merged changes are applied from the source configuration on the left into the target configuration on the right. By default, all differences between both configurations are displayed in the tree (**Additions**, **Deletions**, and **Conflicts**). However, only **Additions** are selected by default for merging. You must explicitly select **Deletions** and **Conflicts** in the tree for merging.

Viewing Differences

In the following screen shot, the **Monitoring Configuration** node has a **Conflict** (exists in both configurations but is not the same). If you select that node in the **Differences** tree, the **Differences Details** panel at the bottom of the

screen shows the details for this configuration entity, and highlights the changed fields.

Merge from Configuration (Source):		Merge into Configuration (Target):																									
Configuration:	<input type="text" value="Default Core Configuration (Historical Ver..."/>	Configuration:	<input type="text" value="Default Core Configuration"/>																								
Version Number:	<input type="text" value="6"/>	Version Number:	<input type="text" value="8"/>																								
Version Timestamp:	<input type="text" value="Wed, 21 Jul 2010 10:09:34 +0100"/>	Version Timestamp:	<input type="text" value="Wed, 21 Jul 2010 10:12:47 +0100"/>																								
Comment:	<input type="text" value="disabled monitoring of message path ."/>	Comment:	<input type="text" value="enabled monitoring of message path :"/>																								
<input type="button" value="Swap"/>																											
Difference Counts: <div style="display: flex; justify-content: space-between; align-items: center;"> Total Differences: <input type="text" value="27"/> Additions: <input type="text" value="0"/> Deletions: <input type="text" value="26"/> Conflicts: <input type="text" value="1"/> <input type="button" value="Compare"/> </div>																											
Differences: 																											
Difference Details: <div style="border: 1px solid #ccc; margin-bottom: 5px; padding: 5px;">[RealtimeMonitoring]name=Monitoring Configuration</div> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Field</th> <th style="width: 40%;">Default Core Configuration (Historical Version - 6 ...)</th> <th style="width: 40%;">Default Core Configuration</th> </tr> </thead> <tbody> <tr> <td>dbConn</td> <td>-1</td> <td>-1</td> </tr> <tr> <td>messageMonitoringEnabled</td> <td>0</td> <td>1</td> </tr> <tr> <td>name</td> <td>Monitoring Configuration</td> <td>Monitoring Configuration</td> </tr> <tr> <td>metricsStoringEnabled</td> <td>0</td> <td>0</td> </tr> <tr> <td>loadorder</td> <td>95</td> <td>95</td> </tr> <tr> <td>keepMsgContentForMsgMo...</td> <td>0</td> <td>1</td> </tr> <tr> <td>watchAgeOrNumber</td> <td>0</td> <td>0</td> </tr> </tbody> </table>				Field	Default Core Configuration (Historical Version - 6 ...)	Default Core Configuration	dbConn	-1	-1	messageMonitoringEnabled	0	1	name	Monitoring Configuration	Monitoring Configuration	metricsStoringEnabled	0	0	loadorder	95	95	keepMsgContentForMsgMo...	0	1	watchAgeOrNumber	0	0
Field	Default Core Configuration (Historical Version - 6 ...)	Default Core Configuration																									
dbConn	-1	-1																									
messageMonitoringEnabled	0	1																									
name	Monitoring Configuration	Monitoring Configuration																									
metricsStoringEnabled	0	0																									
loadorder	95	95																									
keepMsgContentForMsgMo...	0	1																									
watchAgeOrNumber	0	0																									
<input type="button" value="Merge"/>																											

For example, the `messageMonitoringEnabled` field value differs between both configurations. When you perform a **Merge**, the value of this field in the **Merge into Configuration (Target)** is updated to that in the **Merge from Configuration (Source)**.

If a difference is marked as an **Addition**, this means that it exists in the source configuration, but not in the target configuration. When you perform a **Merge**, you add the entities into the target configuration. If a difference is marked as a **Deletion**, this means that it exists in the target configuration, but not in the source configuration. When you perform a **Merge**, you delete the entities from the target configuration. In this example, the generated filter circuits for the `HelloWorldService` would be deleted in a merge.

Some configuration entities contain references to other entities. In this case, an icon is displayed for the field in the **Dif-**

ference Details panel. If you double-click a row with an icon, you can drill down to view further **Difference Details** dialogs for those entities.

Swapping the Source and Target

You can swap the source and target configurations by clicking the **Swap** button. When this button is clicked, **Additions** become **Deletions**, and **Deletions** become **Additions** in the **Differences** tree.

To refresh the **Differences** tree, click the **Compare** button. If modifications are made in the **Profile Editor** tab after the **Compare and Merge** tab is opened, a refresh of the comparison is required to show new differences.

Filtering Differences

To filter nodes from the **Differences** tree view based on their type, right-click in the tree, and select from the following **View Nodes** options:

Additions	Exist in the source configuration being imported but not in the destination configuration. Displayed as a green plus icon.
Deletions	Exist in the destination configuration but not in the source configuration being imported. Displayed as a red minus icon.
Conflicts	Exist in both configurations but are not the same. Displayed as a yellow warning icon.
External Connection Conflicts	External Connections that exist in both configurations but are not the same. Displayed as a yellow warning icon.
Identical	Exist in both configurations and are exactly the same. Displayed as grayed out nodes.

By default, all differences are displayed, and identical nodes are not displayed.

Selecting Differences for Merging

You can merge all differences in the tree, or a subset of differences by right-clicking, and selecting the **Select Nodes for Merging** options:

- **Additions**
- **Deletions**
- **Conflicts**
- **External Connection Conflicts**

By default, only **Additions** are automatically selected for merging. Alternatively, you can manually select the check boxes on the tree nodes. When you click the **Merge** button, only the tree items that are selected are merged into the configuration on the right. You must also take care to merge any entities that are referred to.

The **Difference Counts** field shows the number of entities that are classed as **Additions**, **Deletions**, and **Conflicts**. If all differences are merged, these counts are reset to zero. If there are no differences between both configurations, the tree is empty and the **Difference Counts** is set to zero.

Migrating from a Staging to a Production Environment

Overview

The Policy Studio enables you to manage Configuration Profiles for Oracle Enterprise Gateways in development, test, and production environments. You can use the **Profile Repository** tab to transfer and migrate policies from one environment to another. From the main menu, select **Window -> Show View -> Profile Repository**.

For example, a Configuration Profile is first designed in a development environment by a developer. The developer deploys this Configuration Profile to a Process in the development environment. Using the Policy Studio, the developer can create a version for each significant change they make to the Configuration Profile, which enables them to roll back to a previous version if required.

When the developer has finished, a test engineer takes the Configuration Profile from development and deploys it to a process in the QA environment to perform system testing. During testing, the developer may be required to make changes to the Configuration Profile, and pass them to the QA environment. The QA engineer merges these changes into the test Configuration Profile, creating a new version for each significant update. Finally, when testing is complete, the final version of the Configuration Profile is deployed to a production environment.

This topic describes the recommended steps for moving a Configuration Profile from one environment to another. The approach is the same regardless of moving from development to test, or from test to production. This topic refers to the source and target environment, where both the source and target may be development, test, or production. It describes how a new Configuration Profile is transferred from one environment to another, and how incremental changes are made and need to be merged from one environment to another.

Accessing Environments

You can use a single Policy Center server to manage all Configuration Profiles and Processes across development, test, and production environments. However, in some cases, the various environments are on completely different networks. In this case, separate instances of the Policy Center server may be required.

To perform a transfer or merge of a Configuration Profile from one environment to another, the source and target Configuration Profiles must be available in the same Policy Center server instance. The User performing the merge must be permitted to load both Configuration Profiles into the **Profile Editor** tab.

You can transfer Configuration Profiles between Policy Center instances in test and production environments by first exporting the Configuration Profile using the Policy Studio connected to the Policy Center server in the test environment. You can then import the Configuration Profile to create a new Configuration using the Policy Studio connected to the Policy Center server in the production environment.

Configuration that Differs between Environments

For example, assume that all policies in the Configuration Profile are the same between the source and target environments (for example, test and production). The only items that differ between environments are those that refer to external resources because there are both test and production instances of these items. Differences are to be expected in the following configuration entities:

- Hosts, ports, and URLs to external resources (for example, database connections, LDAP directories, back-end Web Services, and so on)
- Connection credentials to external resources (for example, username/password, SSL keys, and certificates)
- Keys and certificates (for example, for signing a message, or for SSL connectivity between the Process and clients)

Naming Configuration Entities

To keep a merge operation as straightforward as possible, the names of all configuration entities should be the same in both the source and target environment. For example, policy name should be the same and the names of filters in policies should also be the same.

The differences between configuration entities must be kept at the lowest level possible. For example, you have an HTTP Basic authentication filter that authenticates the username and password against a database repository. When configuring this filter, enter a name for the filter, a name for the repository, and a name for the database connection. All of these names should be the same in the source and target Configuration Profiles.

However, when configuring the details of the database connection, the URL, Username, and Password fields differ between environments. You could, enter the name of the database connection differently in the source and target Configuration Profiles (for example, **Test System Database** and **Production System Database**). However, Oracle recommends that you name the database connection as the same in both (for example, **System Database**). This keeps the differences at the lowest level possible, which makes the merge operation easier.

All certificates should have an associated environment-independent alias name. The alias name should be the same in both the source and target environments. Note that the default alias name is the distinguished name of the certificate, which should not be used when merging Configuration Profiles from a source to a target environment.

For example, assume that the Enterprise Gateway server SSL certificate in the testing environment has a distinguished name of CN=test-host.acme.com, C=ie. The production Enterprise Gateway server SSL certificate has a distinguished name of CN=production-host.acme.com, C=ie. By assigning an alias of **Server SSL Certificate**, the certificate used for the same purpose in the test and production environments can be identified using the same alias name. The alias name is assigned when a certificate is created or imported, and can be updated at any time using the **Certificates** panel.

Initial Transfer from Source to Target Environment

Assume that the source Configuration Profile is available to the User in their Policy Center server instance. For information on how to transfer Configuration Profiles managed by multiple server instances, see [Accessibility of Environments](#). Initially, there is no target Configuration Profile because this is the initial transfer of a Configuration Profile (the first time to set up an environment). To perform an initial transfer of configuration from a source to a target environment, do the following:

Create a copy of the source configuration

Perform the following steps:

1. On the **Profile Repository** tab, select the source Configuration Profile, right-click, and select **Create via Copy**.
2. Enter a name for the new Configuration Profile (for example, **Production Configuration**).
3. Enter an optional comment (for example, **Copied from Testing Configuration**).
4. Click **OK**.

Update external resources to point to the target environment

You now have a target Configuration Profile, but it is not yet ready to deploy to the target Process. Right-click the new target Configuration Profile, and select **Edit** to load the Configuration Profile into the **Profile Editor** tab.

You must manually find all configuration entities that refer to external resources in the source environment and update them to point to resources in the target environment.

Deploy to the target environment

Having gone through each environment-specific configuration entity and reconfigured it to point to a target environment resource, you must now create a new version using the **Commit Version** menu option. You are now ready to deploy the target Configuration Profile to the target Process in the target environment (for example, the Enterprise Gateway in the production environment).

In the Policy Studio, select the target Process on the **Deployment Details** tab on the **Oracle Enterprise Gateway Dash-**

board. Select the Configuration Profile, and click the **Deploy** button to deploy the Configuration Profile to the Process.

Configuration that may need to be Modified

The following list highlights the configuration entities that may need to be modified:

Certificates, Keys, and URLs:

Certificates, keys, and CRLs differ between source and target environments. As described above, each certificate should have the same alias name in the source and target environments. Use the **Certificate** tree node to view all certificates in the opened Configuration Profile in the **Profile Editor** tab. Select each certificate, and click the **Edit** button. Import the target environment certificate, (or key and certificate), and save the changes. Leave the alias name unchanged.

The following configuration entities may refer to certificates:

- HTTPS Interface
- File Logger
- Database Logger
- The following Filters:
 - SAML Assertion Insertion filters, SAML PDP filters, WS-Security Username Token filters, XML-Signature Generation and Verification filters, CRL(static) filter, Certificate Chain filter, Find Certificate filter, Connection filter
- The following External Connections:
 - OCSP Connection, XKMS Connection, URL Sets (Entrust, OCSP, SAML PDP, XKMS)

Using the alias name, the reference to the Certificate appears constant across the source and target environments (the alias name is displayed). No changes need to be applied to the configuration entities listed above.

CRLs:

All static CRLs that are specific to the source environment must be removed and replaced with the target environment CRLs. This must be done using the Static CRL filter.

External Connections:

An easy way to find a lot of the external resources is to open the **External Connections** profile in the **Profile Editor** tab. All child nodes (with the exception of the **Authentication Repository Profiles**) potentially point to resources that are environment-specific. Take care when updating these resources to leave the **Name** fields the same (only update the fields that really need to be changed).

Alert Systems:

The system that a Process sends an alert to is an external resource (Email, SNMP, OPSEC, Windows Event Log, and Local Syslog or Remote Syslog). Use the **Alerts** node in the opened **External Connections** profile in the **Profile Editor** tab to update these items to point to resources in the target environment. As before, leave all **Name** fields unchanged. Update all other fields as required.

UDDI Registry Connection:

The UDDI Registry Connections are currently only used by the Policy Studio to locate WSDL files. The connections contained in a Configuration Profile may need to be updated to point to a UDDI registry in the target environment. In the **Core Configuration** profile, right-click the **Web Service** node under the **Web Service Repository** node, and select the **Register Web Service** menu option. Select the **WSDL from UDDI** option, and click the **Browse UDDI** button. Edit all Registry Connections from the first wizard page, ensuring that the **Registry Name** field remains unchanged. Update all other fields as required.

Remote Host:

Remote Host configuration may differ between source and target Configuration Profiles. You can find Remote Hosts under the Process node in the **Core Configuration** on the **Profile Editor** tab. Modify these manually to point to a target environment server after copying the source Configuration Profile.

JMS Service:

JMS Service configuration may differ between source and target Configuration Profiles. JMS Services can be found under the Process node in the **Core Configuration** on the **Profile Editor** tab. Modify these manually to point to a target environment server after copying the source Configuration Profile. Note that a JMS Service is referred to as the **Messaging system** filter. This is updated in the filter automatically.

Static Router Filter:

The Static Router filter **Host** field refers to the back-end Web Service to which messages are routed. This hostname may differ between source and target environments. Oracle recommends that you do not enter a physical hostname here, instead enter the name of a Remote Host that maps to the physical host. You can configure Remote Hosts by right-clicking the Process node in the **Core Configuration** on the **Profile Editor** tab. The Remote Host configuration differs between the source and target environments, while the Static Router remains constant. Note that the **Name** field value of the Remote Host must be the same in both environments.

SMTP Filter:

The SMTP filter refers directly to the SMTP server. You must modify this filter manually to point to a target environment server if the copied source Configuration Profile contains this filter.

Comparison of Source and Target after Initial Transfer

It is useful to perform a **Compare** between the source and target Configuration Profiles to familiarize yourself with the **Difference** tree. The Configuration Profiles should be equal, with the exception of the environment-specific configuration entities described above. Select both the source and target Configuration Profiles on the **Profile Repository** tab or the **Profile Editor** tab, right-click, and select **Compare**. The **Difference** tree shows the following differences:

- Additions and Deletions of Certificates
- Conflicts between entities that refer to certificates (for example, Connection filter, signing filters, HTTPS interfaces)
- Conflicts between database connections
- Conflicts between LDAP directories
- Conflicts between SiteMinder, or SOA Security Manager connections
- Conflicts between Tivoli connections
- Conflicts between all types of URL sets and ClearTrust connection sets (child nodes are marked as Additions and Deletions)
- Conflicts between any Alert systems
- Conflicts between syslog loggers
- Conflicts between Remote Hosts
- Conflicts between UDDI Registry Connections
- Conflicts between Messaging system (JMS) filters
- Conflicts between SMTP filters
- Additions and Deletions of JMS Services

Hiding External Connection Conflicts

You can hide known entity types that are deemed to be External Connection types. Right-click the tree, and select **View Nodes**, and ensure that the **External Connection Conflicts** option is unselected. The following differences are now hidden from the tree view:

- Conflicts between database connections
- Conflicts between LDAP directories
- Conflicts between SiteMinder, or SOA Security Manager connections
- Conflicts between Tivoli connections
- Conflicts between all types of URL sets and ClearTrust connection sets (child nodes are marked as Additions and Deletions)
- Conflicts between any Alert systems

- Conflicts between Remote Hosts
- Conflicts between UDDI Registry Connections

This list contains the conflicts that you should be able to ignore as you merge more and more changes from the source to the target. These conflicts hold the environment-specific connection configuration information.

You are left with the following differences being displayed (when External Connection Conflicts are filtered out):

- Additions and Deletions of Certificates
- Conflicts between entities that refer to certificates (for example, Connection filter, signing filters, HTTPS interfaces)
- Conflicts between syslog loggers
- Conflicts between Messaging system (JMS) filters
- Conflicts between SMTP filters
- Additions and Deletions of JMS Services

When you merge more changes between the source and target Configuration Profiles, you must be careful with the Certificates and those entities that refer to Certificates. The source Certificates always appear as Additions, and the target Certificates as Deletions. You most likely never want to delete the target Certificates during a merge operation, so you should make sure that these are never selected in the Difference tree for merging.

You also most likely do not want to import any more source Certificates, so these Additions should be unselected. However, you may wish to import an entity that refers to a source Certificate. You may perform the import without the Certificate, but you must ensure to manually edit that entity after the merge and select a target Certificate.

The other differences relating to loggers, JMS, and SMTP filters must be handled carefully. In some cases, you may not wish to merge these changes from the source to the target. The data displayed in the Difference table for each of these nodes should be examined and a decision then made whether the difference is one you wish to merge or not.

Merging Updates from Source to Target Environment

After the initial transfer of the configuration from the source to the target environment, it is likely that modifications to the configuration need to pass from the source to the target over time. When you wish to add changes from a source Configuration Profile to an existing target Configuration Profile, you must ensure (as before in the initial transfer use case) that both configurations are accessible to the user, who can load both Configuration Profiles into the **Profile Editor** tab. For details on how to transfer configurations managed by multiple instances, see the [Accessing Environments](#).

You must select both the source and target Configuration Profiles in the **Profile Repository** or **Profile Editor** tab, right-click, and select **Compare**. By default, all differences are displayed in the tree, and only the Additions are selected for merging. You should you carry out the following steps:

1. Right-click the tree, select the **View Nodes** option, and ensure that the **External Connection Conflicts** option is unselected. This hides the configuration entities that are known to refer to external resources, and exist in both the source and target Configuration Profiles. Filtering these differences from the view allows you to concentrate on the differences you wish to merge.
2. Examine each node marked as an Addition, Deletion, or Conflict, and select those you wish to merge.
3. A new External Connection type may need to be merged into the target Configuration Profile. In this case, ensure that it is manually updated after the merge in the same way as described for the initial transfer of the Configuration Profile over to the target environment. If an existing External Connection type is removed in the merge, no manual intervention should be required. Addition and Deletions of External Connection types are not filtered from the tree view by unselecting the External Connection Conflicts option in the **View Nodes** menu.
4. Click the **Merge** button.
5. Switch to the **Profile Editor** tab, and ensure the merged entities are now in your target Configuration Profile.

6. When the target Configuration Profile has been checked to ensure all changes are in order, you must create a new version using the **Commit Version** menu option.
7. The updated target Configuration Profile is now available to deploy to a target Process.

Note:

The following entity types are treated as **External Connections** by the **View Nodes** and **Merge Nodes** menu option filters:

- SiteMinderConnection
- TransactionMinderConnection
- TivoliSettings
- LdapDirectory
- UrlSet
- ConnectionSet
- DbConnection
- EmailAlertSystem
- NTAlertSystem
- OpsecAlertSystem
- SnmpAlertSystem
- SyslogAlertSystem
- RemoteHost
- RegistryConnection
- Certificate
- JMSService
- SyslogServer
- SMTPServer

Managing the Audit Trail

Overview

The Oracle Server Process (for example, Enterprise Gateway, Policy Center, or Service Monitor) generates an audit trail for each of the key actions that occurs in the Policy Studio on Configurations, Processes, and Users. For example, this includes when a user logs in, or when a process configuration is updated. All items are written to a file-based audit trail that is stored on the same machine as that on which the server is running. The audit trail rolls over on date change and also when a maximum file size is reached.

In addition, you can sign the audit trail to guarantee its integrity using the private key of a system User. This User's key should be stored in the Policy Studio's Certificate Store.

Setting up the Audit Trail

To configure the Audit Trail, perform the following steps:

1. In the Policy Studio, click the **Services** tab.
2. Expand the **Processes** node in the tree, and right-click a Process node (for example, **Oracle Enterprise Gateway**).
3. Select **Audit Trail** to display the **Configure Audit Trail** dialog.
4. Configure the following fields to force the server to write an audit trail to file:

Enable Audit Trail:

Select this checkbox to configure the Process to start writing event data to the audit trail.

File Name:

Enter the name of the audit trail file in this field. When an audit trail file rolls over (either because the maximum file size has been reached, or because the date has changed), a suitable increment is appended to the file name. Defaults to `ConfigurationManagementAuditTrail`.

File Extension:

Enter the file extension for the audit trail file. Defaults to `.xml`.

Directory:

Enter the directory for the audit trail file. Defaults to `logs`.

File Size:

Specify the maximum size that an audit trail file is allowed reach before it rolls over to a new file. Defaults to 1000 kilobytes.

Roll Log Daily:

Specify whether to roll over the log file at the start of each day. This is enabled by default.

Number of Files:

Specify the number of log files that are stored. The default number is 20.

Signing Key:

You can sign the audit trail with the private key of a User from the Certificate Store. To do this, click the **Signing Key** button, select a User (whose private key is stored in the Certificate Store), and press the **OK** button. For more details, see the [Certificates](#) topic.

Configuration Audit Trail

The audit trail for Configuration management contains the following data:

- **Level:** Log level (Failure or Success).
- **User:** The User performing the action on the Configuration.
- **Timestamp:** The time the action occurred.
- **Text:** Text description of the action.
- **Action:** The action performed on the Configuration (Create New Configuration, Rename, Change Owner, Version, or Archive).
- **Configuration:** The Configuration name.
- **Version:** Only applies for the Commit action.
- **Comment:** Only applies for the Commit action.
- **New Owner:** Applies if the action is Change Owner.

Process Deployment Audit Trail

The audit trail for Process deployment management contains the following fields:

- **Level:** Log level (Failure or Success).
- **User:** The User performing the action on the Process.
- **Timestamp:** The time the action occurred.
- **Text:** Text description of the action.
- **Action:** The action performed on the Process (Create New Configuration, Rename, Change Owner, Version, or Archive).
- **Host:** The host that the Process runs on.
- **Process:** The name of the Process.
- **Configuration:** The name of the Configuration. Applies to the Deploy action.
- **Version:** Applies to the Deploy action.
- **Comment:** The comment from the Version creation.
- **New Owner:** Applies if the action is Change Owner.

User Audit Trail

The Audit trail for User management contains the following fields:

- **Level:** Log level (Failure or Success).
- **User:** The User performing the action on the User.
- **Timestamp:** The time the action occurred.
- **Text:** Text description of the action.
- **Action:** The action performed on the User (Add, Update, or Delete).
- **Update User:** The User the action was performed on.

Configuring Role-Based Access Control

Overview

Role-Based Access Control (RBAC) enables you to restrict system access to authorized users based on their assigned roles. Using the RBAC model, permissions to perform specific system operations are assigned to specific roles. This means that system users are granted permission to perform specific system operations only through their assigned roles. This simplifies system administration because users do not need to be assigned permissions directly, but instead acquire them through their assigned roles.

RBAC is supported by the Enterprise Gateway, Policy Center, and Service Monitor servers to protect access to the management services provided by each server. Management services are invoked when a user performs the following tasks:

- Accesses the server using the Policy Studio
- Accesses the server using the Service Manager
- Requests the Enterprise Gateway Welcome page (<http://localhost:8090/>) in a web browser
- Requests trace, logs, or diagnostics in a web browser
- Uses the Traffic Monitor tool
- Uses the Real-time Monitoring tools

User access to management services is determined by their role or roles. Each role has a defined set of management services that it can access. A management service is defined by the URI used to access it, and in some cases, the SOAP operation and namespace. For more information, see [Management Service Roles and URIs](#).

By default, a local Policy Center user store available in the Policy Studio is used to authenticate and perform RBAC on all access to the management services. This stores the user's credentials so that their passwords can be verified, and also stores their roles. You can also use an LDAP repository for authentication and RBAC. For details on setting up an LDAP repository, see the [Authentication Repository](#) topic.

Server `acl.policy` File

The `acl.policy` file is found in the `conf` directory of your server installation. This file lists each role and the management services that each role may access. By default, this file defines the following roles:

- Administrators
- Operators
- Auditors
- Deployers

The default `admin` user has the `Administrators` role, which allows access to everything. For full details on the management services that each role has access to, and the URIs that must be listed in the `acl.policy` file to have access to them, see the table in [Management Service Roles and URIs](#).

The roles defined in the `acl.policy` file should exist in the user store used to authenticate the users and load their roles and groups. The default roles are defined in the local Policy Center (PD) user store, which is used to control access to the management services using the **Protect Management and Policy Director Interfaces** policy. If a different user store is used (for example, an LDAP repository), the LDAP groups should be listed in the `acl.policy` file.

`acl.policy` File Format

Each entry in the `acl.policy` file has the following format:

```
grant principal com.vordel.circuit.rbac.VordelPrincipal "role-name" {
  permission com.vordel.circuit.rbac.MgmtServicePermission "URI [SOAP-operation-name]
    [SOAP-operation-namespace]";
};
```

This entry format is explained as follows:

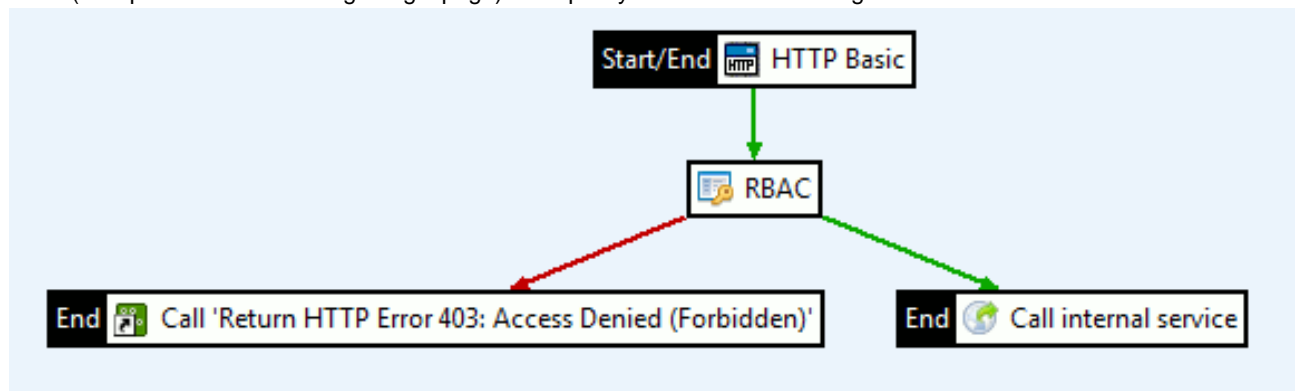
- The `permission` line is repeated for each URI that the role can access. To determine which URIs should be listed for each management service, see the table in [Management Service Roles and URIs](#).
- The SOAP operation name and namespace are optional. They apply only to the SOAP-based management services. If they are omitted for a SOAP-based management service, the role has access to all operations on that interface. If a subset of operations is listed, the user may only invoke those operations that are listed. A separate `permission` line is required for each operation if the user only has access to a subset of operations.
- You can place a wildcard (*) at the end of the URI field. For example, see the permission for the default `Administrators` role in the `acl.policy` file. This means that the role has access to all URIs that start with the URI content that precedes the *.
- In some cases, you must protect a management service by specifying a query string after the URI. Exact matches only are supported for query strings.

Enterprise Gateway Welcome Page

The Enterprise Gateway Welcome page (<http://localhost:8090>) is an index page to the management services for the Enterprise Gateway, and is controlled by RBAC. Connecting to this URL as users with different roles results in different options being displayed. Users with the `Administrators` role can access all available tools, such as Service Manager and Real-time Monitoring, while users in other roles can access a subset of available tools. For example, users with the `Operators`, `Deployers`, or `Auditors` role cannot access Service Manager.

Protect Management and Policy Director Interfaces Policy

By default, the **Protect Management and Policy Director Interfaces** policy controls access to the management services (except the Service Manager login page). This policy includes the following filters:



HTTP Basic Filter

This filter performs the following tasks:

- Verifies the user name and password against the local PD user store. If no credentials are specified or, they are invalid, this always throws an HTTP 401. This allows a retry for browser users.

- Assuming authentication has passed, the user roles are loaded from the user store. (Currently this automatic load of roles is only supported by the local PD user store, which means no additional filters are required to load the roles).
- The username is in the `authentication.subject.id` message attribute, and the user roles are in `authentication.subject.role`.

For more details, see the [HTTP Basic](#) topic.

RBAC Filter

This filter performs the following tasks:

- Reads the roles from the `authentication.subject.role` message attribute.
- Determines what management service is currently being invoked (which URI, and which SOAP operation and namespace where applicable).
- The RBAC filter returns true if one of the roles has access to the management service currently being invoked, as defined in the `acl.policy` file.
- Otherwise, it returns false and the **Return HTTP Error 403: Access Denied (Forbidden)** policy is called. For example, this occurs if a new user is created and they have not yet been assigned any roles.

For more details, see the [RBAC Filter](#) topic.

Call Internal Service Filter

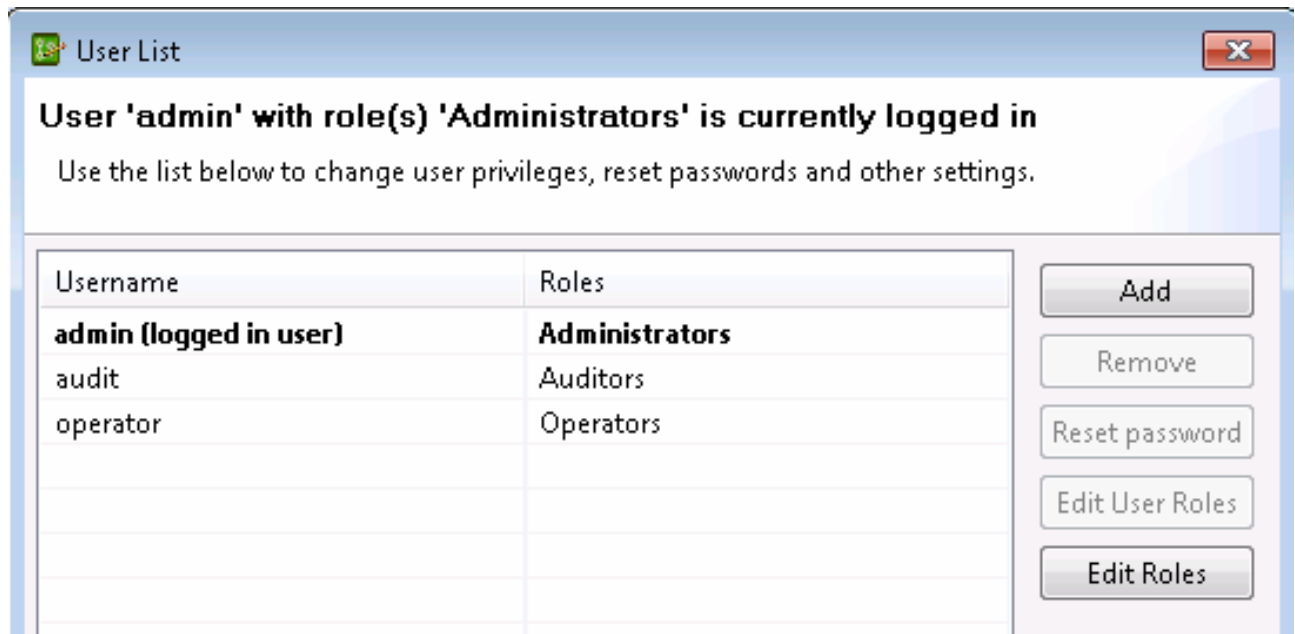
This filter performs the following tasks:

- Assuming the RBAC filter passed, the management service is called.
- The `authentication.subject.id` and `authentication.subject.role` message attributes are passed to the service as HTTP headers. Some of the management services perform further access control checks on the user and their roles at a more in-depth level than URI and SOAP operation (for example, a check based on the parameters contained in the SOAP message).

For more details, see the [Call Internal Service](#) topic.

Policy Center User Store

The local Policy Center user store in the Policy Studio supports user roles. The **User List** screen in the Policy Studio enables the user to view and modify user roles (assuming they have a role that allows this). This screen is displayed as follows:



Managing User Roles

You can select a User in the list to enable the **Edit User Roles** button, and click this button to enable roles for the selected user. You can also click the **Edit Roles** button to add or delete a role from the list of available roles.

Adding a New User Role

When you add a new role to the Policy Center user store, you must modify the `acl.policy` file if RBAC is to be performed on the basis of whether the user has this new role. You must add a new `grant` section to the `acl.policy` file, which lists all the URIs that the new role may access. You must then reboot the server to reload the `acl.policy` file. You do not need to reboot or refresh the server if a user's roles change.

System Administrators Role

By default, the `Administrators` role allows access to all management services. This access is defined in the `acl.policy` file using the `/*` URI. The default `admin` user is assigned to this role. To ensure that users with this role do not lock themselves out of the system, they are not allowed to remove themselves from this role. However, other users that are allowed to change a user's roles are allowed to remove the user from the `Administrators` role.

This special system administrator role is defined in the **Default Settings** as the **Administrator Role**. This setting is only used to provide special protection for the role (a user with this role cannot remove themselves from this role). This ensures that there is always at least one user in the system with this role.

Configuring the Administrator Role

If you wish configure a different role as the system **Administrator Role**, perform the following steps:

1. Log in to the server as a user with the `Administrators` role (the current system **Administrator Role**).
2. Click the **Users** button in the Policy Studio toolbar to open the **User List** dialog.
3. Click **Edit Roles**.
4. Add a new role (for example, `MyAdministrators`), and click **OK**.
5. Select the user that you are logged in as, and click **Edit User Roles**.
6. Add the `MyAdministrators` role to your list of roles, and keep the `Administrators` role. You are not allowed to remove it yet because it is still the system **Administrator Role**.
7. Edit the server configuration in the Policy Studio.

8. Right-click the server process, and select **Settings -> Default Settings** (or **Custom Settings** if used).
9. Set the value of the **Administrator Role** and **Policy Director Super User Role** to `MyAdministrators`. For details on the **Policy Director Super User Role**, see the next section.
10. Deploy the configuration.
11. Add the following to the `acl.policy` file:

```
grant principal com.vordel.circuit.rbac.VordelPrincipal "MyAdministrators" {
    permission com.vordel.circuit.rbac.MgmtServicePermission "/*";
};
```

12. Reboot the server.
13. Restart the Policy Studio, and open the **User List** dialog.
14. Remove all users from the `Administrators` role by clicking **Edit User Roles** on each user with that role, assuming this role is no longer required.
15. You can now remove the `Administrators` role by clicking **Edit Roles**.

PD Superuser Privilege

PD Superuser privilege gives a specified role special system administrator rights in the system, for example:

- Add, delete, and update other users.
- Reset user passwords.
- Add another superuser.
- Perform actions only allowed by a Configuration Profile owner (for example, transfer ownership to another user).
- Perform actions only allowed by a Process owner (for example, deploy a new Configuration Profile to a Process owned by another user).

The PD superuser privilege is checked in the Policy Center API. RBAC checks are also performed in the **RBAC** filter before getting to the Policy Center API. The superuser checks performed on the user in this API are as follows:

- Ensure is superuser to create or delete a role or user.
- Ensure is superuser/or logged in user to reset the password.
- Ensure is superuser to modify a user (change its roles). Users cannot remove their own PD superuser role, or system admin role.
- Ensure is superuser or configuration/process owner to reassign configurations or processes.
- Ensure is superuser/configuration group owner to update a configuration.
- Ensure is superuser/process group owner to create a process, get connection credentials, or modify a process.
- Ensure is superuser/process group and configuration group owner to deploy a configuration to a process.

Configuring the PD Superuser Role

By default, a user has the PD superuser privilege if they have the default `Administrators` role. This means that the default `admin` user has the PD superuser privilege. The role used to determine whether a user has the PD superuser is defined in the **System Settings** as **Policy Director Super User Role**. If you wish to separate the PD superuser privilege out of the `Administrators` role, perform the following steps:

1. Log in to the server as a user with the `Administrators` role.
2. Click the **Users** button in the Policy Studio toolbar to open the **User List** dialog.

3. Click **Edit Roles**.
4. Add a new role (for example, `PDA administrators`), and click **OK**.
5. Select the user you are logged in as, and click **Edit User Roles**.
6. Add the `PDA administrators` role to your list of roles, and keep the `Administrators` role.
7. Edit the server configuration in the Policy Studio.
8. Right-click the server process, and select **Settings -> Default Settings** (or **Custom Settings** if used).
9. Set the value of the **Administrator Role** and **Policy Director Super User Role** to be the name of the role that determines if the user has the PD superuser privilege.
10. Deploy the configuration. Only users with the `PDA administrators` role are now able to perform PD superuser actions.

Important Note:

Having the `PDA administrators` role does not ensure access to the management services. If required, add this access to the role by editing the `acl.policy` file, and reboot the server when the changes are made. To give the `PDA administrators` role access to all management services, add the following entry to the `acl.policy` file:

```
grant principal com.vordel.circuit.rbac.VordelPrincipal "PDA administrators" {
  permission com.vordel.circuit.rbac.MgmtServicePermission "/*";
};
```

Management Service Roles and URIs

You can use the following table for reference purposes when making changes to the `acl.policy` file. It defines each management service, and the default roles that have access to them. It also lists the URIs that must be listed in the `acl.policy` to have access to the management service. Where applicable, the SOAP operation names of the management service are also listed.

Management Service	Default Roles	URI
Welcome page on <code>http://localhost:8090/</code>	<ul style="list-style-type: none"> Administrators Operators Deployers Auditors 	<ul style="list-style-type: none"> / /index.html /images/* /css/* /favicon.ico
Service Manager	<ul style="list-style-type: none"> Administrators 	<ul style="list-style-type: none"> /manager/* /VAADIN/*
Real-time Monitoring	<ul style="list-style-type: none"> Administrators Operators 	<ul style="list-style-type: none"> /monitoring/* /metrics
Traffic Monitor	<ul style="list-style-type: none"> Administrators Operators 	<ul style="list-style-type: none"> /trafficmonitor/* /ops/*

Trace Files	<ul style="list-style-type: none"> Administrators Operators 	<ul style="list-style-type: none"> /file/view?type=trace&format=html /file/view*?type=trace
Audit (Log) Files	<ul style="list-style-type: none"> Administrators Deployers Auditors 	<ul style="list-style-type: none"> /file/view?type=audit&format=html /file/view*?type=audit
Diagnostics	<ul style="list-style-type: none"> Administrators Operators 	<ul style="list-style-type: none"> /file/view/diagnostics
Documentation	<ul style="list-style-type: none"> Administrators Operators Deployers Auditors 	<ul style="list-style-type: none"> /docs/*
Management API	<ul style="list-style-type: none"> Administrators Deployers (read-only and deploy operations) Auditors (read-only operations) 	<p>URI:</p> <ul style="list-style-type: none"> /runtime/management/ManagementAgent <p>Namespace:</p> <ul style="list-style-type: none"> http://www.vordel.com/2006/07/22/pd/ManagementAgent <p>Operations:</p> <ul style="list-style-type: none"> reloadConfiguration getHostName getProductKey getProductName getProductVersion deploy getActiveDeploymentInfo getStoreContents createNewStore createDuplicateStore deleteStore

Policy Center API	<ul style="list-style-type: none"> • Administrators • Deployers (read-only and deploy operations) • Auditors (read-only operations and change own password) 	<p>URI:</p> <ul style="list-style-type: none"> • /configuration/deployments/DeploymentService <p>Operations:</p> <ul style="list-style-type: none"> • login • logout • listConfigurationFlavors • getConfiguration • commit • deploy • setActiveConfiguration • getActiveConfiguration • getActiveConfiguration-Fingerprint • listUsers • getUserDetails • getUserRoles • createUser • modifyUser • modifyUserRoles • setUserPassword • removeUser • listProcessGroups • getProcessDetails • getProcessDetailsForID • listProcesses • createProcess • modifyProcess • removeProcess • reassignProcesses • getConnectionCredentials • listConfigurationGroups • getConfigurationDetails • getConfigurationDetails-ForID • listConfigurations • createConfiguration • spawnConfiguration • importConfiguration • importConfiguration-FromProcess • modifyConfiguration • archiveConfiguration • reassignConfigurations
-------------------	--	--

		<ul style="list-style-type: none"> • listVersions • getVersionDetails • getLatestVersion • createProcessGroup • modifyProcessGroup • removeProcessGroup • reassignProcessGroup • createConfigurationGroup • modifyConfigurationGroup • removeConfigurationGroup • reassignConfigurationGroup • getRoleDetails • listRoles • createRole • removeRole • hasAccess • hasAdminRole
--	--	---

Management Agent API Operations

The Management Agent API operations are described as follows:

Operation	Description
createNewStore	When the Policy Studio is connected to Policy Center, a Deployment Wizard is available. When a user deploys using the wizard, the Policy Center API <code>deploy</code> method is invoked. This in turn calls <code>createNewStore</code> on the management agent API of the Enterprise Gateway when the store needs to be created, followed by <code>deploy</code> on the management API of the Enterprise Gateway.
deploy	Invoked by the Policy Center <code>setActiveConfiguration</code> method which is called when a user clicks deploy when editing the active configuration.
getActiveDeploymentInfo	Called when a user edits the active configuration of a process.
getHostName	Called by Policy Center on managed processes (Enterprise Gateway also calls on itself).
getProductKey	Called by Policy Center on managed processes (Enterprise Gateway also calls on itself).
getProductName	Called by Policy Center on managed processes (Enterprise Gateway also calls on itself).
getProductVersion	Called by Policy Center on managed processes (Enterprise Gateway also calls on itself).
getStoreContents	Called when a user edits the active configuration of a process.

Policy Center API Operations

The Policy Center API operations are described as follows:

Operation	Description
archiveConfiguration	Called when a user selects to archive a configuration on the Profile Repository view.
createConfiguration	Called when a user selects to Create a baseline Configuration , or Create New Configuration on their Profile Repository view.
createProcess	Called when a user adds a process to their process list.
createUser	Create a new user.
create*	Create users, roles, and so on.
commit	Called when a user commits a new version of a configuration. This can be done in the Profile Repository view, or while editing an active configuration.
deploy	When the Policy Studio is connected to Policy Center, a Deployment Wizard is available. When a user deploys using the wizard, this method is invoked. This in turn calls <code>createNewStore</code> on the management agent API of the Enterprise Gateway when the store needs to be created, followed by <code>deploy</code> on the management API of the Enterprise Gateway.
getActiveConfiguration	Called when a user edits the active configuration of a process. The management API methods <code>getActiveDeploymentInfo</code> and <code>getStoreContents</code> are also called. Note that when you connect to the Enterprise Gateway using the Policy Center server, <code>getActiveConfiguration</code> is called on Policy Center and <code>getActiveDeploymentInfo</code> and <code>getStoreContents</code> are called on the Enterprise Gateway by Policy Center.
getActiveConfigurationFingerprint	Determines the status of deployment for a process (has it changed since last known deployment).
getConfiguration, getConfigurationDetails-ForID	Called when a user edits a configuration in the Profile Repository . Also called when a user exports a configuration.
get*	Read-only operations to retrieve details of users, roles, configurations, process and so on, which are stored in <code>pdEntities.xml</code> .
hasAccess	Called by the Policy Studio when connected to the Enterprise Gateway to determine if the user has access to a particular SOAP API method according to RBAC.
hasAdminRole	Called to determine if the user has the Policy Center Super User Role .
importConfiguration	Called when a user selects Create Configuration via Import in the Profile Repository view.
importConfigurationFromProcess	Called when a user selects Download the Active Configuration for a process.
login	User logs into the server using the Policy Studio.
logout	User logs out of the server using the Policy Studio.
list*	Read-only operations to retrieve users, roles, configura-

Operation	Description
	tions, processes, and so on, which are stored in <code>pdEntities.xml</code> .
<code>modifyConfiguration</code>	Rename a configuration in the Profile Repository view.
<code>modifyProcess</code>	Modify the connection credentials for a process in the Connection Details tab.
<code>modify*</code>	Modify user roles, and so on.
<code>remove*</code>	Delete users, processes, and roles.
<code>reassign*</code>	Called when a user changes ownership of a process or configuration.
<code>setActiveConfiguration</code>	Called when a user clicks the Deploy button when they are editing an active configuration. This invokes the <code>deploy</code> method on the management agent. When the Policy Studio user is connected to a Enterprise Gateway using Policy Center, <code>setActiveConfiguration</code> is called on Policy Center and the <code>deploy</code> is called on the Enterprise Gateway.
<code>setUserPassword</code>	Reset the user password.
<code>spawnConfiguration</code>	Called when a user select to Create via Copy a configuration in the Profile Repository view.

Startup Instructions

Overview

This topic describes how to start the Enterprise Gateway and Policy Studio on all platforms in both *console mode* and as a background service in *service mode*. For details on launching Service Manager in your browser, see [Getting Started with Service Manager](#). For details on Enterprise Gateway components and concepts, see the [Oracle Enterprise Gateway Architecture](#).

Setting Passphrases

By default, data is stored unencrypted in the Enterprise Gateway configuration store. However, you can encrypt certain sensitive information, such as passwords and private keys using a passphrase. When the passphrase has been set, this encrypts the Enterprise Gateway configuration data. You must enter the passphrase when connecting to the Enterprise Gateway configuration data both using the Policy Studio and the Enterprise Gateway.

For more details on configuring this passphrase, see [Setting the Encryption Passphrase](#).

Start in Console Mode

The following instructions describe how to start the Enterprise Gateway in *console* mode on Windows and Linux/Solaris systems:

Windows

Complete the following steps to start the Enterprise Gateway on a Windows system:

1. Open a DOS prompt at the `/win32/bin` directory of your Enterprise Gateway installation.
2. To start the Enterprise Gateway, run the `enterprisegateway.bat` file from this directory.
3. If you are using an encryption passphrase, you are prompted for this passphrase. Enter the correct encryption passphrase, and press Return. For more details, see [Setting Passphrases](#).
4. When the Enterprise Gateway has successfully started up, you can run the `policystudio.exe` file from your Policy Studio installation directory.
5. When the Policy Studio is starting up, you are prompted for connection details for the Enterprise Gateway. For more details, see [Connecting to the Enterprise Gateway](#).

Linux/Solaris

To start the Enterprise Gateway and the Policy Studio on Linux/Solaris systems, complete the following instructions:

1. Open a shell at the `/posix/bin` directory of your Enterprise Gateway installation.
2. To start the Enterprise Gateway, run the `enterprisegateway.sh` file from this directory. For example:

```
prompt# ./enterprisegateway
```

3. If you are using an encryption passphrase, you are prompted for this passphrase. Enter the correct encryption passphrase and press Return. For more details, see [Setting Passphrases](#).
4. When the Enterprise Gateway has successfully started up, run the `policystudio.sh` file in your Policy Studio installation directory:

```
prompt# cd /usr/home/policystudio
prompt# ./policystudio
```

Connecting to the Enterprise Gateway

When starting up the Policy Studio, you are prompted for details on how to connect to the Enterprise Gateway. These include details such as the server session, host, port, user name, and password. The default connection URL is:

```
http://HOST:8090/configuration/deployments/DeploymentService
```

where `HOST` points to the IP address or hostname of the machine on which the Enterprise Gateway is running. For more information on configuring these settings, see [Connection Details](#).

Start in Service Mode

The Enterprise Gateway can also be started in *service* mode as a Windows service, or as a daemon on Linux and Solaris systems.

Windows

The following instructions describe how to install, start, stop, and remove the Enterprise Gateway as a Windows service. The following examples refer to the directory in which you installed the Enterprise Gateway as `INSTALL_DIR`.

You can install and start the Enterprise Gateway as a Windows Service from the Enterprise Gateway Windows Program Group. You can also stop and remove the Service from this group. The Enterprise Gateway Program Group is available from the following location:

Start -> All Programs -> Enterprise Gateway -> Service

The following options are available from this group:

- **Install:**
Installs the Enterprise Gateway as a Service.
- **Start:**
Starts the Enterprise Gateway as a Service. You must install the Enterprise Gateway as a Windows Service using the **Install** option before you can start the service.
- **Stop:**
Stop the Enterprise Gateway as a Service.
- **Remove**
Removes the Enterprise Gateway as a Windows Service.

Important Note:

You can encrypt all sensitive Enterprise Gateway configuration data with an encryption passphrase. For example, you can specify this passphrase in your Enterprise Gateway configuration file, or on the command line when the Enterprise Gateway is starting up. For more details on configuring this passphrase, see [Setting the Encryption Passphrase](#).

Linux/Solaris

The instructions in this section describe how to stop and start the Enterprise Gateway as a daemon on Linux and Solaris systems. Open a shell at the `INSTALL_DIR/posix/bin` directory, where `INSTALL_DIR` refers to the root of your Enterprise Gateway installation.

Start as a Daemon Process:

```
prompt# ./enterprisegateway -d
```

Kill the Daemon Process:

```
prompt# ./enterprisegateway -k
```


Connection Details

Overview

You can use the Policy Studio to manage Enterprise Gateway, Policy Center, and Service Monitor servers. The **Open Connection** dialog enables you to connect to a server URL, and the **Open File** dialog enables you to connect to a server configuration file. By default, the Policy Studio connects to a server URL. This topic describes how to connect using both options.

Connecting to a Server URL

The server exposes a deployment service to its underlying configuration data. This enables Policy Studios running on different machines to that on which the server is installed to manage policies remotely. To connect to the deployment service of a running server, select **File -> Connect to server** from the main menu, or the equivalent button in the toolbar. Configure the following fields on the **Open Connection** dialog:

Saved Sessions:

Select the session that you wish to use from the drop-down list. You can edit a session name by entering a new name and clicking **Save**. You can also add or remove saved sessions using the appropriate button.

Connection Details

The **Connection Details** section enables you to specify the following settings:

Host:

Specify the host to connect to in this field. The default is `localhost`.

Port:

Specify the port to connect on in this field. The default Enterprise Gateway port is 8090.

User Name:

The deployment service is protected by HTTP Basic authentication. You must provide a user name and password so that the Policy Studio can authenticate to the server. By default, the server User Store contains an `admin` user with a `changeme` password, which can be used in this case. You can change this user's details using the [User Store](#) interface.

Password:

Specify the password for the user. The password for the default `admin` user is `changeme`.

Advanced

Click **Advanced** to specify the following setting:

URL:

Enter the URL of the deployment service exposed by the server. For example, the default Enterprise Gateway server URL is `http://HOST:8090/configuration/deployments/DeploymentService`, where `HOST` points to the IP address or host name of the machine on which the Enterprise Gateway is running. You can also connect to the Policy Center or the Service Monitor server URL. The default server URL addresses are as follows:

Component	Address
Enterprise Gateway server	ht- tp://localhost:8090/configuration/deployments/DeploymentService
Policy Center server	ht- tp://localhost:8060/configuration/deployments/DeploymentService
Service Monitor server	ht-

Component	Address
	tp://localhost:8040/configuration/deployments/DeploymentService

Important Note:	To manage multiple Enterprise Gateways in your network, you must connect to the Policy Center server URL.
------------------------	---

Connecting to a Server Configuration File

Because the server configuration data is stored in an XML file by default, you can specify that the Policy Studio connects directly to a server configuration file. To connect to a configuration file, select **File** -> **Open file** from the main menu, or the equivalent button in the toolbar. Complete the following fields on the **Open File** dialog:

File:

Enter or browse to the location of the server configuration file (for example, `INSTALL_DIR\conf\fed\configs.xml`).

Passphrase Key:

All sensitive server configuration data (password, keys, and so on) can be encrypted using a passphrase. If you wish to do this, enter a password in this field when connecting. You must use this password thereafter when connecting to the server.

Unlocking a Server Connection

You can also use the **Open File** dialog to unlock a connection to a server. This is for emergency use when you have changed configuration that results in you being locked out from the **Management Services** on port 8090. In this case, you have misconfigured the authentication filter in the **Protect Management Interfaces** policy. For example, if you created and deployed an LDAP connection without specifying the correct associated user accounts, and are now unable to connect to the Enterprise Gateway server.

To unlock a server connection, perform the following steps:

1. Download all the files in the server's `INSTALL_DIR/conf/fed` directory to the machine on which the Policy Studio is installed.
2. Start the Policy Studio.
3. Connect to the `configs.xml` file that you downloaded from the server in step 1 (for details, see [Connecting to a Server Configuration File](#)).
4. Change the configuration details as required (for example, specify the correct user account details for the LDAP connection on the **External Connections** tab).
5. Upload the files back to the server's `INSTALL_DIR/conf/fed` directory.
6. Connect to the server URL in the Policy Studio.

For more details on Management Services, see [Policy Studio Preferences](#).

Policy Studio Preferences

Overview

The **Preferences** dialog enables you to configure a range of options for the Policy Studio. For example, you can configure the level at which the Policy Studio traces diagnostic output, customize the look-and-feel of the Policy Studio, or configure the timeout for the Policy Studio connection to the Enterprise Gateway. Each of the available settings are discussed in the following sections.

Management Services

The Enterprise Gateway exposes certain interfaces that are purely used for management purposes only, and should only be edited under strict advice from the Oracle Support team. By default, the **Management Services** policies and interfaces, and the Policy Center interfaces and server process are hidden from view in the Policy Studio. You can display them by selecting the **Show Management Services** checkbox.

When this setting is enabled, you can view the **Management Services** policy container in the **Policies** tree. Similarly, the **Management Services** HTTP interfaces are displayed under the Server Process in the **Services** tree. If you are running Policy Center, the **Policy Center Service** is displayed in the **Services** tree. The Policy Center Server Process is also displayed on the **Oracle Enterprise Gateway Dashboard** tab.

For more details, see the [Management Services](#) section in the [Configuring HTTP Services](#) topic.

Important Note:

As stated earlier, you should only modify **Management Services** under strict advice and supervision from the Oracle Support team.

Policy Colors

The **Policy Colors** settings enable you to customize the look-and-feel of the Policy Canvas in the Policy Studio. For example, you can change the colors of the following components:

- **Policy Background:**
Changes the background color of the Policy Canvas.
- **Missing Attribute:**
You can right-click the Policy Canvas, and select **Show All Attributes** from the context menu. When this is selected, each filter displays the list of required and generated message attributes that are relevant for that filter. If a required attribute has not been generated by a previous filter in the policy, the attribute is highlighted in a different color (red by default). You can change this color by selecting an appropriate color using this setting.
- **Success Path:**
You can change the color of the Success Path link using this setting.
- **Failure Path:**
Similarly, you can change the color of the Failure Path link here.
- **Show Link Labels:**
If this option is selected, a Success Path is labeled with the letter S, while a Failure Path is labeled F.

Proxy Settings

You can specify global proxy settings that apply only when downloading WSDL, XSD, and XSLT files from the Policy Studio. These include the following settings:

<i>Proxy Setting</i>	<i>Description</i>
Host	Host name or IP address of the proxy server.

<i>Proxy Setting</i>	<i>Description</i>
Port	Port number on which to connect to the proxy server.
Username	Optional user name when connecting to the proxy server.
Password	Optional password when connecting to the proxy server.

You can also specify individual proxy servers on the **External Connections** tab. These are different from the global proxy settings in the **Preferences** because you can specify these proxy servers at the filter level (in the **Connection** and **Connect To URL** filters). For more details, see the [Proxy Servers](#) topic.

Runtime Dependencies

The **Runtime Dependencies** setting enables you to add JAR files to the Policy Studio classpath. For example, if you write a custom message filter, you must add its JAR file, and any third-party JAR files that it uses, to the **Runtime Dependencies** list.

Click **Add** to select a JAR file to add to the list of dependencies, and click **Apply** when finished. A copy of the JAR file is added to the `plugins` directory in your Policy Studio installation.

Important Note:

You must restart the Policy Studio and the server for these changes to take effect.

Server Connection

When an update is made to a configuration setting in the Policy Studio, the update is stored locally until you deploy it to the Enterprise Gateway. After making one or more configuration updates, you can deploy to the Enterprise Gateway by selecting **Settings -> Deploy** in the main menu. Alternatively, you can click the **Deploy** button in the toolbar, or press F6.

The updated configuration is then pushed to the Enterprise Gateway, which finishes processing any current messages before flushing its cached policy configurations. The new configuration is then stored and loaded. Any subsequent messages received by the Enterprise Gateway then use the new configuration.

When deploying to the Enterprise Gateway, the Policy Studio sends a deployment request to the server. If necessary, you can configure the socket timeout value for this connection in the Policy Studio **Preferences** dialog. Enter the timeout value in milliseconds in the **Server Socket Connection Timeout** field.

Trace Level

You can set the level at which the Policy Studio logs diagnostic output by selecting the appropriate level from the **Tracing Level** drop-down list. Diagnostic output is written to a file in the `/logs` directory of your Policy Studio installation. You can also select **Window -> Show View -> Console** in the main menu to view the trace output in the **Console** window at the bottom of the screen. The default trace level is `INFO`.

Web and XML

The **Web and XML** settings enable you to configure a range of options that affect how XML files are treated in the Policy Studio.

XML Files

This includes the following options:

Creating or saving files	Specifies a line delimiter (for example, Mac, Unix, Windows, or No translation).
---------------------------------	--

Creating files	Specifies a file suffix (<code>.xml</code>), and the type of encoding (for example, <code>ISO 10646/Unicode(UTF-8)</code>).
Validating files	Configures whether to warn when no grammar is specified.

Source

This includes the following options:

Formatting	Specifies a range of formatting options (for example, line width, line breaks, and indentation).
Content assist	Specifies whether to make suggestions and which strategy to use (for example, <code>Lax</code> or <code>Strict</code>).
Grammar constraints	Specifies whether to use inferred grammar in the absence of DTD/Schema.

Syntax Coloring

These settings enable you to associate specific colors with specific XML syntax elements (for example, attribute names, comment delimiters, or processing instruction content).

WS-I Settings

Before importing a WSDL file that contains the definition of a Web Service into the Web Services Repository, you can test the WSDL file for compliance with the Web Service Interoperability (WS-I) Basic Profile. The WS-I Basic Profile contains a number of *Test Assertions* that describe rules for writing WSDL files for maximum interoperability with other WSDL authors, consumers, and other related tools.

The WS-I Settings are described as follows:

WS-I Setting	Description
WS-I Tool Location	Use the Browse button to specify the full path to the Java version of the WS-Interoperability Testing tools (for example, <code>C:\Program Files\WSI_Test_Java_Final_1.1\wsi-test-tools</code>). The WS-I testing tools are used to check a WSDL file for WS-I compliance. You can download them from www.ws-i.org [http://www.ws-i.org].
Results Type	Select the type of WS-I test results that you wish to view in the generated report from the drop-down list. You can select from <code>all</code> , <code>onlyFailed</code> , <code>notPassed</code> , or <code>notInfo</code> .
Message Entry	Specify whether message entries should be included in the report using the checkbox (selected by default).
Failure Message	Specify whether the failure message defined for each test assertion should be included in the report using the checkbox (selected by default).
Assertion Description	Specify whether the description of each test assertion should be included in the report using the checkbox (unselected by default).
Verbose Output	Specify whether verbose output is displayed in the Policy

<i>WS-I Setting</i>	<i>Description</i>
	Studio console window using the checkbox (unselected by default). To view the console window, select Window -> Show Console from the Policy Studio main menu.

For details on running the WS-I Testing Tools, see the [Web Service Repository](#) or [Global Schema Cache](#) topics.

Global Configuration

Overview

For convenience, the Policy Studio contains various global configuration options. For example, it includes libraries of users, X.509 certificates, and schemas that can be added globally and then referenced in filters and policies. This avoids the need to reconfigure details over and over again (for example, each time a schema or certificate is used).

The following global libraries are available in the Policy Studio, each of which are discussed briefly in the sections below:

- Enterprise Gateway Configuration Options
- Web Services Repository
- Processes
- Policies
- Certificates
- Enterprise Gateway Users
- Alerts
- External Connections
- Schema Cache
- Black list
- White list
- Caches

Enterprise Gateway Configuration Options

You can configure the underlying configuration settings for the Enterprise Gateway using the Policy Studio **Settings** menu option. This includes the following options:

- Deploy
- Default Settings
- Logging Settings
- Namespace Settings
- MIME/DIME Settings
- Change Passphrase

For more information, see the [Enterprise Gateway Configuration Options](#) topic.

Web Services Repository

The easiest way to secure a Web Service with the Enterprise Gateway is to import the WSDL (Web Services Description Language) file for the service using the Policy Studio. This creates a Service Handler for the Web Service, which is used to control and validate requests to the Web Service and responses from the Web Service.

The WSDL file is also added to the Web Services Repository, making sure to update the URL of the Web Service to point at the machine on which the Enterprise Gateway is running instead of that on which the Web Service is running. Consumers of the Web Service can then query the Enterprise Gateway for the WSDL file for the Web Service. The consumer then knows to route messages to the Enterprise Gateway instead of attempting to route directly to the Web Service, which most likely will not be available on a public IP address.

The Web Services Repository offers administrators a very simple way of securing a Web Service with minimal impact on

consumers of that service. Because of this, the Web Services Repository should be used as the primary method of setting up policies within the Policy Studio. For more information on using the Repository to setup policies, see the [Web Services Repository](#) tutorial.

Processes

A Process represents a single running instance of the Enterprise Gateway. It enables you to configure at least two interfaces: one for public traffic, and a second for listening for and serving configuration data. The configuration interface should rarely need to be updated. However, it is likely that you will want to add several HTTP interfaces. For example, you may want to add an HTTP interface and an SSL-enabled HTTPS interface.

Furthermore, you can also add features such as the following at the Process level:

- SMTP interfaces to configure email relay
- Policy execution schedulers to run policies at regular time intervals
- JMS listeners to listen for JMS messages
- Packet sniffers to inspect packets at the network level for logging and monitoring
- Directory scanners to scan messages dumped to the file system

Because the Enterprise Gateway can read messages from HTTP, SMTP, JMS, or a directory, this enables it to perform protocol translation. For example, the Enterprise Gateway can read a message from a JMS queue, and then route it on over HTTP to a Web Service. Similarly, the Enterprise Gateway can read XML messages that have been FTP-ed to a directory on the file system and send them to a JMS messaging system or route them over HTTP to a back-end system.

For more information on configuring processes, see the [Processes](#) tutorial.

Policies

A policy is made up of a sequence of modular, reusable message filters, each of which processes the message in a particular way. There are many categories of filters available, including authentication, authorization, content filtering, routing, and many more.

For example, a typical policy might contain an authentication filter, followed by several content-based filters (for example, Schema Validation, Threatening Content, Message Size, XML Complexity, and so on), and provided all configured filters run successfully, the message is routed on to the configured destination.

A policy can be thought of as a network of message filters. A message can traverse different paths through the network depending on what filters succeed or fail. This enables you to configure policies that, for example, route messages that pass one Schema Validation filter to one back-end system, and route messages that pass a *different* Schema Validation filter to a *different* system.

You can use *Policy Containers* to help manage your policies. A Policy Container is typically used to group together a number of similar policies (for example, all authentication policies) or to act as an umbrella around several policies that relate to a particular policy (for example, all policies for the `getQuote` Web Service).

A number of useful policies that ship with the Enterprise Gateway are found in the `Policy Library` Policy Container. This container is pre-populated with policies to return various types of faults to the client and policies to block certain types of threatening content, among others. You can also add your own policies to this container, and create your own Policy Containers as necessary to suit your own requirements.

The `Configuration` Policy Container is used to store the authentication and non-editable content-based filtering that is applied to configuration messages. This should only be changed under strict supervision from the Oracle support team.

Certificates

The Enterprise Gateway must be able to trust X.509 certificates to establish SSL connections with external servers, val-

idate XML Signatures, encrypt XML segments for certain recipients, and for other such cryptographic operations. Similarly, a private key is required to carry out certain other cryptographic operations, such as message signing and decrypting data.

The **Trusted Certificate Store** contains all the certificates that are considered to be trusted by the Enterprise Gateway. Certificates can be imported into or created by the Certificate Store. You can also assign a private key to the public key stored in a certificate, by importing the private key, or by generating one using the provided interface.

For more information on importing and creating certificates, see the [Trusted Certificate Store](#) topic.

Enterprise Gateway User Store

Users are mainly used for authentication purposes in the Enterprise Gateway. In this context, the **User Store** acts as a repository for user information against which users can be authenticated. You can also store user attributes for each user or user group. For example, you can then use these attributes when generating SAML attribute assertions on behalf of the user.

The [Enterprise Gateway Users](#) topic contains more details on how to create users, user groups, and attributes.

System Alerts

The Enterprise Gateway can send system alerts to various error reporting systems in the case of a policy error (for example, when a request is blocked by a policy). Alerts can be sent to a Windows Event Log, UNIX syslog, OPSEC firewall, SNMP NMS, or email recipient.

For more details on how to configure the Enterprise Gateway to send these alerts, see the [System Alerts](#) topic.

External Connections

The Enterprise Gateway can leverage your existing identity management infrastructure and avoid maintaining separate silos of user information. For example, if you already have a database full of user credentials, the Enterprise Gateway can authenticate requests against this database, rather than using its own internal user store. Similarly, the Enterprise Gateway can authorize users, lookup user attributes, and validate certificates against third-party identity management servers.

You can add each connection to an external system as a global **External Connection** in the Policy Studio so that it can be reused across all filters and policies. For example, if you create a circuit that authenticates users against an LDAP directory and then validates an XML signature by retrieving a public key from the same LDAP directory, it makes sense to create a global External Connection for that LDAP directory. You can then select the LDAP Connection in both the authentication and XML signature verification filters, rather than having to reconfigure it in both filters.

For example, you can use the External Connections interface to configure global connections such as the following:

- Authentication Repository Profiles
- Database Connections
- Kerberos Services
- LDAP Connections
- OCSP Connections
- XKMS Connections

You can also use External Connections in cases where you want to configure a group of related URLs. This is most useful in cases where you want to round-robin between a number of related URLs to ensure high availability. When the Enterprise Gateway is configured to use a *URL Connection Set* (instead of just a single URL), it round-robins between the URLs in the set.

For more information on configuring External Connections and Connection Sets, see the [External Connections](#) topic.

Schema Cache

The global **Schema Cache** contains all the XML Schemas that the Enterprise Gateway can use to validate incoming requests against. The **Schema Validation** filter validates the format of an incoming message against a schema from the cache. This ensures that only messages of the correct format are processed by the target system.

For instructions on how to import XML Schemas into the cache, see the [Schema Cache](#) topic. When you have imported your schemas, see the [Schema Validation](#) tutorial for instructions on how to validate XML messages against the schemas in the cache.

Black list and White list

The **White list** is a global library of regular expressions that can be used across several different filters. For example, the **Validate HTTP Headers**, **Validate Query String**, and **Validate Message Attributes** filters all use regular expressions from the **White list** to ensure that various parts of the request contain expected content.

The **White list** is pre-populated with regular expressions that can be used to identify common data formats, such as alphanumeric characters, dates, email addresses, IP addresses, and so on. For example, if a particular HTTP header is expected to contain an email address, the **Email Address** expression from the library can be run against the HTTP header to ensure that it contains an email address as expected. This is yet another way that the Enterprise Gateway can ensure that only the correct data reaches the Web Service.

While the **White list** contains regular expressions to identify valid data, the **Black list** contains regular expressions that are used to identify common attack signatures. For example, this includes expressions to scan for SQL injection attacks, buffer overflow attacks, ASCII control characters, DTD entity expansion attacks, and many more.

You can run various parts of the request message against the regular expressions contained in the **Black list** library. For example, the HTTP headers, request query string, and message (MIME) parts can be scanned for SQL injection attacks by selecting the SQL-type expressions from the **Black list**. The **Threatening Content** filter also uses regular expressions from the **Black list** to identify attack signatures in request messages.

For more details on running regular expressions, see the following topics:

- [Validate HTTP Headers](#)
- [Validate Query String](#)
- [Validate Message Attributes](#)
- [Threatening Content](#)

Caches

You can configure the Enterprise Gateway to cache responses from a back-end Web Service. For example, if the Enterprise Gateway receives two successive identical requests it can (if configured) take the response for this request from the cache instead of routing the request on to the Web Service and asking it to generate the response again.

As a result, excess traffic is diverted from the Web Service making it more responsive to requests for other services, the Enterprise Gateway is saved the processing effort of routing identical requests unnecessarily to the Web Service, and the client benefits from the far shorter response time.

Local caches can be configured for each running instance of the Enterprise Gateway. If you have deployed multiple Enterprise Gateways throughout your network, you can configure a distributed cache where cache events on one cache are replicated across all others. For example, if a response message is cached at one instance of the Enterprise Gateway, it is added to all other caches.

For more details on how to configure the Enterprise Gateway to use local and distributed caches, see the [Global Caches](#) topic.

Enterprise Gateway Configuration Options

Overview

You can configure the underlying configuration settings for the Enterprise Gateway using the the Policy Studio **Settings** menu option. You can also use the **Settings** and **Deploy** buttons in the toolbar. The following configuration options are available.

Deploy

Whenever you make changes to a filter or policy using the Policy Studio, you must deploy to the Enterprise Gateway for the changes to take affect. You can use the **Settings** -> **Deploy** menu option or the **Deploy** button in the toolbar to achieve this. If the server is processing a number of messages when the deploy command is issued, all of the messages are processed using the existing policy. New messages are queued until this batch of messages are completely processed. When the new policy data has been stored and loaded by the server, the queued messages are processed using the new policy.

Important:

To deploy to the Enterprise Gateway, the Policy Studio sends a deployment request to the Enterprise Gateway server. If necessary, you can configure the socket timeout value for this connection in the Policy Studio **Preferences** dialog, available from **Window** -> **Preferences**. For more details, see [Policy Studio Preferences](#).

Default Settings

The **Default Settings** entered in this dialog are applied to all instances of the Enterprise Gateway that use this particular configuration. For example, you can change the trace level, timeouts, cache sizes, and other such global information. For more details, see [Default Settings](#).

Note:

You can overwrite these settings at the Process level. You can do this by right-clicking the Process in the Policy Studio tree view, and selecting **Settings** -> **Custom**.

Default Logging Settings

This option enables you to configure the default logging behavior of the Enterprise Gateway. The Enterprise Gateway can be configured to log to a database, file, UNIX syslog, the system console, or the Oracle Logging Console. For more details, see [Logging Configuration](#).

MIME/DIME Settings

The Enterprise Gateway can filter MIME messages based on the content types (or MIME types) of the individual parts of the message. The **MIME/DIME Settings** dialog lists the default MIME types that the Enterprise Gateway can filter on. These types are then used by the **Content Types** filter to determine which MIME types to block or allow through to the back end Web Service. For more details, see [MIME/DIME Settings](#).

Namespace Settings

The **Namespace Settings** dialog can be used to determine the versions of SOAP, WSSE (Web Services Security), and WSU (Web Services Utility) that the Enterprise Gateway supports. For more details, see [Namespace Settings](#).

Change Passphrase

By default, Enterprise Gateway configuration data is stored unencrypted. However, you can encrypt certain sensitive information, such as passwords and private keys, using a passphrase. For more details, see [Setting the Encryption Passphrase](#).

Web Service Repository

Overview

The **Web Services Repository** stores information about Web Services whose definitions have been imported using Policy Studio or Service Manager. The WSDL files that contain these Web Services definitions are stored together with their related XML Schemas. Clients of the Web Service can then query the repository for the WSDL file, which they can use to build and send messages to the Web Service using the Enterprise Gateway.

When you import WSDL files into the repository, this auto-generates a **Service Handler** that is used to control and validate requests to the Web Service and responses from the Web Service. You can import the WSDL file from the file system, a URL, or from a UDDI registry. You can also test the WSDL file for compliance with Web Services Interoperability (WS-I) standards.

This topic describes how to test for WS-I compliance, how to import a Web Service definition into the repository, and shows what is created at each step.

Testing WS-I Compliance

Before importing the WSDL file, you can check it for compliance with the WS-I Basic Profile. The Basic Profile consists of a set of assertions and guidelines on how to ensure maximum interoperability between different implementations of Web Services. For example, there are recommendations on what style of SOAP to use (*document/literal*), how schema information is included in WSDL files, and how message parts are defined to avoid ambiguity for consumers of WSDL files.

The Policy Studio uses the Java version of the WS-Interoperability Testing Tools to test imported WSDL files for compliance with the recommendations in the Basic Profile. A report is generated showing which recommendations have passed and which have failed. While you can still import a WSDL file that does not comply with the Basic Profile, there is no certainty that consumers of the Web Service can use it without encountering problems.

Important Note:

Before you run the WS-I compliance test, you must ensure that the Java version of the Interoperability Testing Tools is installed on the machine on which the Policy Studio is running. You can download these tools from www.ws-i.org [<http://www.ws-i.org>].

To configure the location of the WS-I testing tools, select **Window -> Preferences** from the Policy Studio main menu. In the **Preferences** dialog, select the **WS-I Settings**, and browse to the location of the WS-I testing tools. You must specify the *full path* to these tools (for example, `C:\Program Files\WSI_Test_Java_Final_1.1\ws-i-test-tools`). For more details on configuring WS-I settings, see the [Policy Studio Preferences](#) topic.

Running the WS-I Compliance Test

To run the WS-I compliance test on a WSDL file, perform the following steps:

1. Select **Tools -> Run WS-I Compliance Test** from the Policy Studio main menu.
2. In the **Run WS-I Compliance Test** dialog, browse to the **WSDL File** or specify the **WSDL URL**.
3. Click **OK**. The WS-I Analysis tools run in the background in Policy Studio.

The results of the compliance test are displayed in your browser in a **WS-I Profile Conformance Report**. The overall result of the compliance test is displayed in the **Summary** section. The results of the WS-I compliance tests are grouped by type in the **Artifact: description** section. For example, you can access details for a specific port type, operation, or message by clicking the appropriate link in the **Entry List** table. Each **Entry** displays the results for the relevant WS-I Test Assertions.

Registering the WSDL File

The **Web Services Repository** is displayed as a top-level node in the Policy Studio **Policies** tree view. WSDL files are imported into Web Service Groups, which provide a convenient way of keeping groups of related Web Service definitions together. You can import a WSDL file into the default group by right-clicking the **Web Services** node, and selecting **Register Web Service**.

Alternatively, you can add a new Web Services group by right-clicking the default **Web Services** group, or the **Web Services Repository** node, and selecting **Add a new Web Services group**. When the new group is added, you can right-click it in the tree, and select **Register Web Service**.

Loading the WSDL File

In the **Import WSDL** wizard, the **Load WSDL** screen enables you to choose the WSDL location from the following options:

- File system
- URL
- UDDI registry

Select the appropriate option depending on the location of the WSDL that you wish to import. If you wish to retrieve a WSDL file from a UDDI directory, see the [Retrieving WSDL Files from a UDDI Registry](#) topic.

Click **Next**.

Selecting WSDL Operations

The **WSDL Operations** screen of the wizard displays all operations defined in the WSDL file. The **Relative Path**, **Binding**, and **Namespace** of each operation are also displayed.

Select the operations that you wish to create policy resolvers for. The Policy Studio uses the Web Service location, SOAP operation, and SOAP Action specified in the WSDL to create **Relative Path**, **SOAP Operation**, and **SOAP Action** policy resolving filters in the **Service Handler**.

Click **Next**.

WSDL Import Settings

In the **WSDL Import Settings** screen, the **Remove unselected operations from the WSDL** checkbox specifies whether the Policy Studio removes unselected operations from the WSDL file that is stored in the repository. As a result, removed operations are not exposed to clients that download the WSDL file for the Web Service.

Important Note:

When a request is made to the Enterprise Gateway for a WSDL file that has been imported into the **Web Services Repository**, it changes the address of the Web Service specified in the `location` attribute of the `<soap:address>` element to point to the machine on which the Enterprise Gateway is running, rather than the machine hosting the Web Service. This means that when a client downloads the WSDL file for the Web Service, it routes messages through the Enterprise Gateway instead of attempting to send messages directly to the Web Service, which typically is not available on a public IP address. For a detailed example, see [Publishing the WSDL](#).

Click **Next**.

Deploy Policy

The final screen in the wizard enables you to select where to deploy the newly created policy (or policies). The **Deploy Policy** screen displays a list of all available Services and their corresponding Relative Paths. Select a Relative Path under which the policy is deployed. All requests arriving on the selected path are dispatched to the newly created policy.

Click **Finish**.

Secure Virtual Service

When you click **Finish** in the wizard, the **Secure Virtual Service** dialog is displayed. This enables you to specify policies to enforce security between a client and the Enterprise Gateway. For more details, see [Securing a Virtual Service using Policies](#).

In addition, if the imported WSDL file contains WS-Policy assertions, you are prompted to configure settings to enforce security between the Enterprise Gateway and the Web Service. For more details, see [Configuring Security Policies from WSDL Files](#).

WSDL Import Summary

When you have finished configuring security policies, the **Summary** dialog displays details on the imported WSDL file. For example, this includes the location of the WSDL file, and a tab for each Web Service virtualized by the Enterprise Gateway. Each tab includes the path to the Web Service that is published by the Enterprise Gateway.

WSDL Import Summary Options

The **Summary** dialog also enables you to configure the following options on the tab for each Web Service:

Validation	If you wish to use a dedicated validation policy for all messages sent to the Web Service, select this checkbox, and click the browse button to configure a policy in the dialog. For example, this enables you to delegate to a custom validation policy used by multiple Web Services.
Routing	If you wish to use a dedicated routing policy to send all messages on to the Web Service, select this checkbox, and click the browse button to configure a policy in the dialog. For example, this enables you to delegate to a custom routing policy used by multiple Web Services.
WSDL Access Options	Select whether to make the WSDL for this Web Service available to clients. The Allow the Enterprise Gateway to publish WSDL to clients checkbox is selected by default. The published WSDL represents a virtualized view of the Web Service. Clients can retrieve the WSDL from the Enterprise Gateway, generate SOAP requests, and send them to the Enterprise Gateway, which routes them on to the Web Service. For more details, see Publishing the WSDL .

These options enable you to configure the underlying auto-generated Service Handler (**Web Service Filter**) without having to navigate to it in the **Policies** tree. These are the most commonly modified **Web Service Filter** options after importing a WSDL file. Changes made in the **Summary** dialog are visible in the underlying **Web Service Filter**. For more details, see the [Web Service Filter](#) topic.

You can also access these options from the **Services** tab after the WSDL file has been imported. Right-click the appropriate Web Service Resolver node, and select **Quick-Edit Policy** to display a dialog that enables you to configure these options.

What is Created?

What is created when you import the WSDL file depends on whether you configured a policy to enforce security between

the client and the Enterprise Gateway, and whether the WSDL file contains any WS-Policy annotations. However, assuming that the default options are selected in the **WSDL Import** wizard, the following list summarizes what is created by the wizard:

Web Service

A new Web Service tree node is created for each imported WSDL file in the **Policies** tree view. This tree node contains the WSDL file, together with any imported resources, such as other WSDL files or schema files. Click the new Web Service node in the tree to view the list of imported WSDL files and XML Schemas. The Schemas are listed by namespace. You can view the imported WSDL file by clicking the location of the WSDL file under the Web Service node in the tree. If you have selected to remove unselected operations from the WSDL, these operations are removed from the stored WSDL.

Web Service Resolver

A new Web Service Resolver node is created for each imported Web Service in the **Services** tree view. The Web Service Resolver is used to identify messages destined for this Web Service, and to map them to the **Service Handler (Web Service Filter)** for the Web Service.

Service Handler

A **Service Handler** for the Web Service is created under the generated policy in the **Policies** tree. This is used to control and validate requests to the Web Service and responses from the Web Service. The **Service Handler** is named after the Web Service (for example, **Service Handler for 'HelloWorldService'**). The **Service Handler** is a **Web Service Filter**, and is used to control the following:

- Routing
- Validation
- Message request/response processing
- WSDL options
- Monitoring options

For more details, see the [Web Service Filter](#) topic.

Policy Container

A container for the newly generated policies is created under the **Generated Circuits** node in the **Policies** tree. The new container is named after the service (for example, **Web Services.HelloWorldService**).

Policy

A policy for the Web Service is created in the generated policy container. The policy name includes the relative path to the service (for example, **/HelloWorldService/HelloWorld**). Clients can specify WSDL on a request query string to retrieve the WSDL file (for example, `http://localhost:8080/HelloWorldService/HelloWorld?WSDL`). For more details, see [Publishing the WSDL](#).

Security Policies

If you decided to configure a WS-policy to enforce security between the client and the Enterprise Gateway (as described earlier in [Secure Virtual Service](#)), or if the imported WSDL file contains WS-Policy assertions, a number of additional policies are automatically created in the generated policy container. These generated policies include the filters required to generate and/or validate the relevant security tokens (for example, SAML tokens, WS-Security Username tokens, and WS-Addressing headers). These policies perform the necessary cryptographic operations (for example, signing/verifying and encryption/decryption) to meet the security requirements of the specified policies.

Publishing the WSDL

When the WSDL has been imported into the **Web Services Repository**, it can be retrieved by clients. In effect, by importing the WSDL into the repository, you are *publishing* the WSDL. In this way, consumers of the services defined in the WSDL can learn how to communicate with those services by retrieving the WSDL for those services. However, to do this, the location of the service must be changed to reflect the fact that the Enterprise Gateway now sits between the client and the defined service.

For example, assume that the WSDL file states that a particular service resides at `http://www.example.com/services/myService`:

```
<service name="myService">
  <port binding="SoapBinding" name="mySample">
    <wsdl:address location="http://www.example.com/services/myService"/>
  </port>
</service>
```

When deployed behind the Enterprise Gateway, this URL is no longer accessible to consumers of the service. Because of this, clients must send SOAP messages through the Enterprise Gateway to access the service. In other words, they must now address the machine hosting the Enterprise Gateway instead of that directly hosting the service.

When the WSDL file has been published to the repository, clients can retrieve it. However, when returning the WSDL to the client, the Enterprise Gateway dynamically changes the value of the `location` attribute in the `service` element in the WSDL file to point to the machine on which the Enterprise Gateway resides. The details regarding the physical location of the Web Service are preserved in the **Connection** filters, which are responsible for routing messages on to the service.

Assuming that the Enterprise Gateway is running on port 8080 on a machine called `SERVICES`, the location specified in the exported WSDL file is changed to the following:

```
<service name="myService">
  <port binding="SoapBinding" name="mySample">
    <wsdl:address location="http://SERVICES:8080/services/myService"/>
  </port>
</service>
```

When the modified WSDL file is distributed to the client, it now routes messages to the machine hosting the Enterprise Gateway instead of attempting to directly access the Web Service.

Accessing the WSDL

For the client to access this modified WSDL file, the Policy Studio provides a WSDL retrieval facility whereby clients can query the **Web Services Repository** for the WSDL file for a particular Web Service. To do this, the client must pass the name `WSDL` on the query string to the Relative Path mapped to the policy for this Web Service.

For example, if the policy is deployed under `http://SERVICES:8080/services/getQuote`, the client can retrieve the WSDL for this Web Service by sending a request to `http://SERVICES:8080/services/getQuote?WSDL`. When the client has a copy of the updated WSDL file, it knows how to create correctly formatted messages for the service, and perhaps more importantly, it knows to route messages to the Enterprise Gateway rather than to the Web Service directly.

Publishing to UDDI

For details on how to publish a WSDL file registered in the Web Services Repository to a UDDI directory, see the [Publishing WSDL Files to a UDDI Registry](#) topic.

Setting the Encryption Passphrase

Encryption Passphrase Overview

By default, Enterprise Gateway configuration data is stored unencrypted. However, you can encrypt certain sensitive information, such as passwords and private keys, using a passphrase. When the passphrase has been set (and the data has been encrypted with it), you must enter the passphrase when connecting to the Enterprise Gateway with the Policy Studio, or when the Enterprise Gateway is starting up, so that the encrypted data can be decrypted.

Important Note:

It is *crucial* that you remember the passphrase when you change it. Failure to remember the passphrase results in the loss of private key data.

This topic describes how to set the passphrase for the first time, and then how to specify this passphrase when connecting to the Enterprise Gateway with the Policy Studio, in your Enterprise Gateway configuration file, or when the Enterprise Gateway is starting up. It also describes how to change the passphrase when it has been set initially.

Setting the Passphrase for the First Time

Complete the following steps to set the encryption passphrase for the first time:

1. Start the Policy Studio using the `polycystudio` command from your Policy Studio installation directory.
2. In the Policy Studio, click the **Open File** button in the toolbar on the **Home** tab. Alternatively, in the main menu, select **File -> Open File**.
3. Browse to the Enterprise Gateway `install_dir/conf/fed/configs.xml` file, and click **Finish**. Leave the **Passphrase Key** field blank when setting the passphrase for the first time.

Important Note:

If the Enterprise Gateway server is on a remote machine, you must copy its `install_dir/conf/fed` directory to the local machine that the Policy Studio is running on, before connecting to the `configs.xml` file. You can then set the passphrase, and copy the `/fed` directory back to the `install_dir/conf` directory on the machine where the Enterprise Gateway is running.

4. When the configuration file is loaded, in the Policy Studio main menu, select **Settings -> Settings -> Change Passphrase**. Alternatively, click the **Settings** button in the toolbar, and select **Change Passphrase**.
5. In the **Change Encryption Passphrase** dialog, enter the passphrase that you want to encrypt sensitive data with in the **New Passphrase** field. Make sure to leave the **Old Passphrase** field blank when setting the passphrase for the first time.
6. Click **OK**.

You must now specify the new passphrase when connecting to the Enterprise Gateway configuration with the Policy Studio and when the Enterprise Gateway starts up. The following sections describe how to do this for both components.

Connecting to the Enterprise Gateway with Policy Studio

When you have set the encryption passphrase for the Enterprise Gateway configuration data, you must specify this passphrase every time you connect to the Enterprise Gateway with the Policy Studio using the Enterprise Gateway's management interface. You can enter it in the **Passphrase Key** field of the **Connection Details** dialog, which is displayed when the Policy Studio is starting up.

It is important to note the different roles of the **Passphrase Key** and **Password** values that are specified on this screen:

Passphrase Key	Used to decrypt sensitive data (for example, private keys)
-----------------------	--

	that have already been encrypted. Not required by default, and only needed if the steps outlined above have been performed.
Password	Used to authenticate to the Enterprise Gateway's management interface using HTTP basic authentication. Required by default.

Connecting to Enterprise Gateway Configuration Data

For the Enterprise Gateway to read (decrypt) encrypted data from its configuration, it must be primed with the passphrase key. You can do this either by specifying the passphrase directly in the Enterprise Gateway configuration file, or by prompting for it when the Enterprise Gateway is starting up:

Specifying the Passphrase in the Configuration File

You can specify the passphrase directly in the Enterprise Gateway's configuration file. To do this, open the `userconfig.dtd` file in the `/conf` directory of your product installation. This DTD file contains values for general system settings, including the username and password to use to authenticate to the management interface exposed by the Enterprise Gateway, and also (if required) the passphrase key to use to decrypt encrypted Enterprise Gateway configuration data.

Typically, the passphrase is only entered directly in the file if the Enterprise Gateway must be started as a Windows Service or UNIX daemon. In this case, it is not possible for an administrator to enter a password manually when the Enterprise Gateway is starting. To avoid this problem, the password must be entered in the configuration file. To do this, specify the passphrase as the value for the `server.entitystore.secret` entity as follows, where "myPassphrase" is the encryption passphrase:

```
<!ENTITY server.entitystore.secret      "myPassphrase">
```

Prompting for the Passphrase on Server Startup

If you do not wish to specify the passphrase in the configuration file, and do not need to start the Enterprise Gateway as a Windows Service or UNIX daemon, you can configure the Enterprise Gateway to prompt the administrator for the passphrase when starting up. To do this, enter the special value "(prompt)" as the value of the `server.entitystore.secret` entity as follows:

```
<!ENTITY server.entitystore.secret      "(prompt)">
```

Important Note:

If you use this option, you must take care to remember the encryption passphrase. Failure to use the correct passphrase results in loss of private key data, and may prevent the Enterprise Gateway from functioning correctly.

Changing the Passphrase

If you have already specified a passphrase to use to encrypt the data, you can change this passphrase by clicking the **Settings** button in the toolbar, and selecting **Change Passphrase**. Complete the following fields on the **Change Encryption Passphrase** dialog:

Old Passphrase:

Enter the old passphrase that you wish to change in this field.

New Passphrase:

Enter the new passphrase here.

Old Passphrase:

Confirm the new passphrase in this field.

Retrieving WSDL Files from a UDDI Registry

Overview

A Web Services Description Language (WSDL) file defines the interface to a Web Service, or list of services. It lists the operations exposed by the service, the wire format for operation requests, and the data types in the request body. It also specifies the location of the Web Service as a URL that clients can access to use the exposed operations. For example, the Policy Studio can extract this information from the WSDL file to generate the following filters, which can be incorporated into policies:

- **Relative Path Resolver**
- **SOAPAction Resolver**
- **SOAP Operation Resolver**
- **Connection Handler**
- **Static Router**
- **Schema Validation**

You can use WSDL files to register Web Services in the **Web Services Repository** and to add XML Schemas to the global **Schema Cache**. The Policy Studio can retrieve a WSDL file from the file system, from a URL, or from a UDDI registry. This topic explains how to retrieve a WSDL file from a UDDI registry. For details on how to register WSDL files, see [Web Service Repository](#). For details on how to publish WSDL files, see [Publishing WSDL Files to a UDDI Registry](#).

You can also browse a UDDI registry in the Policy Studio directly without registering a WSDL file. In the **Policies** tab on the left, right-click the **Web Services Repository** node, and select **Browse UDDI Registry**.

UDDI: A Brief Introduction

Universal Description, Discovery and Integration (UDDI) is an OASIS-led initiative that enables businesses to publish and discover Web Services on the Internet. A business publishes services that it provides to a public XML-based registry so that other businesses can dynamically look up the registry and discover these services. Enough information is published to the registry to enable other businesses to find services and communicate with them. In addition, businesses can also publish services to a private or semi-private registry for internal use.

A business registration in a UDDI registry includes the following components:

- **Green Pages:**
Contains technical information about the services exposed by the business
- **Yellow Pages:**
Categorizes the services according to standard taxonomies and categorization systems
- **White Pages:**
Gives general information about the business, such as name, address, and contact information

You can search the UDDI registry according to a whole range of search criteria, which is of key importance to the Policy Studio. You can search the registry to retrieve the WSDL file for a particular service. The Policy Studio can then use this WSDL file to create a policy for the service, or to extract a schema from the WSDL to check the format of messages attempting to use the operations exposed by the Web Service.

For a more detailed description of UDDI, see the UDDI specification. In the meantime, the next section gives high-level definitions of some of the terms displayed in the Policy Studio interface.

UDDI Definitions

Because UDDI terminology is used in Policy Studio screens such as the **Import WSDL** wizard, the following list of definitions explains some common UDDI terms. For more detailed explanations, see the UDDI specification.

businessEntity

This represents all known information about a particular business (for example, name, description, and contact information). A `businessEntity` can contain a number of `businessService` entities. A `businessEntity` may have an `identifierBag`, which is a list of name-value pairs for identifiers, such as Data Universal Numbering System (DUNS) numbers or taxonomy identifiers. A `businessEntity` may also have a `categoryBag`, which is a list of name-value pairs used to tag the `businessEntity` with classification information such as industry, product, or geographic codes. There is no mapping for a WSDL item to a `businessEntity`. When a WSDL file is published, you must specify a `businessEntity` for the `businessService`.

businessService

A `businessService` represents a logical service classification, and is used to describe a Web Service provided by a business. It contains descriptive information in business terms outlining the type of technical services found in each `businessService` element. A `businessService` may have a `categoryBag`, and may contain a number of `bindingTemplate` entities. In the WSDL to UDDI mapping, a `businessService` represents a `wsdl:service`. A `businessService` has a `businessEntity` as its parent in the UDDI registry.

bindingTemplate

A `bindingTemplate` contains pointers to the technical descriptions and the access point URL of the Web Service, but does not contain the details of the service specification. A `bindingTemplate` may contain references to a number of `tModel` entities, which do include details of the service specification. In the WSDL to UDDI mapping, a `bindingTemplate` represents a `wsdl:port`.

tModel

A `tModel` is a Web service type definition, which is used to categorize a service type. A `tModel` consists of a key, a name, a description, and a URL. `tModels` are referred to by other entities in the registry. The primary role of the `tModel` is to represent a technical specification (for example, WSDL file). A specification designer can establish a unique technical identity in a UDDI registry by registering information about the specification in a `tModel`. Other parties can express the availability of Web services that are compliant with a specification by including a reference to the `tModel` in their `bindingTemplate` data.

This approach facilitates searching for registered Web services that are compatible with a particular specification. `tModels` are also used in `identifierBag` and `categoryBag` structures to define organizational identity and various classifications. In this way, a `tModel` reference represents a relationship between the keyed name-value pairs to the super-name, or namespace in which the name-value pairs are meaningful. A `tModel` may have an `identifierBag` and a `categoryBag`. In the WSDL to UDDI mapping, a `tModel` represents a `wsdl:binding` or `wsdl:portType`.

Identifier

The purpose of identifiers in a UDDI registry is to enable others to find the published information using more formal identifiers such as DUNS numbers, Global Location Numbers (GLN), tax identifiers, or any other kind of organizational identifiers, regardless of whether these are private or shared.

The following are identification systems used commonly in UDDI registries:

<i>Identification System</i>	<i>Name</i>	<i>tModel Key</i>
Dun and Bradstreet D-U-N-S Number	dnb-com:D-U-N-S	uuid:8609C81E-EE1F-4D5A-B202-3EB13AD01823
Thomas Registry Suppliers	thomasregister-com:supplierID	uuid:B1B1BAF5-2329-43E6-AE13-BA8E97195039

Category

Entities in the registry may be categorized according to categorization system defined in a `tModel` (for example, geographical region). The `businessEntity`, `businessService`, and `tModel` types have an optional `categoryBag`. This is a collection of categories, each of which has a name, value, and `tModel` key.

The following are categorization systems used commonly in UDDI registries:

<i>Categorization System:</i>	<i>Name:</i>	<i>tModel Key:</i>
UDDI Type Taxonomy	uddi-org:types	uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4
North American Industry Classification System (NAICS) 1997 Release	ntis-gov:naics:1997	uuid:C0B9FE13-179F-413D-8A5B-5004DB8E5BB2

Example tModel Mapping for WSDL portType

The following shows an example `tModel` mapped for a WSDL `portType`:

```
<tModel tModelKey="uuid:e8cf1163-8234-4b35-865f-94a7322e40c3" >
  <name>
    StockQuotePortType
  </name>
  <overviewDoc>
    <overviewURL>
      http://location/sample.wsdl
    </overviewURL>
  </overviewDoc>

  <categoryBag>
    <keyedReference
      tModelKey="uuid:d01987d1-ab2e-3013-9be2-2a66eb99d824"
      keyName="portType namespace"
      keyValue="http://example.com/stockquote/" />
    <keyedReference
      tModelKey="uuid:6e090afa-33e5-36eb-81b7-1ca18373f457"
      keyName="WSDL type"
      keyValue="portType" />
  </categoryBag>
</tModel>
```

In this example, the `tModel` name is the same as the local name of the WSDL `portType` (in this case, `StockQuotePortType`), and the `overviewURL` links to the WSDL file. The `categoryBag` specifies the WSDL namespace, and shows that the `tModel` is for a `portType`.

Configuring a Registry Connection

You first need to select the UDDI registry that you want to search for WSDL files. Complete the following fields to select or add a UDDI registry:

Select Registry:

Select an existing UDDI registry to browse for WSDL files from the **Registry** drop-down list. To configure the location of a new UDDI registry, click **Add**. Similarly, to edit an existing UDDI registry location, click **Edit**. Then configure the fields in the **Registry Connection Details** dialog. For more details, see [Connecting to a UDDI Registry](#).

Select Locale:

You can select an optional language locale from this list. The default is `No Locale`.

WSDL Search

When you have configured a UDDI registry connection, you can search the registry using a variety of different search options on the **Search** tab. **WSDL Search** is the default option. This enables you to search for the UDDI entries that the WSDL file is mapped to. You can also do this using the **Advanced Search** option. The following different types of WSDL searches are available:

WSDL portType (UDDI tModel):

Searches for a `uddi:tModel` that corresponds to a `wsdl:portType`. You can enter optional search criteria for specific categories in the `uddi:tModel` (for example, **Namespace of portType**).

WSDL binding (UDDI tModel):

Searches for a `uddi:tModel` that corresponds to a `wsdl:binding`. You can enter optional search criteria for specific categories in the `uddi:tModel` (for example, **Name of binding**, or **Binding Transport Type**).

WSDL service (UDDI businessService):

Searches for a `uddi:businessService` that corresponds to a `wsdl:service`. You can enter optional search criteria for specific categories in the `uddi:businessService` (for example, **Namespace of service**).

WSDL port (UDDI bindingTemplate):

Searches for a `uddi:bindingTemplate` that corresponds to a `wsdl:port`. This search is more complex because a `serviceKey` is required to find a `uddi:bindingTemplate`. This means that at least two queries are carried out, first to find the `uddi:businessService`, and another to find the `uddi:bindingTemplate`.

For example, a `bindingTemplate` contains a reference to the `tModel` for the `wsdl:portType`. You can use the `tModel` key to find all implementations (`bindingTemplates`) for that `wsdl:portType`. The search looks for `businessServices` that have `bindingTemplates` that refer to the `tModel` for the `wsdl:portType`. Then with the `serviceKey`, you can find the `bindingTemplate` that refers to the `tModel` for the `wsdl:portType`.

In all cases, click **Next** to start the WSDL search. The **Search Results** tree shows the `tModel` URIs as top-level nodes. These URIs are all WSDL URIs, and you can use these to generate policies on import by selecting the URI, and clicking the **Finish** button.

You can click any of the nodes in the tree to display detailed properties about that node in the table below the **Search Results** tree. The properties listed depend on the type of the node that is selected. You can also right-click a node to edit it (for example, add a description, add a category or identifier node, or delete a duplicate node).

Quick Search

The **Quick Search** option enables you to search the UDDI registry for a specific `tModel` name or category.

tModel Name:

You can enter a **tModel Name** for a fine-grained search. This is a partial or full name pattern with wildcard searching as specified by the *SQL-92 LIKE* specification. The wildcard characters are percent `%`, and underscore `_`, where an underscore matches any single character and a percent matches zero or more characters.

Categories:

You can select one of the following options to search by:

wsdlSpec	Search for <code>tModels</code> classified as <code>wsdlSpec</code> (<code>uddi-org:types category set to wsdlSpec</code>). This is the default.
Reusable WS-Policy Expressions	Search for <code>tModels</code> classified as reusable WS-Policy Expressions.
All	Search for all <code>tModels</code> .

Click **Next** to start the search. The **Search Results** tree shows the `tModel` URIs as top-level nodes. These URIs are all WSDL URIs, and you can use these to generate policies on import by selecting the URI, and clicking the **Finish** button.

You can click any of the nodes in the tree to display detailed properties about that node in the table below the **Search Results** tree. The properties listed depend on the type of the node that is selected. You can also right-click a node to edit it (for example, add a description, add a category or identifier node, or delete a duplicate node).

Name Search

The **Name Search** option enables you to search for a `businessEntity`, `businessService` or `tModel` by name. In the **Select Registry Data Type**, select one of the following UDDI entity levels to search for:

- **businessEntity**
- **businessService**
- **tModel**

You can enter a name in the **Name** field to narrow the search. You can also use wildcards in the name. The name applies to a `businessEntity`, `businessService`, or `tModel`, depending on which registry entity type has been selected. If no name is entered, all entities of the selected type are retrieved.

Click the **Search** button to start the search. The search results display the matching entities in the tree. For example, if you enter `MyTestBusiness` for **Name**, and select **businessEntity**, this searches for a `businessEntity` with the name `MyTestBusiness`. Child nodes of the matching `businessEntity` nodes are also shown. `tModels` are displayed in the results if any child nodes of the `businessEntity` refer to `tModels`. Only referred to `tModels` are displayed. The same applies if you search for a `businessService`. If you select `tModel`, and search for `tModels`, only `tModels` are displayed.

Important Note:

The `tModel` URIs shown in the resulting tree may not all be categorized as `wsdlSpec` according to the `uddi-org:types` categorization system. You can choose to load any of these URIs as a WSDL file, but you are warned if it is not categorized as `wsdlSpec`.

As before, you can click any node in the results tree to display properties about that node in the table. You can also right-click a node to edit it (for example, add a description, add a category or identifier node, or delete a duplicate node).

UDDI v3 Name Searches

By default, a UDDI v3 name search is an exact match. To perform a search using wildcards (for example, `%`, `_`, and so on), you must select the **approximateMatch** find qualifier in the **Advanced Options** tab. This applies to anywhere you enter a name for search purposes (for example, **Name Search**, **Quick Search**, and **Advanced Search**).

Advanced Search

The **Advanced Search** option enables you to search the UDDI registry using any combination of **Names**, **Keys**, **tModels**, **Discovery URLs**, **Categories**, and **Identifiers**. You can also specify the entity level to search for in the tree. All of these options combine to provide a very powerful search facility. You can specify search criteria for any of the categories listed above by right-clicking the folder node in the **Enter Search Criteria** tree, and selecting the **Add** menu option. You can enter more than one search criteria of the same type (for example, two **Key** search criteria).

Important Note:

The `tModel` URIs shown in the resulting tree may not all be categorized as `wsdlSpec` according to the `uddi-org:types` categorization system. You can choose to load any of these URIs as a WSDL file, but you are warned if it is not categorized as `wsdlSpec`.

The following options enable you to add a search criteria for each of the types listed in the **Enter Search Criteria** tree. All search criteria are configured by right-clicking the folder node, and selecting the **Add** menu option.

Names:

Enter a name to be used in the search in the **Name** field in the **Name Search Criterion** dialog. For example, the name could be the **businessEntity** name. The name is a partial or full name pattern with wildcards allowed as specified by the *SQL-92 LIKE* specification. The wildcard characters are percent %, and underscore _, where an underscore matches any single character and a percent matches zero or more characters. A name search criterion can be used for `businessEntity`, `businessService`, and `tModel` level searches.

Keys:

In the **Key Search Criterion** dialog, you can specify a key to search the registry for in the **Key** field. The key value is a Universally Unique Identifier (UUID) value for a registry object. You can use the **Key Search Criterion** on all levels of searches. If one or more keys are specified with no other search criteria, the keys are interpreted as the keys of the selected type of registry object and used for a direct lookup, instead of a find/search operation. For example, if you enter `key1` and `key2`, and select the `businessService` entity type, the search retrieves the `businessService` object with key `key1`, and another `businessService` with key `key2`.

If you enter a key with other search criteria, a key criterion is interpreted as follows:

- For a `businessService` entity lookup, the key is the `businessKey` of the services.
- For a `bindingTemplate` entity lookup, the key is the `serviceKey` of the binding templates.
- Not applicable for any other object type.

tModels:

You can enter a key in the **tModel Key** field on the **tModel Search Criterion** screen. The key entered should correspond to the UUID of the `tModel` associated with the type of object you are searching for. A `tModel` search criterion may be used for `businessEntity`, `businessService`, and `bindingTemplate` level searches.

Discovery URLs:

Enter a URL in the **Discovery URL** field on the **Discovery URL Search Criterion** dialog. The **Use Type** field is optional, but can be used to further fine-grain the search by type. You can use a Discovery URL search criterion for `businessEntity` level searches only.

Categories:

Select a previously configured categorization system from the **Type** drop-down list in the **Category Search Criterion** dialog. This is pre-populated with a list of common categorization systems. You can add a new categorization system using the **Add** button.

In the **Add/Edit Category** dialog, enter a **Name**, **Description**, and **UUID** for the new category type in the fields provided. When the categorization system has been selected or added, enter a value to search for in the **Value** field. The **Name** field is optional.

Identifiers:

Select a previously configured identification system from the **Type** drop-down list in the **Identifier Search Criterion** dialog. This is pre-populated with well-known identification systems. To add a new identification system, click the **Add** button.

In the **Add/Edit Identifier** dialog, enter a **Name**, **Description**, and **UUID** for the new identifier in the fields provided.

Select Registry Data Type:

Select one of the following UDDI entity levels to search for:

- **businessEntity**
- **businessService**
- **bindingTemplate**
- **tModel**

The search also displays child nodes of the matching nodes. `tModels` are also returned if they are referred to.

Advanced Options

This tab enables you to configure various aspects of the search conditions specified on the previous tabs. The following options are available:

<i>UDDI Find Qualifier:</i>	<i>Description:</i>
andAllKeys	By default, identifier search criteria are ORed together. This setting ensures that they are ANDed instead. This is already the default for <code>categoryBag</code> and <code>tModelBag</code> .
approximateMatch (v3)	This applies to a UDDI v3 registry only. Specifies wildcard searching as defined by the <code>uddi-org:approximatematch:SQL99 tModel</code> , which means approximate matching where percent sign (%) indicates any number of characters, and underscore (_) indicates any single character. The backslash character (\) is an escape character for the percent sign, underscore and backslash characters. This option adjusts the matching behavior for <code>name</code> , <code>keyValue</code> and <code>keyName</code> (where applicable).
binarySort (v3)	This applies to a UDDI v3 registry only. Enables greater speed in sorting, and causes a binary sort by name, as represented in Unicode codepoints.
bindingSubset (v3)	This applies to a UDDI v3 registry only. Specifies that the search uses only <code>categoryBag</code> elements from contained <code>bindingTemplate</code> elements in the registered data, and ignores any entries found in the <code>categoryBag</code> that are not direct descendents of registered <code>businessEntity</code> or <code>businessService</code> elements.
caseInsensitiveMatch (v3)	This applies to a UDDI v3 registry only. Specifies that the matching for <code>name</code> , <code>keyValue</code> and <code>keyName</code> (where applicable) should be performed without regard to case.
caseInsensitiveSort (v3)	This applies to a UDDI v3 registry only. Specifies that the result set should be sorted without regard to case. This overrides the default case sensitive sorting behavior.
caseSensitiveMatch (v3)	This applies to a UDDI v3 registry only. Specifies that the matching for <code>name</code> , <code>keyValue</code> and <code>keyName</code> (where applicable) should be case sensitive. This is the default behavior.
caseSensitiveSort (v3)	This applies to a UDDI v3 registry only. Specifies that the result set should be sorted with regard to case. This is the default behavior.
combineCategoryBags	Makes the <code>categoryBag</code> entries of a <code>businessEntity</code> behave as if all <code>categoryBags</code> found at the <code>businessEntity</code> level and in all contained or referenced <code>businessServices</code> are combined. Searching for a category yields a positive match on a registered business if any of the <code>categoryBags</code> contained in a <code>businessEntity</code> (including the <code>categoryBags</code> in contained or referenced <code>businessServices</code>) contain the filter criteria.
diacriticInsensitiveMatch (v3)	This applies to a UDDI v3 registry only. Specifies that matching for <code>name</code> , <code>keyValue</code> and <code>keyName</code> (where applicable) should be performed without regard to diacritics.

	Support for this qualifier by nodes is optional.
diacriticSensitiveMatch (v3)	This applies to a UDDI v3 registry only. Specifies that matching for <code>name</code> , <code>keyValue</code> and <code>keyName</code> (where applicable) should be performed with regard to diacritics. This is the default behavior.
exactMatch (v3)	This applies to a UDDI v3 registry only. Specifies that only entries with <code>name</code> , <code>keyValue</code> and <code>keyName</code> (where applicable) that exactly match the name argument passed in, after normalization, are returned. This qualifier is sensitive to case and diacritics where applicable. This is the default behavior.
exactNameMatch (v2)	This applies to a UDDI v2 registry only. Specifies that the name entered as part of the search criteria must exactly match the name specified in the UDDI registry.
orAllKeys	By default, <code>tModel</code> and category search criteria are ANDed. This setting ORs these criteria instead.
orLikeKeys	When a bag container contains multiple <code>keyedReference</code> elements (<code>categoryBag</code> or <code>identifierBag</code>), any <code>keyedReference</code> filters from the same namespace (for example, with the same <code>tModelKey</code> value) are ORed together rather than ANDed. For example, this enables you to search for any of these four values from this namespace, and any of these two values from this namespace.
serviceSubset	Causes the component of the search that involves categorization to use only the <code>categoryBags</code> from directly contained or referenced <code>businessServices</code> in the registered data. The search results return only those businesses that match based on this modified behavior, in conjunction with any other search arguments provided.
signaturePresent (v3)	This applies to a UDDI v3 registry only. This restricts the result to entities that contain, or are contained in, an XML Digital Signature element. The <code>Signature</code> element should be verified by the client. This option, or the presence of a <code>Signature</code> element, should only be used to refine a search result, and should not be used as a verification mechanism by UDDI clients.
sortByDateAsc (v3)	This applies to a UDDI v3 registry only. Sorts the results alphabetically in order of ascending date.
sortByDateDsc (v3)	This applies to a UDDI v3 registry only. Sorts the results alphabetically in order of descending date.
sortByNameAsc	Sorts the results alphabetically in order of ascending name.
sortByNameDsc	Sorts the results alphabetically in order of descending name.
suppressProjectedServices (v3)	This applies to a UDDI v3 registry only. Specifies that service projections must not be returned when searching for services or businesses. This option is enabled by default when searching for a service without a <code>businessKey</code> .
UTS-10 (v3)	This applies to a UDDI v3 registry only. Specifies sorting of results based on the Unicode Collation Algorithm on elements normalized according to Unicode Normalization

	Form C. A sort is performed according to the Unicode Collation Element Table in conjunction with the Unicode Collation Algorithm on the name field, and normalized using Unicode Normalization Form C. Support for this qualifier by nodes is optional.
--	---

Publish

Click the **Publish** radio button to view the **Published UDDI Entities Tree View**. This enables you to manually publish UDDI entities to the specified UDDI registry (for example, `businessEntity`, `businessService`, `bindingTemplate`, and `tModel` entities). You must already have the appropriate permissions to write to the UDDI registry.

Adding a businessEntity

To add a business, perform the following steps:

1. Right-click the tree view, and select **Add businessEntity**.
2. In the **Business** dialog, enter a **Name** and **Description** for the business.
3. Click **OK**.
4. You can right-click the new `businessEntity` node to add child UDDI entities in the tree (for example, `businessService`, `Category`, and `Identifier` entities).

Adding a tModel

To add a `tModel`, perform the following steps:

1. Right-click the tree view, and select **Add tModel**.
2. In the **tModel** dialog, enter a **Name**, **Description**, and **Overview URL** for the `tModel`. For example, you can use the **Overview URL** to specify the location of a WSDL file.
3. Click **OK**.
4. You can right-click the new `tModel` node to add child UDDI entities in the tree (for example, `Category` and `Identifier` entities).

As before, you can click any node in the results tree to display properties about that node in the table. You can also right-click a node to edit it (for example, add a description, add a category or identifier node, or delete a duplicate node). At any stage, you can click the **Clear** button on the right to clear the entire contents of the tree. This does not delete the contents of the registry.

For more details on UDDI entities such as `businessEntity` and `tModel`, see the [UDDI Definitions](#) section. For details on how to publish Web Services automatically using a wizard, see [Publishing WSDL files to a UDDI Registry](#).

For more details on UDDI entities such as `businessEntity` and `tModel`, see the [UDDI Definitions](#) section.

Connecting to a UDDI Registry

Overview

This topic explains how to configure a connection to a UDDI registry in the **Registry Connection Details** dialog. It explains how to configure connections to UDDI v2 and UDDI v3 registries, also how to secure a connection over SSL.

Configuring a Registry Connection

Configure the following fields in the **Registry Connection Details** dialog:

Registry Name:

Enter the display name for the UDDI registry.

UDDI v2:

Select this option to use UDDI v2.

UDDI v3:

Select this option to use UDDI v3.

Inquiry URL:

Enter the URL on which to search the UDDI registry (for example, `http://HOSTNAME:PORT/uddi/inquiry`).

Publish URL:

Enter the URL on which to publish to the UDDI registry, if required (for example, `http://HOSTNAME:PORT/uddi/publishing`).

Security URL (UDDI v3):

For UDDI v3 only, enter the URL for the security service, if required (for example, `http://HOSTNAME:PORT/uddi/security.wsdl`).

Important Note:

For UDDI v3, the **Inquiry URL**, **Publish URL**, and **Security URL** specify the URLs of the WSDL for the inquiry, publishing, and security Web services that the registry exposes. These fields can use the same URL if the WSDL for each service is at the same URL.

For example, a WSDL file at `http://HOSTNAME:PORT/uddi/uddi_v3_registry.wsdl` can contain three URLs (`http://HOSTNAME:PORT/uddi/inquiry`, `http://HOSTNAME:PORT/uddi/publishing`, and `http://HOSTNAME:PORT/uddi/security`). These are the service endpoint URLs that the Policy Studio uses to browse and publish to the registry. These URLs are not set in the connection dialog, but discovered using the WSDL. However, for UDDI v2, WSDL is *not* used to discover the service endpoints, so you must enter these URLs directly in the connection dialog.

Max Rows:

Enter the maximum number of entries returned by a search. Defaults to 20.

Registry Authentication:

The registry authentication settings are as follows:

Type	This optional field applies to UDDI v2 only. The only supported authentication type is UDDI_GET_AUTHTOKEN.
Username	Enter the username required to authenticate to the registry, if required.
Password	Enter the password for this user, if required.

The username and password apply to UDDI v2 and v3. These are generally required for publishing, but depend on the configuration on the registry side.

HTTP Proxy:

The HTTP proxy settings apply to UDDI v2 and v3:

Proxy Host	If the UDDI registry location entered above requires a connection to be made through an HTTP proxy, enter the host name of the proxy.
Proxy Port	If a proxy is required, enter the port on which the proxy server is listening.
Username	If the proxy has been configured to only accept authenticated requests, the Policy Studio sends this username and password to the proxy using HTTP Basic authentication.
Password	Enter the password to use with the username specified in the field above.

HTTPS Proxy:

The HTTPS proxy settings apply to UDDI v2 and v3:

SSL Proxy Host	If the Inquiry URL or Publish URL uses the HTTPS protocol, the SSL proxy host entered is used instead of the HTTP proxy entered above. In this case, the HTTP proxy settings are not used.
Proxy Port	Enter the port that the SSL proxy is listening on.

Securing a Connection to a UDDI Registry

You may wish to communicate with the UDDI registry over SSL. All communication may not need to be over SSL (for example, you may wish publish over SSL, and perform inquiry calls without SSL). For UDDI v2 and v3, you can use a mix of `http` and `https` URLs for WSDL and service address locations.

You can specify some or all of the **Inquiry URL**, **Publish URL**, and **Security URL** settings as `https` URLs. For example, with UDDI v3, you could use a single URL like the following:

```
https://HOSTNAME:PORT/uddi/wsdl/uddi_v3_registry.wsdl
```

If any URLs (WSDL or service address location) use `https`, you must configure the Policy Studio so that it trusts the registry SSL certificate.

Configuring the Policy Studio to Trust a Registry Certificate

For an SSL connection, you must configure the registry server certificate as a trusted certificate. Assuming mutual authentication is not required, the simplest way to configure an SSL connection between the Policy Studio and UDDI registry is to add the registry certificate to the Policy Studio default truststore (the `cacerts` file). You can do this by performing the following steps in the Policy Studio:

1. Click the **Certificates** tab on the left.
2. Click **Create/Import**, and click **Import Certificate**.
3. Browse to the UDDI registry SSL certificate file, and click **Open**.
4. Click **Use Subject** on the right of the **Alias Name** field, and click **OK**. The registry SSL Certificate is now imported into the **Certificate Store**, and must be added to the Java keystore.
5. Click **Keystore** on the **Certificate** screen.
6. Click **Browse** next to the **Keystore** field.
7. Browse to the following file:
INSTALL_DIR/policystudio/jre/lib/security/cacerts
8. Click **Open**, and enter the **Keystore password**. The default password is: `changeit`.
9. Click **Add to Keystore**.
10. Browse to the registry SSL certificate imported earlier, select it, and click **OK**.
11. Restart the Policy Studio. You should now be able to connect to the registry over SSL.

Configuring Mutual SSL Authentication

If mutual SSL authentication is required (if the Policy Studio must authenticate to the registry), the Policy Studio must have an SSL private key and certificate. In this case, you should create a keystore containing the Policy Studio key and certificate. You must configure the Policy Studio to load this file. For example, edit the `INSTALL_DIR/policystudio/policystudio.ini` file, and add the following arguments:

```
-Djavax.net.ssl.keyStore=/home/oracle/osr-client.jks
-Djavax.net.ssl.keyStorePassword=changeit
```

This example shows an `osr-client.jks` keystore file used with Oracle Service Registry (OSR), which is the UDDI registry provided by Oracle.

Note:

You can also use the Policy Studio to create a new keystore (`.jks`) file. Click **New keystore** instead of browsing to the `cacerts` file as described earlier.

Publishing WSDL Files to a UDDI Registry

Overview

You can register Web Services in the **Web Services Repository** using Web Services Description Language (WSDL) files. The Policy Studio can retrieve a WSDL file from the file system, from a URL, or from a UDDI registry. When you have registered a WSDL file in the **Web Services Repository**, you can use the **Publish WSDL Wizard** to publish the WSDL file to a UDDI registry. You can also use the **Find WSDL Wizard** to search for the selected WSDL file in a UDDI registry. This topic explains how to perform both of these tasks.

For background information and an introduction to general UDDI concepts, see [Retrieving WSDL Files from a UDDI Registry](#). For details on how to register WSDL files, see [Web Service Repository](#).

Finding WSDL Files

You can search a UDDI registry to determine if a Web Service is already published in the registry. To search for a selected WSDL file in a specified UDDI registry, perform the following steps:

1. In the **Policies** tab on the left, expand the **Web Services Repository** tree node.
2. Right-click a WSDL node (for example, `http://HOSTNAME/TestService/StockQuote.svc?WSDL`, where `HOSTNAME` is the endpoint host of the Web service).
3. Select **Find in UDDI Registry** to launch the **Find WSDL Wizard**.
4. In the **Find WSDL** screen, select a **UDDI Registry** from the list. You can add or edit a registry connection using the buttons provided. For details on configuring a registry connection, see [Connecting to a UDDI Registry](#).
5. You can select an optional language **Locale** from the list. The default is `No Locale`.
6. Click **Next**. The **WSDL Found in UDDI Registry** screen displays the result of the search in a tree. The **Node Counts** field shows the total numbers of each UDDI entity type returned from the search (`businessEntity`, `businessService`, `bindingTemplate`, and `tModel`).
7. You can right-click to edit a UDDI entity node in the tree, if necessary (for example, add a description, add a category or identifier node, or delete a duplicate node).
8. Click the **Refresh** button to run the search again.
9. Click **Finish**.

The **Find WSDL Wizard** provides a quick and easy way of finding a selected WSDL file published in a UDDI registry. For more fine-grained ways of searching a UDDI registry (for example, for specific WSDL or UDDI entities), see [Retrieving WSDL Files from a UDDI Registry](#).

Publishing WSDL Files

To publish a WSDL file registered in the **Web Services Repository** to a UDDI registry, perform the following steps:

1. In the **Policies** tab on the left, expand the **Web Services Repository** tree node.
2. Right-click a WSDL node (for example, `http://HOSTNAME/TestService/StockQuote.svc?WSDL`, where `HOSTNAME` is the host from which the Web service is registered).
3. Select **Publish WSDL to UDDI Registry** to launch the **Publish WSDL Wizard**.
4. Perform the steps in the wizard screens described in the next sections.

Step 1: Enter Virtualized Service Address and WSDL URL for Publishing in UDDI Registry

When you register a WSDL file in the Web Services Repository, the Enterprise Gateway exposes a *virtualized* version of the Web Service. The host and port for the Web Service are changed dynamically to point to the machine running the Enterprise Gateway. The client can then retrieve the WSDL for the virtualized Web Service from the Enterprise Gateway,

without knowing its real location.

This screen enables you to optionally override the service address locations in the WSDL file with the virtualized addresses exposed by the Enterprise Gateway. You can also override the WSDL URL published to the UDDI registry. Complete the following fields:

Mapping of Service Addresses to Virtualized Service Addresses

You can enter multiple virtual service address mappings for each service address specified in the selected WSDL file. If you do not enter a mapping, the original address location in the WSDL file is published to the UDDI registry. If one or more mappings are provided, corresponding UDDI `bindingTemplates` are published in the UDDI registry, each with a different access point (virtual service address). This enables you to publish the access points of a service when it is exposed on different ports/schemes using the Enterprise Gateway.

When you launch the wizard, the mapping table is populated with a row for each `wsdl:service`, `wsdl:port`, `soap:address`, `soap12:address`, or `http:address` in the selected WSDL file. To modify an existing entry, select a row in the table, and click **Edit**. Alternatively, click **Add** to add an entry. In the **Virtualize Service Address** dialog, enter the virtualized service address. For example, if the Enterprise Gateway is running on a machine named `roadrunner`, the new URL on which the Web service is available to clients is: `http://roadrunner:8080/TestServices/StockQuote.svc`.

WSDL URL:

You can enter a WSDL URL to be published to the UDDI registry in the corresponding `tModel overviewURL` fields. If you do not enter a URL, the WSDL URL in the **Original WSDL URL** field is used. For example, an endpoint service is at `http://coyote.qa.acmecorp.com/TestService/StockQuote.svc`. Assume this service is virtualized in the Enterprise Gateway and exposed at `http://GATEWAY:8080/TestService/StockQuote.svc`, where `GATEWAY` is the machine on which the Enterprise Gateway is running. The `http://GATEWAY:8080/TestService/StockQuote.svc` URL is entered as the virtual service address, and `http://GATEWAY:8080/TestService/StockQuote.svc?WSDL` is entered as the WSDL URL to Publish.

Note:

If incorrect URLs are published, you can edit these in the UDDI tree in later steps in this wizard, or when browsing the registry.

Click **Next** when finished.

Step 2: View WSDL to UDDI Mapping Result

You can use this screen to view the unpublished mapping of the WSDL file to a UDDI registry structure. You can also edit a specific mapping in the tree view. This screen includes the following fields:

Mapping of WSDL to a UDDI Registry Structure:

The unpublished mappings from the WSDL file to the UDDI registry are displayed in the table. For example, this includes the relevant `businessService`, `bindingTemplate`, `tModel`, `Identifier`, `Category` mappings. You can select a tree node to display its values in the table below.

You can optionally edit the values for a specific mapping in the table (for example, update a value, or add a key or description for the selected UDDI entity). You can also right-click a tree node to edit it (for example, add a description, add a category or identifier node, or delete a duplicate node).

Retrieve service address from WSDL instead of bindingTemplate access point:

When selected, this ensures that the `bindingTemplate` access point does not contain the service port address, and is set to WSDL instead. This means that you must retrieve the WSDL to get the service access point. When selected, the `bindingTemplate` contains an additional `tModelInstanceInfo` that points to the `uddi:uddi.org.wsdl:address tModel`. This option is not selected by default.

Include WS-Policy as:

When selected, you can choose one of the following options to specify how WS-Policy statements in the WSDL file are included in the registry:

Remote Policy Expressions	Each WS-Policy URL in the WSDL that is associated with a mapped UDDI entity is accessed remotely. For example, a <code>businessService</code> is categorized with the <code>uddi:w3.org:ws-policy:v1.5:attachment:remotepolicyreference tModel</code> where the <code>keyValue</code> holds the remote WS-Policy URL. This is the default option.
Reusable Policy Expressions	Each WS-Policy URL in the WSDL that is associated with a mapped UDDI entity has a separate <code>tModel</code> published for it. Other UDDI entities (for example, <code>businessService</code>) can then refer to these <code>tModels</code> . These are reusable because UDDI entities published in the future can also use these <code>tModels</code> . You can do this in Step 4: Select a duplicate publishing approach by selecting the Reuse duplicate tModels option.

Click **Next** when finished.

Step 3: Select a Registry for Publishing

Use this screen to select a UDDI registry in which to publish the WSDL to UDDI mapping. Complete the following fields:

Select Registry

Select an existing UDDI registry to browse for WSDL files from the **Registry** drop-down list. To configure the location of a new UDDI registry, click **Add**. Similarly, to edit an existing UDDI registry location, click **Edit**. For details on how to configure a UDDI connection, see [Connecting to a UDDI Registry](#).

Select Locale:

You can select an optional language locale from this list. The default is `No Locale`.

Click **Next** when finished.

Step 4: Select a Duplicate Publishing Approach

This screen is displayed only if mapped WSDL entities already exist in the UDDI registry. Otherwise, the wizard skips to step 5. This screen includes the following fields:

Select Duplicate Mappings

The **Mapped WSDL to publish** pane on the left displays the unpublished WSDL mappings from Step 2. The **Duplicates for WSDL mappings in UDDI registry** pane on the right displays the nodes already published in the registry. The **Node List** at the bottom right shows a breakdown of the duplicate nodes.

Edit Duplicate Mappings

You can eliminate duplicate mappings by right-clicking a tree node in the right or left pane, and selecting edit to update a specific mapping in the dialog. Select the **Refresh** button on the right to run the search again, and view the updated **Node List**. Alternatively, you can configure the options in the next field.

Select Publishing Approach for Duplicate Entries:

Select one of the following options:

Reuse duplicate tModels	Publishes the selected entries from the tree on the left, and reuses the selected duplicate entries in the tree on the right. This is the default option. Some or all duplicate <code>tModels</code> (for example, for <code>portType</code> , <code>binding</code> , and reusable WS-Policy expressions) that already exist in the re-
--------------------------------	---

	gistry can be reused. This means that a new <code>businessService</code> that points to existing <code>tModels</code> is published. Any entries selected on the left are published, and any referred to <code>tModels</code> on the left now point to selected duplicate <code>tModels</code> on the right. By default, this option selects all <code>businessServices</code> on the left, and all duplicate <code>tModels</code> on the right. If there is more than one duplicate <code>tModels</code> , only the first is selected.
Overwrite duplicates	Publishes the selected entries from the tree on the left, and overwrites the selected duplicate entries in the tree on the right. When a UDDI entity is overwritten, its UUID key stays the same, but all the data associated with it is overwritten. This is not just a transfer of additions or differences. You can also overwrite some duplicates and create some new entries. By default, this option selects all <code>businessServices</code> and <code>tModels</code> on the left and all duplicate <code>businessServices</code> and <code>tModels</code> on the right. If there is more than one duplicate, only the first is selected. The default overwrites all selected duplicates and does not create any new UDDI entries, unless there is a new referred to <code>tModel</code> (for example, for a reusable WS-Policy expression).
Ignore duplicates	Publishes the selected entries from the tree on the left, and ignores all duplicates. You can proceed to publish the mapped WSDL to UDDI data. New UDDI entries are created for each item that is selected in the tree on the left.

Click **Next** when finished.

Note:

If you select duplicate `businessServices` in the tree, and select **Overwrite duplicates**, the wizard skips to Step 6 when you click **Next**.

Step 5: Create or Search for Business

Use this screen to specify a `businessEntity` for the Web Service. You can create a new `businessEntity` or search for an existing one in the UDDI registry. Complete the following fields:

Creating a New `businessEntity`

This is the default option. Enter a **Name** and **Description** for the `businessEntity`, and click **Publish**.

Searching for an Existing `businessEntity`

To search for an existing `businessEntity` name, perform the following steps:

1. Select the **Search for an existing `businessEntity` in the UDDI registry** option.
2. In the **Search** tab, ensure the **Name Search** option is selected.
3. Enter a **Name** option (for example, `Acme Corporation`).

Alternatively, you can select the **Advanced Search** option to search by different criteria such as **Keys**, **Categories**, or **tModels**. For more details, see [Retrieving WSDL Files from a UDDI Registry](#).

Advanced Options

You can also select a range of search options on the **Advanced** tab (for example, **Exact match**, **Case sensitive**, or **Ser-**

vice subset). For more details, see [Retrieving WSDL Files from a UDDI Registry](#).

The **Node Counts** field shows the total numbers of each UDDI entity type returned from the search (`businessEntity`, `businessService`, `bindingTemplate`, and `tModel`).

Click **Next** when finished.

Step 6: Publish WSDL

Use this screen to publish the WSDL to the UDDI registry.

Selected businessEntity for Publishing:

This field displays the name and `tModel` key of the `businessEntity` to be published. Click the **Publish WSDL** button on the right.

Published WSDL:

This field displays the tree of the UDDI mapping for the WSDL file. You can right-click to edit or delete any nodes in the tree if necessary, and click **Refresh** to run the search again. Click **Publish WSDL** to publish your updates.

Click **Finish**.

Default Settings

Overview

The **Default Settings** dialog enables you to set several global configuration settings to optimize the behavior of the Enterprise Gateway for your environment. You can overwrite these settings at the Process level by right-clicking the Process node in the Policy Studio tree, and selecting **Settings -> Custom**.

To configure the **Default Settings**, in the Policy Studio main menu, select **Settings -> Settings -> Default Settings**. This displays the **Default Settings** dialog. Alternatively, in the toolbar, click the drop-down option on the **Settings** button, and select **Default Settings**.

After changing any of the settings, you must deploy to the Enterprise Gateway for the changes to be enforced. You can do this in the Policy Studio main menu by selecting **Settings -> Deploy**. Alternatively, click the **Deploy** button in the toolbar, or press F6.

Settings

You can configure the following settings in the **Default Settings** dialog:

Setting	Purpose
Active Timeout	When the Enterprise Gateway receives a large HTTP request, it reads the request off the network when it becomes available. If the time between reading successive blocks of data exceeds the Active Timeout , the Enterprise Gateway closes the connection. This guards against a host closing the connection in the middle of sending data. For example, if the host's network connection is pulled out of the machine while in the middle of sending data to the Enterprise Gateway. When the Enterprise Gateway has read all the available data off the network, it waits the Active Timeout period of time before closing the connection. Note: You can configure this setting on a per-host basis using the Remote Hosts interface.
Administrator Role	Configures a special system administrator role that provides protection for the specified role (a user with this role cannot remove themselves from this role). This ensures that there is always at least one user in the system with this role. By default, the configured Administrator Role allows access to all management services. Defaults to the <code>Administrators</code> role. For more details, see the Configuring Role-Based Access Control topic.
Date Format	Configures the format of the date for the purposes of tracing, logging, and reporting. For more information, see http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html
Cache Refresh Interval	Configures the number of seconds that the server caches data loaded from an external source (external database, LDAP directory, and so on) before refreshing the data from that source. The default value is 5 seconds. If you do not wish any caching to occur, set this value to 0.
Idle Timeout	The Enterprise Gateway supports HTTP 1.1 persistent connections. The Idle Timeout is the time that the Enterprise

Setting	Purpose
	<p>Gateway waits after sending a message over a persistent connection before it closes the connection. Typically, the host tells the Enterprise Gateway that it wants to use a persistent connection. The Enterprise Gateway acknowledges this instruction and decides to keep the connection open for a certain amount of time after sending the message to the host. If the connection is not reused within the Idle Timeout period, the Enterprise Gateway closes the connection.</p> <p>Note: You can configure this setting on a per-host basis using the Remote Hosts interface.</p>
LDAP Service Provider	<p>Specifies the service provider used for looking up an LDAP server (for example, <code>com.sun.jndi.ldap.LdapCtxFactory</code>). The provider is typically used to connect to LDAP directories for certificate and attribute retrieval.</p>
Maximum Memory per Request	<p>The maximum amount of memory allocated to each request.</p> <p>Note: You can configure this setting on a per-host basis using the Remote Hosts interface.</p>
Policy Director Process Connect Timeout	<p>When the Policy Director connects to a process to deploy or retrieve configuration, the connection fails if no activity happens on that connection for the duration specified by this setting (in seconds). The default is 300 seconds (5 minutes). Increase this setting if very large configurations are being deployed to the process, or configurations that may stall the process on initialization (for example, due to database timeout).</p>
Policy Director Process Ping Connect Timeout	<p>If an attempt to connect to a known process to ping it stalls for greater than this configured time in seconds, the attempt is aborted. This time should always be configured to be less than the process ping interval. Defaults to 30 seconds.</p>
Policy Director Process Ping Interval	<p>Specifies the polling interval in seconds for contacting each known process to check it is alive and retrieve its configuration status. Defaults to 60 seconds.</p>
Policy Director User Session Timeout	<p>This setting logs out a Policy Director user if there is no activity in the specified time period in seconds. Defaults to 1800 seconds (30 minutes).</p>
Send desired servername to server during TLS negotiation	<p>Specifies whether to add a field to outbound TLS/SSL calls that shows the name that the client used to connect. For example, this can be useful if the server handles several different domains, and needs to present different certificates depending on the name that the client used to connect.</p>
Realm	<p>Specifies the realm for authentication purposes.</p>
Schema Pool Size	<p>Sets the size of the Schema Parser pool.</p>
Server Brand	<p>Specifies the branding to be used in the Enterprise Gateway.</p>

Setting	Purpose
LDAP Time Out	The timeout in milliseconds for the LDAP connection. If a connection has not been created in this time frame, the operation times out. Similarly, if a lookup operation has not succeeded in this time frame, the operation fails. If this setting is not configured, or set to zero, the TCP timeout for the platform is used, which defaults to 3 minutes.
Token Drift Time	The number of seconds drift allowed for WS-Security tokens. This is important in cases where the Enterprise Gateway is checking the date on incoming WS-Security tokens. It is likely that the machine on which the token was created is out-of-sync with the machine on which the Enterprise Gateway is running. The drift time allows for differences in the respective machine clock times.
Server's SSL cert's name must match name of requested server	Ensures that the certificate presented by the server matches the name of the host address being connected to. This prevents host spoofing and man-in-the-middle attacks. This setting is enabled by default.
Use Validation on SAX Parsers	Disabled by default for performance reasons. However, to perform SAX validation when parsing XML messages, you can select this setting.
Super User Role	<p>Configures a special system administrator role that has special privileges in the system, for example:</p> <ul style="list-style-type: none"> • Add, delete, and update other users. • Reset user passwords. • Add another superuser. • Perform actions only allowed by a Configuration Profile owner (for example, transfer ownership to another user). • Perform actions only allowed by a Process owner (for example, deploy a new Configuration Profile to a Process owned by another user). <p>Defaults to the <code>Administrators</code> role. This means that the default <code>admin</code> user has the PD superuser privilege, and it cannot delete itself. However, you can also separate the PD superuser privilege from the <code>Administrators</code> role. For more details, see the Configuring Role-Based Access Control topic.</p>
Trace Level	Enables you to set the trace level for the Enterprise Gateway at runtime. Select the appropriate option from the Trace Level drop-down list.

Namespace Settings

Overview

The Enterprise Gateway exposes a global setting that allows you to configure what versions of the SOAP and WSSE specifications it supports. Furthermore, it allows you to specify what attribute is used to identify the XML Signature referenced within a SOAP message.

To configure the **Namespace Settings**, in the Policy Studio main menu, select **Settings -> Settings -> Namespace Settings**. This displays the **Namespace Settings** dialog. Alternatively, in the toolbar, click the drop-down option on the **Settings** button, and select **Namespace Settings**.

Signature ID Attribute

The **Signature ID Attribute** tab allows you to list the supported attributes that can be used by the Enterprise Gateway to identify a Signature reference within an XML message.

An XML-signature `<signedInfo>` section may reference signed data via the `URI` attribute. The `URI` value may contain an id that identifies data in the message. The referenced data will hold the "URI" field value in one of its attributes.

By default, the server will use the "Id" attribute for each of the WSSE namespaces listed above to locate referenced signed data. The following sample XML Signature illustrates the use of the "Id" attribute:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <dsig:Signature id="Sample" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
      <dsig:SignedInfo>
        ...
        <dsig:Reference URI="#Oracle:sLmDCph3tGZ10">
          ...
        </dsig:Reference>
      </dsig:SignedInfo>
    </dsig:Signature>
  </soap:Header>
  <soap:Body>
    <getProduct wsu:Id="Oracle:sLmDCph3tGZ10"
      xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
      <Name>SOA Test Client</Name>
      <Company>Company</Company>
    </getProduct>
  </soap:Body>
</soap:Envelope>
```

It is clear from this example that the Signature reference identified by the `URI` attribute of the `<Reference>` element refers to the nodeset identified with the `Id` attribute, i.e. the `<getProduct>` block.

Because different toolkits and implementations of the XML-Signature specification can use attributes other than the `Id` attribute, the Enterprise Gateway allows the user to specify other attributes that should be supported in this manner. By default, the Enterprise Gateway supports the `Id`, `ID`, and `AssertionID` attributes for the purposes of identifying the signed content within an XML Signature.

However it is possible to add more attributes by clicking the **Add** button and adding the attribute in the interface provided. The priorities of attributes can be altered by clicking the **Up** and **Down** buttons. For example, if most of the XML Signatures processed by the Enterprise Gateway use the `ID` attribute, this attribute should be given the highest priority.

WSSE Namespace

The **WSSE Namespace** tab is used to specify the WSSE (and corresponding WSSU) namespaces that are supported by the Enterprise Gateway.

The Enterprise Gateway attempts to identify WS Security blocks belonging to the WSSE namespaces listed in this table. It first attempts to locate Security blocks belonging to the first listed namespace, followed by the second, then the third, and so on until all namespaces have been utilized. If no Security blocks can be found for any of the listed namespaces, the message will be rejected on the grounds that the Enterprise Gateway does not support the namespace specified in the message. To add a new namespace, click the add button.

It is important to note that every WSSE namespace has a corresponding WSSU namespace. For example, the following WSSE and WSSU namespaces are inextricably bound:

<i>WSSE Namespace</i>	<code>http://schemas.xmlsoap.org/ws/2003/06/secext</code>
<i>WSSU Namespace</i>	<code>http://schemas.xmlsoap.org/ws/2003/06/utility</code>

First, enter the WSSE namespace in the **Name** field. Then enter the corresponding WSSU namespace in the **WSSU Namespace** field.

SOAP Namespace

The **SOAP Namespace** tab can be used to configure the SOAP namespaces that are supported by the Enterprise Gateway. In a similar manner to the way in which the Enterprise Gateway handles WSSE namespaces, the Enterprise Gateway will attempt to identify SOAP messages belonging to the listed namespaces in the order given in the table.

The default behavior is to attempt to identify SOAP 1.1 messages first, and for this reason, the SOAP 1.1 namespace is listed first in the table. The Enterprise Gateway will only attempt to identify the message as a SOAP 1.2 message if it can't be categorized as a SOAP 1.1 message first.

Find Filter Dialog

Overview

The **Find Filter Dialog** can be used to find a named filter of a particular type. It is especially useful in cases where many policies have been configured with the Policy Studio and the administrator wants to find a specific filter. For example, you could find a schema filter called, "Check against SOAP Schema".

Configuration

The **Find Filter Dialog** is available from the **Edit -> Find Filter** main menu option. Enter the name of the filter to find in the **Name** field, and then select its type from the **Filter Type** dropdown. Click the **Search** button to find the named filter.

If you want to search for all filters of a particular type, select the type from the **Filter Type** dropdown and leave the **Name** field blank. Click the **Search** button to find all filters of this type. The filters will be displayed in the **Search Results** table.

Similarly, to display a listing of all configured filters, simply leave both fields blank and click the **Search** button.

It is possible to right-click on a filter displayed in the **Search Results** table and then select any of the following options from the context menu:

Edit:

Use this option to edit the selected filter.

Delete:

Select this option to delete the selected filter from the policy in which it was configured.

Log Level:

Select the log level(s) that this filter will log at using the relevant menu options. All filters can log at "Failure", "Fatal", and "Success" levels. To edit the log message for this filter, select the **Edit** option from the context menu to display the configuration data for the filter, and then click the **Next** button to edit the log messages.

View Item:

When this menu option is selected, the filter will be displayed in the policy tree-view on the left hand side of the Policy Studio. Furthermore, the policy in which this filter runs will be displayed in the policy palette.

Service Monitor Installation Guide

Overview

This topic describes the steps involved in installing and setting up Service Monitor.

Prerequisites

The prerequisites before installing are as follows:

Enterprise Gateway Installation

Because Service Monitor reports on transactions processed by the Enterprise Gateway in real-time, the Enterprise Gateway must already be installed.

JDBC Database

The Enterprise Gateway stores and maintains the monitoring and transaction data read by Service Monitor in a JDBC-compliant database. Service Monitor provides setup scripts for the following databases:

- MySQL
- Microsoft SQL Server
- Oracle
- IBM DB2

Extracting Service Monitor

Extract the compressed file containing the Service Monitor files to a suitable location.

Windows

Unzip the Service Monitor zip file to a suitable location. For demonstration purposes, this document assumes that the Service Monitor zip file is extracted to `C:\Oracle`.

Linux/Solaris

To extract the gzip file, perform the following steps:

1. gunzip the Service Monitor gzip file, for example:

```
$gunzip oracleservicemonitor.tar.gz
```

For demonstration purposes, this document assumes that Service Monitor is extracted to `/usr/local/oracle`.

2. Extract the Service Monitor tar archive, for example:

```
$tar -xvf oracleservicemonitor.tar
```

In this document, the location that you extract the Service Monitor application into is referred to as the `INSTALL_DIR`.

Setting up the Database

Service Monitor reads message metrics from a database and displays this information in a visual format to administrators. This is the same database in which the Enterprise Gateway stores its audit trail and message metrics data. You first need to create this database using the database product of your choice (MySQL, Microsoft SQL Server, Oracle, or IBM DB2). For details on how to do this, see the product documentation for your chosen database. In this document, the example database is named `reports`, but you can use whatever name you wish.

When the database is created, you can then set up the relevant database tables. Service Monitor provides SQL scripts to set up the database tables for each of the supported databases. For the purposes of this installation guide, this document assumes that you are using a MySQL database server, and that you have already installed a MySQL database server on the local machine. SQL scripts for setting up the database tables are provided in the `INSTALL_DIR/system/conf/sql` folder in the `/mysql`, `/mssql`, `/oracle`, and `/db2` directories.

When you have created the `reports` database, run the SQL commands in the `db_schema.sql` file. Choose the appropriate version of this file from one of the `mssql`, `mysql`, `oracle`, or `db2` folders, depending on the database you intend to store the metrics in. The following example shows the output after creating and setting up the tables in a MySQL database:

```
mysql> create database reports;
Query OK, 1 row affected (0.00 sec)

mysql> use reports;
Database changed
mysql> show tables;
+-----+
| Tables_in_reports |
+-----+
| audit_log_points  |
| audit_log_sign    |
| audit_message_payload |
| metrics_alerts    |
| metrics_data      |
| persistent_stat_targets |
| process_ids       |
+-----+

7 rows in set (0.02 sec)
```

Note:

If you wish to use a MySQL database to store message metrics, you can simply copy and paste the contents of the `db_schema.sql` file into the MySQL command prompt (`mysql>`).

Configuring the Service Monitor Database

By default, Service Monitor is configured to read message metrics from a MySQL database stored on the local machine. Typically, you may wish to use an alternative database, change the password on the default database connection, or add users. To configure the Service Monitor database, perform the following steps:

1. Start the Service Monitor server by running the `servicemonitor` command from the following directory of your Service Monitor installation.

Windows	<code>win32\bin</code>
----------------	------------------------

UNIX/Linux	posix/bin
------------	-----------

2. In your Policy Studio installation directory, enter the `polycystudio` command.
3. On the Policy Studio **Home** tab, click the following URL to connect to Service Monitor:

`http://HOST:8040/configuration/deployments/DeploymentService`

where `HOST` points to the IP address or hostname of the machine on which Service Monitor is running.

4. Specify your user name and password details. For more information, see [Connection Details](#).
5. Click **Finish**.
6. On the **Dashboard**, double-click the **Service Monitor** process to load its configuration.
7. Click the **External Connections** button on the left of the Policy Studio.
8. Expand the **Default Database** tree node.
9. Right-click the **Default Database Connection** tree node, and select **Edit**.
10. The **Database Connection** dialog enables you to configure the database connection details. By default, the connection is configured to read metrics data from the `reports` database. Edit the details for the **Default Database Connection** on this dialog, making sure to point to the `reports` database. For example, you should enter a non-default database user name and password:

Name	Default Database Connection
URL	<code>jdbc:mysql://127.0.0.1:3306/reports</code>
User Name	<code>root</code>

The following table lists examples of connection URLs for the supported databases, where `reports` is the name of the database and `DB_HOST` is the IP address or host name of the machine on which the database is running:

<i>Database</i>	<i>Example Connection URL</i>
Oracle	<code>jdbc:oracle:thin:@DB_HOST:1521:reports</code>
Microsoft SQL Server	<code>jd-bc:sqlserver://DB_HOST:1433;DatabaseName=reports;integratedSecurity=false;</code>
MySQL	<code>jdbc:mysql://DB_HOST:3306/reports</code>
IBM DB2	<code>jdbc:db2://DB_HOST:50000/reports</code>

However, in most cases, you may wish to create a new database connection to connect to a database other than the default local database. You can do this by right-clicking **Database Connections** in the Policy Studio tree view, and selecting **Add a Database Connection**. For more details on using the **Configure Database Connection** dialog, see [Database Connection](#).

- Click the **Deploy** button in the toolbar. Alternatively, press F6.

You can verify that your database connection is configured correctly in the following trace file:
`INSTALL_DIR/tracing/Oracle Service Monitor.trc.`

Configuring the Service Monitor TCP Port

Service Monitor is configured to listen on port 8040 by default. If you have another process already using this port on the machine on which you have installed Service Monitor, you must configure Service Monitor to listen on different port. You can do this in the Policy Studio as follows:

- Click the **Services** button at the bottom left of the screen.
- Expand the **Processes** -> **Service Monitor** -> **Service Monitor Services** node.
- Right-click the HTTP Interface node identified by the default IP address and port setting (*:8040), and select **Edit**.
- In the **Configure HTTP Interface** dialog, edit the value in the **Port** field to change the port on which Service Monitor serves reports. For more details, see [Configuring HTTP Services](#).

Configuring the Enterprise Gateway

This section explains how to configure the Enterprise Gateway to store message metrics in the previously created `reports` database.

Connect to the Enterprise Gateway

To connect to the Enterprise Gateway in Policy Studio, perform the following steps:

- Start the Enterprise Gateway by running the `enterprisegateway` command from the following directory of your product installation.

Windows	Win32\bin
UNIX/Linux	posix/bin

- On the Policy Studio **Home** tab, click a server session to make a connection to a server.
- Specify your connection details (host, port, user name, and password). The default connection URL is:

```
http://HOST:8090/configuration/deployments/DeploymentService
```

where `HOST` points to the IP address or hostname of the machine on which the Enterprise Gateway is running. For more information, see [Connection Details](#).

- Click **OK**.
- On the Enterprise Gateway Dashboard, double-click the **Enterprise Gateway** process to load its configuration.

Configure the Database Connection

To configure the database connection, perform the following steps:

- Click the **External Connections** button on the left of the Policy Studio.
- Expand the **Database Connections** tree node.

3. Right-click the **Default Database Connection** tree node, and select **Edit**.
4. Configure the database connection to point to the same `reports` database created earlier. For more details, see [Configuring the Service Monitor Database](#).
5. Click **OK**.

Configure Database Logging

To configure the database logging settings, perform the following steps:

1. In the Policy Studio tree view, right-click the **Oracle Enterprise Gateway** process, and select **Logging ->Custom**.
2. Click **Yes**.
3. On the **Configure Logging** dialog, select the **Database** tab, and select the **Enable logging to database** checkbox.
4. Select the **Default Database Connection** from the drop-down if appropriate. Alternatively, select a database connection that you have configured. You must ensure that your database connection points to the same `reports` database created earlier. For more details, see [Configuring the Service Monitor Database](#).
5. Click **OK**.

Configure Monitoring Settings

To configure the Enterprise Gateway to monitor traffic and store message metrics in the `reports` database, perform the following steps:

1. Right-click the Oracle Enterprise Gateway process, and select **Monitoring -> Metrics**.
2. On the **Real-time monitoring settings** dialog, select the **Enable real time monitoring** checkbox.
3. In the **Reports settings** panel, select the **Store real time monitoring data for charts/reports** checkbox, and specify the following settings:

Use the following database	Specify the database connection that points to the <code>reports</code> database from the drop-down list.
Time window to store	The time period over which metrics are accumulated and enables you to view messages processed over this interval (for example, the number of messages processed over a 5 second interval).

4. Click **OK**.
5. If you wish to enable monitoring of traffic per Enterprise Gateway, right-click the **Oracle Enterprise Gateway** process, and select **Monitoring -> Traffic**.
6. In the **Traffic Monitor Settings** dialog, ensure **Enable Traffic Monitor** is selected.
7. Click **OK**.

Important Note:

Enabling traffic monitoring has a negative impact on performance. If you wish to maximize performance, do not enable these settings. For more details, see [Configuring Traffic Monitoring](#).

Deploy to the Enterprise Gateway

When you have made these configuration changes, you must deploy them to the Enterprise Gateway. To do this, select **Settings -> Deploy** from the Policy Studio main menu. Alternatively, click the **Deploy** button in the toolbar, or press F6. The Enterprise Gateway now sends both audit trail and message metrics data to the `reports` database. This database is then queried by Service Monitor to produce reports showing system health, service usage, clients, message size and volume, and so on.

It is important to note that the **Monitoring Settings** configured here are process-wide. You can also configure an HTTP Services Group to send metrics data to the database selected at the process level. To do this, right-click the HTTP Ser-

vices Group (for example, **Default Services**), and select **Edit**. To monitor traffic in this Group, ensure that the **Include in real-time monitoring** checkbox is selected. Similarly, you can disable monitoring for this Group by unselecting this checkbox.

Verify your Database Connection Settings

You can verify that your database connection is configured correctly in the `INSTALL_DIR/tracing/Oracle Enterprise Gateway.trc` file. For more details on tracing, see [Troubleshooting](#).

Launching Service Monitor

To launch Service Monitor, perform the following steps:

1. Start Service Monitor using the `servicemonitor` script, in the `/bin` directory of your Service Monitor installation.
2. Using the default port, you can connect to the Service Monitor interface in a browser at the following URL:

```
http://HOST:8040/
```

where `HOST` points to the IP address or hostname of the machine on which Service Monitor is installed.

3. Log in using the default `admin` user with password `changeme`. You can edit this user in Policy Studio using the **Users** interface from the Policy Studio tree view.

Note:

Service Monitor produces reports based on message metrics that are stored by the Enterprise Gateway when processing XML messages. To produce a graph that shows the number of connections made by the Enterprise Gateway to a particular Web Service, you must first set up a policy that routes messages to that Web Service. When this policy has been configured, you must send some messages through this policy so that they are routed on to the target Web Service.

In addition, if you change from using an existing database with remote hosts/clients configured to another database with a different set of remote hosts/clients configured, you must restart both the Enterprise Gateway and Service Monitor.

Further Information

For more details on using Policy Studio management features, see [Getting Started with Managing Deployments](#). For more details on viewing real-time monitoring and transaction data using Service Monitor, see the [Service Monitor User Guide](#).

Service Monitor User Guide

Overview

This topic describes how to use the various screens available on the Service Monitor web-based interface. It assumes that you have already performed the steps described in the [Service Monitor Installation Guide](#).

Topology Overview

To monitor the traffic between Enterprise Gateway instances and various Web Services, Clients, and Remote Hosts, running throughout your network, you must add the Enterprise Gateway instance as a *node* on the **Topology overview** page. This page shows all nodes (Enterprise Gateway instances) that are sending monitored traffic to protected Web Services, Clients, and Remote Hosts.

Each of these Enterprise Gateway nodes must already be configured to store message metrics in the same database that Service Monitor is configured to read from. This enables the **Real-time Monitoring**, **Reports**, and **Audit Trail** screens to obtain the Enterprise Gateway metrics and logs that they display from this database.

On the **Topology overview** page, if the database on the Enterprise Gateway server is configured correctly, a database icon with a green up arrow is displayed on the Enterprise Gateway node. However, if the Enterprise Gateway is not configured correctly, a red down arrow is displayed. For details on configuring database connections, see the [Service Monitor Installation Guide](#).

Each node on the **Topology overview** page shows the total number of messages processed by the Enterprise Gateway that it represents. The message count is updated every 5 seconds. If the Service Monitor can not connect to a node, a connection error is displayed on the node.

Adding a New Node

To add a new node, click the **Add node** button at the bottom of the page. You must configure the following fields when adding a new node:

Host URL:

Enter the URL of the Enterprise Gateway that is protecting traffic that you want to monitor. The URL entered here should include the host name and port number of the Enterprise Gateway, for example:

```
http://HostName:8090/metrics
```

where `HostName` is the host name of the machine on which the Enterprise Gateway is running, and port `8090` is the port used by the Enterprise Gateway's management interface.

Custom Name:

Enter an optional friendly name for this node. This may be useful to help distinguish between different instances of the Enterprise Gateway in cases where you have deployed several Enterprise Gateways throughout your network.

Username:

If the management interface on the node you are connecting to requires authentication (for example, HTTP Basic authentication), you must enter the username to authenticate with in this field.

Password:

Enter the password for this user in the field provided.

On the **Topology overview** page, you can right-click a node, and view reports for all nodes, real-time monitoring for that node, or the audit trail for that node, using the **Reports**, **Real-time Monitoring**, and **Audit Trail** options respectively. There are also buttons for these options available on the bottom of the **Topology overview** page. The following sections describe each of these options:

- [Reports](#)
- [Real-time Monitoring](#)
- [Audit Trail](#)

Reports

Service Monitor uses message metrics stored in a centralized database by the Enterprise Gateway instances running throughout your network. The Enterprise Gateway stores metrics for the Web Services that it exposes (virtualized services), and for the Web Services, and Client and Remote Host connections that it protects. Service Monitor can then generate usage reports and charts based on this data. The **Reports** screen is responsible for rendering these charts. The following configuration options apply to each of the three tabs available on this screen.

Per-process statistics:

Reports can be generated for monitored objects *per-process* or *per-cluster*. Currently, all Enterprise Gateway processes in Service Monitor's database belong to a single cluster (domain). An *identity* makes metrics aggregated from corresponding objects in Enterprise Gateway configurations available in a cluster. For example, Web Services of the same name configured on Enterprise Gateway processes in a cluster make up a single Web Service identity.

So if Service A is configured on host 1, and Service A on host 2 (both in the same cluster), these make up the Service A identity. In reports with per-cluster statistics, the values for service A are aggregated from statistics for service A on both hosts. When generating reports with per-process statistics, Service A on host 1, and Service A on host 2 are taken as separate entities, and separate metrics for each are available.

In Service Monitor, you can specify whether per-process or per-cluster statistics are generated using the **Per-process statistics** checkbox, which is disabled by default. Depending on this selection, the drop-down list of items to generate in the report shows individual Web Service items on individual hosts or Web Service identities (one Web Service identity for all Web Services under the same name configured on hosts in the cluster). When this setting is unselected, the **Remote hosts** list displays cluster-wide Remote host identities. When selected, the **Remote hosts** list displays individual per-process Remote hosts.

Aggregated Metrics:

Aggregated Metrics display metrics for a group of monitored objects (for example, all Web Services) across one or more nodes over a specific time period. This report displays the time line using a graph. This enables you to view the total performance metrics for all Web Services, Remote Hosts, or Clients.

Totals:

Use this option to show all values for a specific monitored object (for example, a Web Service) over a specific time period. The report uses a bar chart to display the total value of a single metric for a monitored object over a specified time period. This enables you to compare the performance metrics of specific Web Services, Remote Hosts, or Clients.

Time Window:

You must select the time window size that you want to generate reports for using the button at the top of the screen. The selected option *must* match the time window specified in the **Real time monitoring settings** dialog when configuring the real-time monitoring settings for the Enterprise Gateway. For example, if you selected **5 minutes** for the **Time window to store** on the **Real time monitoring settings** dialog, you must select the **5 minute time window metrics** when generating reports on the following tabs:

- Remote Hosts
- Web Services
- Clients

In addition, all monitored Enterprise Gateway nodes must use the same **Time window to store** setting.

Remote Hosts:

The Remote Hosts drop-down list contains all Remote Hosts that have been configured on Enterprise Gateways for which monitoring has been enabled. You can select one or more Remote Hosts to produce a report for.

Aggregates reports for Remote Hosts show the following:

- Response times in terms of the number of requests processed in the specified response time ranges.
- Remote Host uptime, which details the results of a Watchdog configured for a Remote Host. You can add Watchdogs to Remote Hosts by right-clicking the Remote Host node on **Services** tab in the Policy Studio, and selecting **Watchdog -> Add**. For more details, see the [Watchdog Configuration](#) topic.
- Total number of bytes transferred into and out of this Remote Host.

Totals reports for Remote Hosts display the following metrics:

- Response times.
- Bytes transferred.

Web Services:

The **Web Services** drop-down list includes all Web Services that have been invoked on the Enterprise Gateway configured for monitoring. In particular, the **Set Web Service Context** filter must have been invoked. You can select one or more Web Services from the list to generate reports on these Web Services.

Aggregates and Totals reports show the total number of messages that have been processed by the Web Service, in terms of those that have been blocked, those that have passed, and those that have failed.

Clients:

On this tab, the **Web Services** drop-down list displays all Web Services that have been invoked and had their **Set Web Service Context** filter called. Select one or more Web Services from this list. The **Clients** drop-down list is then populated with all clients that have been authenticated for the selected Web Services.

Aggregates and Totals reports show message metrics for clients that have accessed the selected Web Service. This enables administrators to view which users are calling which Web Services.

Real-Time Monitoring

You must enable real-time monitoring on the Enterprise Gateway to monitor real-time usage statistics on the **Real-time monitoring** screens.

Enabling Real-Time Monitoring

To enable real-time monitoring, perform the following steps:

1. In the Policy Studio tree view, right-click the HTTP Services Group (for example, **Default Services**), and select **Edit**.
2. In the **HTTP Services** dialog, ensure the **Include in real-time monitoring** checkbox is selected.
3. Click **OK**.

To enable monitoring for the entire Enterprise Gateway process, perform the following steps:

1. On the **Services** tab in the Policy Studio, right-click the **Oracle Enterprise Gateway** process, and select **Monitoring -> Traffic**.
2. In the **Traffic Monitor Settings** dialog, ensure the **Enable Traffic Monitor** setting is selected.
3. Click **OK**.

4. Right-click the **Oracle Enterprise Gateway** Process node, and select **Monitoring -> Metrics**.
5. In the **Real time monitoring settings** dialog, ensure the **Enable real time monitoring** setting is selected.
6. Click **OK**.

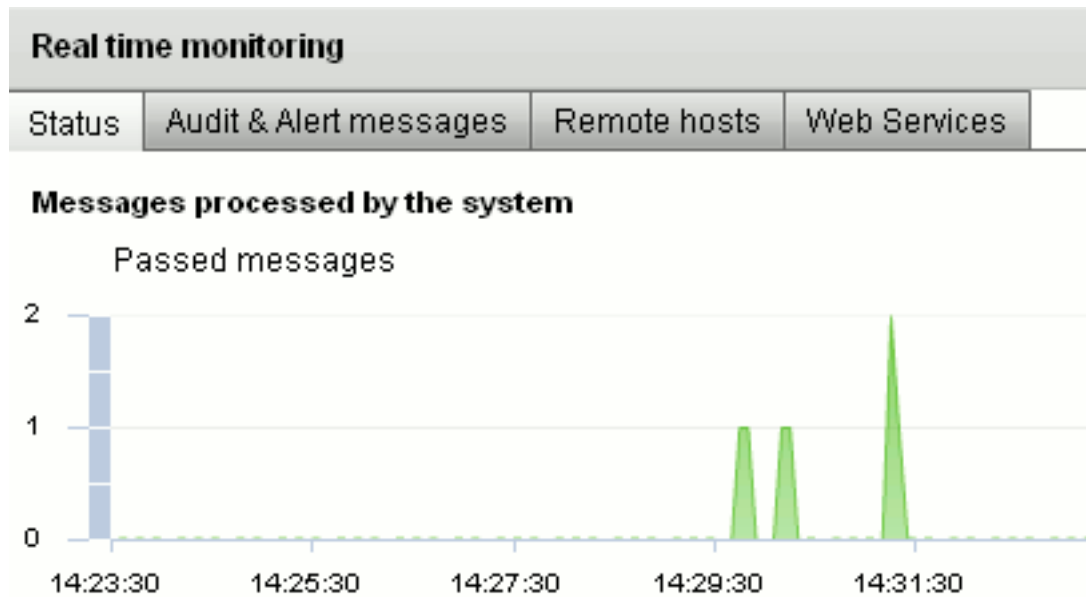
Important Note:

Enabling traffic monitoring has a negative impact on performance. If you wish to maximize performance, do not enable these settings. For more details, see [Configuring Traffic Monitoring](#).

When you have configured the Enterprise Gateway for monitoring, you can monitor traffic secured (or blocked) by the Enterprise Gateway using the **Real-time monitoring** screen. The tabs on this screen are described as follows:

Status:

This tab shows the total number of messages processed by the system. There are separate charts to illustrate graphically the number of messages that have passed, that have been blocked, and that have failed.

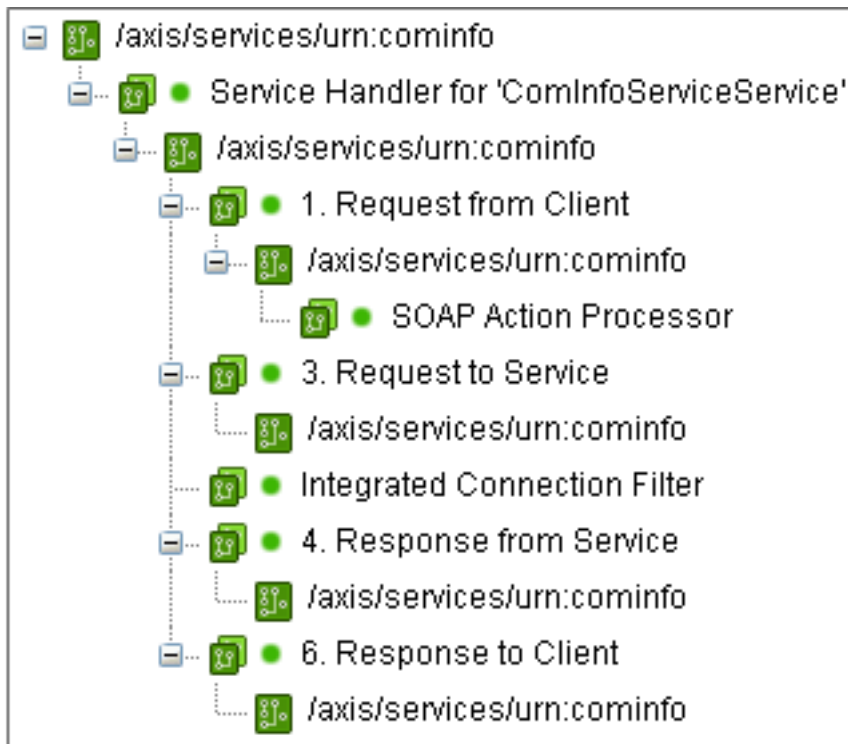


The **Latest HTTP Messages** table on this tab shows traffic monitoring statistics for all HTTP messages processed by the currently selected Enterprise Gateway. You can click the **Refresh** button to update this list. For example, the available details per message include the following:

Time	The time that the message was processed at.
Status	The HTTP status code of the message (for example, 200 for OK, or 403 for Forbidden).
Method	The HTTP method (for example, GET or POST).
Local Address	The local address on which the Enterprise Gateway is installed.
URI	The relative path that the message is sent to (for example, axis/services/urn:cominfo).
Remote Address	The remote address on which the Web Service resides.
Remote Name	The remote hostname on which the Web Service resides.

The **Real-time monitoring** console also displays the message path and the contents of each request message and response message. For example, to view the message path and the content of the response message sent by a Web Service through the Enterprise Gateway, perform the following steps:

1. In the **Latest HTTP Messages** list, select the message whose contents you wish to view. This displays the message path for the selected message under the list on the left.
2. Select the outbound message from the Enterprise Gateway, which is second in the list on the right of the message path.
3. Click the **View** button for the **Response** message to view its contents.
4. Alternatively, click the **Download** button for the **Response** message to download its contents and save them to a file.



Audit and Alert messages:

This tab displays tables for **Audit messages**, **Alert messages**, and **SLA breach messages**. Each table displays various details about the message, including time, message, message ID, and the filter or source of the message.

Each table displays the last 50 messages for the category that it represents. For example, the **Audit messages** table displays the last 50 log messages, while the SLA (Service Level Agreement) breach table shows the last 50 SLA breaches that have been triggered throughout the system. For more information on triggering alerts, setting SLAs, and configuring custom logging, see [Alerting and Logging](#).

Remote hosts:

This tab lists all remote hosts that have been configured on a particular node. The **Configured remote hosts** table shows the number of messages that have been sent to this remote host, together with the total number of bytes sent to and received from this host.

You can view more detailed information about a particular remote host by clicking it in the table. The **Message details**

table shows general performance metrics for the remote host, including total number of requests, number of requests/second, bytes sent and received, and transfer rate in bytes/second.

The **Remote hosts response codes** table shows the total number of each class of HTTP response status that was returned by the remote host. Typically, a remote host returns a 200 status code when it has processed the message correctly, but it may return a 4xx or 5xx message to indicate that an error has occurred. You can use this table to gauge whether the remote host is processing messages successfully or rejecting them.

The **Remote hosts response times** table categorizes the response times of the remote host in terms of certain pre-configured time intervals. Administrators can easily examine how their remote hosts are responding to requests in this table.

Remote host: flyer:7070

Display ☒ Totals or data for last ... ☐ 5 seconds ☐ 5 minutes ☐ 1 hour

Message details:	Remote host response codes:	Remote host response times:
Number of requests: 5	1xx: 0	< 10 ms: 1
Requests / second: 0	2xx: 5	< 100 ms: 4
Bytes transfered (in & out): 5894	3xx: 0	< 500 ms: 0
Bytes transfered / sec. (in & out): 0	4xx: 0	< 1000 ms: 0
Bytes sent: 3099	5xx: 0	< 3s: 0
Bytes received: 2795		< 6s: 0
		< 10s: 0
		< 30s: 0
		< 60s: 0
		> 60s: 0

Finally, you can use the **Remote host up/down** table in conjunction with Watchdogs that have been configured on a remote host to determine the availability of the remote host. When a Watchdog is configured for a remote host, it polls the remote hosts at regular intervals by sending requests to it, and examining the responses. This table displays the total number of Watchdog requests that have been deemed a success, and those that have been deemed a failure, depending on the criteria configured for the Watchdog.

To add a Watchdog to a remote host, right-click the remote host node in the Policy Studio tree, and select **Watchdog -> Add**. For more details, see [Configuring an HTTP Watchdog](#).

Web Services:

The **Web Services** tab shows usage statistics for services that are exposed by a particular node. These Web Services are created and exposed by importing service definitions from WSDL files. To do this, in the Policy Studio tree view, right-click the **Web Services** node, and select **Register Web Service**. Follow the steps in the wizard to expose one or more of the operations defined in the WSDL. Alternatively, you can also register Web Services using the web-based Service Manager tool. For details, see [Managing Web Services](#).

Traffic for Web Services that have been exposed by the Enterprise Gateway in this manner can be monitored on this tab. The **Web service usage** table lists all Web Services that are currently exposed by the Enterprise Gateway. You can click

a Web Service in the table to display more detailed metrics for this service.

Note:

A Web Service must have been called before it is displayed in the **Web service usage** table.

Audit Trail

The Audit Trail enables you to filter log messages generated by a particular node. You can view the filters by clicking the **Advanced Search** button. You can then filter log messages by time period, severity level, filter type, and filter name. You can even filter log messages based on the text of the log message itself. When you have configured the relevant filters, you can click the **Run** button to search all log messages that match the specified criteria.

The log messages that match the search criteria specified in the filter are displayed in the table. The details in the table include the log level (severity), time, log message text, and message ID.

Scheduled Reports

Overview

You can schedule reports to run on a regular basis, and have the results emailed to the user in PDF format. These reports include summary values at the top (for example, the number of requests, SLA breaches, alerts triggered, and unique clients in a specific week) followed by a table of Web Services, and their aggregated usage data (for example, the number of requests on each Web Service).

By default, the report data is for the current week of the report, which is compared to the week before. You can configure the current week of the report to be the actual current week or any previous week (provided there is corresponding data in the database).

To configure scheduled reports, right-click the **Service Monitor** node on the **Services** tab, and select **Database Archive**.

Database Configuration

You can configure the following database settings in the dialog:

Name:

Enter an appropriate name for the database archive.

Database:

Select an existing database connection from the drop-down list. Defaults to the `Default Database Connection`. For details on creating database connections, see the [Database Connection](#) topic.

Scheduled Report Configuration

You can configure the following scheduled report settings:

Send weekly scheduled report:

Select whether to enable scheduled reports. When selected, by default, this runs a scheduled weekly report on Monday morning at 0:01. For details on configuring a different time schedule, see [Configuring a Time Schedule](#).

Metrics time window:

Select the time window for metrics data in the database used when generating reports. You can select `5 seconds`, `5 minutes`, or `1 hour` from the drop-down list. Defaults to `5 seconds`. You must configure all Enterprise Gateways that store metrics in the database to use the same time window.

Email Recipient (To):

Enter the recipient of the automatically generated report email. Use a semicolon-separated list of email addresses to send alerts to multiple recipients.

Email Sender (From):

The generated report emails appear *from* the sender email address specified here. It is important to note that some mail servers do not allow relaying mail when the sender in the **From** field is not recognized by the server.

SMTP Server Settings:

Specify the following fields:

Outgoing Mail Server (SMTP)	Specify the SMTP server that the Enterprise Gateway uses to relay the report email.
Port	Specify the SMTP server port to connect to. Defaults to port 25.
Connection Security	Select the connection security used to send the report

	email (SSL, TLS, or NONE). Defaults to NONE.
--	--

Log on Using:

If you are required to authenticate to the SMTP server, specify the following fields:

User Name:	Enter the user name for authentication.
Password:	Enter the password for the user name specified.

Advanced Options

You can configure the following advanced options:

Email Debugging:

You can select this setting to find out more information about errors encountered by the Enterprise Gateway when attempting to send email alerts. All trace files are written to the `/trace` directory of your Enterprise Gateway installation.

Don't delete intermediary report files:

This option is for debugging scheduled reports. PDF reports are generated through several stages. When this setting is enabled, the temporary files used in the process are not deleted when the report is sent. These files are stored in the system temporary directory. There are two files for each report: an XML file with metrics data for the report, and a generated PDF report.

Configuring a Time Schedule

You can configure additional report schedule settings using the **Entity Explorer** tool. For example, you can specify exactly when the reports are scheduled to run, and how many weeks back from the current time the current week is in the reports. To configure these settings, perform the following steps:

1. To start the **Entity Explorer** tool, enter the `esexplorer.bat` command from your Service Monitor installation directory.
2. From the main menu, select **Store -> Connect**.
3. Click **Browse**, and open the following file in your Service Monitor installation:
`InstallDir/conf/fed/PrimaryStore.xml`
4. On the **Entities** tab on the left, expand the **Entity Stores** tree, and select the **Reporter** node.
5. In the panel on right, double-click the `reportScheduledTime` field in the **Value** column.
6. Enter a cron expression that specifies when you wish the report to run. For example, the default expression is `0 1 0 ? * MON`, which specifies to run the report each Monday morning at 0:01. For more details on how to use this syntax, see the section on Cron Expressions in the [Policy Execution Scheduling](#) topic.
7. You can also use the `reportEndWeekOffset` setting to specify how many weeks back from the current time that the current week is in the report. The default value is 1, which generates a report for the full previous week from the current time that the report is run, and compares with the values from the week before that. For example, if the report is scheduled to run on Monday morning, it includes the previous week, not only those few hours of the current week. If the value is 0, the report only includes data from the current week until now, and compares with the values from the previous week. You can use values higher than 1 to generate a report for any previous week for which there is corresponding data in the database.
8. Click **Update** when finished.

Configuring Processes

Overview

This topic shows how to configure a Process, which represents a running instance of the Enterprise Gateway. You can configure the options described in the following sections at the Process level.

Add HTTP Services

You can add a container for HTTP-related services, including HTTP and HTTPS Interfaces, Directory Scanners, Static Content Providers, Servlet Applications, and Packet Sniffers.

HTTP Services act as a container for all HTTP-related interfaces to the Enterprise Gateway's core messaging pipeline. You can configure HTTP and HTTPS interfaces to accept plain HTTP and SSL messages respectively. A Relative Path interface is available to map requests received on a particular URI or path to a specific policy. The Static Content Provider interface can retrieve static files from a specified directory, while the Servlet Application enables you to deploy servlets under the service. Finally, the Packet Sniffer interface can read packets directly of the network interface, assemble them into HTTP messages, and dispatch them to a particular policy. The [Configuring HTTP Services](#) topic explains how to configure the available HTTP Interfaces.

Add SMTP Services

Simple Mail Transfer Protocol (SMTP) support enables the Enterprise Gateway to receive email and to act as a mail relay. The Enterprise Gateway can accept email messages using the SMTP protocol, and forward them to a mail server. You can also configure optional policy circuits for specific SMTP commands (for example, `HELO/EHLO` and `AUTH`). The [Configuring SMTP Services](#) topic explains how to configure SMTP services, interfaces, and handler circuits.

Add Policy Execution Scheduler

Policy Execution Scheduling enables you to schedule the execution of any policy on a specified date and time in a recurring manner. The Enterprise Gateway provides a pre-configured library of schedules to select from. You can also add your own schedules to the library. The [Policy Execution Scheduling](#) topic explains how to add a policy execution schedule, and how to add schedules.

Messaging System

You can configure the Enterprise Gateway to read JMS messages from a JMS queue or topic, run them through a policy, and then route onwards to a Web Service or JMS queue or topic.

The Enterprise Gateway can consume a JMS queue or topic as a means of passing XML messages to its core message processing pipeline. When the message has entered the pipeline, it can be validated against all authentication, authorization, and content-based message filters. Having passed all configured message filters, it can be routed to a destination Web Service over HTTP, or it can be dropped back on to a JMS queue or topic using the **Messaging System** Connection filter. For more details, see the [Messaging System](#) topic.

Directory Scanner

The **Directory Scanner** reads XML files from a specified directory and dispatches them to a selected policy. This enables you to search a local directory for XML files, which can then be fed into a security policy for validation. Typically, XML files are FTP-ed or saved to the file system by another application. The Enterprise Gateway can then pick these files up, run the full array of authentication, authorization, and content-based filters on the messages, and then route them over HTTP or JMS to a back-end system. For more details, see the [Directory Scanner](#) topic.

POP Client

The **POP Client** enables you to poll a POP mail server to read email messages from it, and pass them into a policy for processing. For more details, see the [POP Client](#) topic.

Add Remote Host

Remote Host settings configure the way in which the Enterprise Gateway routes to another host machine. For example, if a destination server may not fully support HTTP 1.1, you can configure **Remote Host** settings for the server to optimize the way in which the Enterprise Gateway sends messages to it. Similarly, if the server requires an exceptionally long timeout, you can configure this in the **Remote Host** settings. For more details, see the [Remote Hosts](#) topic.

TIBCO

You can configure a TIBCO Rendezvous® Listener or a TIBCO Enterprise Messaging System™ Consumer. For more details, see the following topics:

- [TIBCO Rendezvous Listener](#)
- [TIBCO Enterprise Messaging System Consumer](#)

Process Settings

You can configure per-process global configuration settings by right-clicking the Process, and selecting the **Settings** option. For more details on configuring Process settings, see the [General Settings](#) topic.

Process Logging

You can configure a Process to log messages to a database, file system, GUI Console, log files, or UNIX syslog. A Log Viewer for examining log entries is also available. For more details, see the [Logging Configuration](#) topic.

Monitoring

The Enterprise Gateway can store useful statistics about the messages that it processes in a database. The Service Monitor monitoring tool can then poll this database, and produce charts and graphs detailing how the Enterprise Gateway is performing. For more details, see the [Service Monitor Index Page](#).

The Enterprise Gateway also provides the Traffic Monitor tool for operational diagnostics. This is a web-based message log of the HTTP and HTTPS traffic that it processes. For more details, see the topic on [Monitoring Traffic](#).

Cryptographic Acceleration

The Enterprise Gateway can leverage the OpenSSL Engine API to offload complex cryptographic operations (for example, RSA and DSA) to a hardware-based cryptographic accelerator, and to act as an extra layer of security when storing private keys on a Hardware Security Module (HSM).

The Enterprise Gateway uses OpenSSL to perform cryptographic operations, such as encryption and decryption, signature generation and validation, and SSL tunneling. OpenSSL exposes an *Engine API*, which enables you to plug in alternative implementations of some or all of the cryptographic operations implemented by OpenSSL. OpenSSL can, when configured appropriately, call the engine's implementation of these operations instead of its own. For more information on configuring the Enterprise Gateway to use an OpenSSL engine, see the [Cryptographic Acceleration](#) topic.

Tivoli

You can configure how a Process connects to an instance of an IBM Tivoli Access Manager server. Each Process can connect to a single Tivoli server. For more details, see the Global Configuration section in the [Tivoli Integration](#) topic.

Kerberos

You can configure Process-wide Kerberos settings such as the Kerberos configuration file to the Enterprise Gateway, which contains information about the location of the Kerberos Key Distribution Center (KDC), encryption algorithms and keys, and domain realms. You can also configure options for APIs used by the Kerberos system, such as the Generic Security Services (GSS) and Simple and Protected GSSAPI Negotiation (SPNEGO) APIs. For more details, see the [Kerberos Configuration](#) topic.

Oracle Security Service Module Settings

You can configure the Enterprise Gateway to act as an Oracle Security Service Module (SSM) to enable integration with Oracle Entitlements Server. The Enterprise Gateway acts as a Java SSM, which delegates to Oracle Entitlements Server. For example, you can authenticate and authorize a user for a particular resource against an Oracle Entitlements Server repository. For more details, see the [Oracle Security Service Module Settings](#) topic.

Audit Trail

The Enterprise Gateway generates an audit trail for each of the key actions that occurs in the Policy Studio on configurations, processes, and users (for example, when a user logs in, or updates configuration). All items are written to a file-based audit trail stored on the same machine on which the server process is running. For more details, see the [Audit Trail](#) topic.

Configuring HTTP Services

Overview

The Enterprise Gateway uses *HTTP Services* to service traffic from various HTTP-based sources. The following list summarizes the list of available HTTP Services:

HTTP Interfaces

HTTP Interfaces are used in the Enterprise Gateway to define the ports and IP addresses on which the Enterprise Gateway Process listens for incoming requests. You can also add *HTTPS Interfaces* and select the SSL certificate to use to authenticate to clients, and also which certificates are considered trusted for establishing SSL connections with clients.

Relative Path

While HTTP and HTTPS Interfaces open sockets and listen for traffic, you can configure *Relative Paths* so that when a request is received on that path, the Enterprise Gateway Process can map it to a specific policy or chain of policies.

Static Content Provider

You can use a *Static Content Provider* to serve static HTTP content from a particular directory. In this case, the Enterprise Gateway Process is effectively acting as a Web Server.

Servlet Applications

The Enterprise Gateway Process can act as a servlet container, which enables you to drop servlets into the HTTP Services configuration. This should only be used by developers with very specific requirements and under strict advice from the Oracle support team.

Packet Sniffer

Finally, you can add a *Packet Sniffer* to intercept network packets from the client, assemble these packets into complete HTTP messages, and log these messages to an audit trail. Because the Packet Sniffer operates at the network layer (unlike an HTTP-based traffic monitor at the application layer), the packets are intercepted transparently to the client. Because of this, the Packet Sniffer is a *passive service*, which is typically used for SOA management and monitoring instead of general policy enforcement.

This topic describes how to configure the available HTTP Services.

HTTP Services Groups

An *HTTP Services Group* is a container around one or more HTTP Services. Usually, an HTTP Services Group is configured for a particular type of HTTP Service. For example, it would be typical to have an *HTTP Interfaces Group* that contains the configured HTTP Interfaces, and another *Static Providers Group* to manage Static Content Providers. While organizing HTTP Services by type eases the task of managing Services, the Enterprise Gateway is flexible enough to enable administrators to organize Services into Groups according to whatever scheme best suits their requirements.

This section describes a scenario where HTTP Services Groups can prove useful. It first considers an HTTP Service Group that handles HTTP traffic, and then shows how you can use a second SSL Service Group to process SSL traffic on a separate channel.

HTTP Interfaces and Relative Paths

HTTP Services Groups should consist of at least one HTTP Interface together with at least one Relative Path. The HTTP Interface determines which TCP port the Process listens on, while you can use the Relative Path to map a request received on a particular path (request URI) to specific policies. You can add several HTTP Interfaces to the Groups, in which case requests received on any one of the opened ports are processed in the same manner. For example, `http://[HOST]:8080/test` and `http://[HOST]:8081/test` requests can both be processed by the same policy (mapped to the `/test` Relative Path).

Similarly, you can add multiple Relative Paths to this HTTP Services Group, where each Path is bound to a specific policy or chain of policies. For example, if a request is made to `http://[HOST]:8080/a`, it is processed by Policy A;

whereas if a request is made to `http://[HOST]:8080/b`, it is handled by Policy B. As a side-effect of this configuration, requests made to the other Interface are also processed by the same policy, meaning that a request made to `http://[HOST]:8081/b` is also handled by Policy B.

Effectively, this means that the Relative Paths configured under the HTTP Services Group are bound to all HTTP Interfaces configured for that Group. In other words, if you have two Interfaces listening on ports 8080 and 8081, requests to `http://[HOST]:8080/a` and `http://[HOST]:8081/a` are handled identically by the Process.

Example HTTP Service Group

But what happens if you want to make a distinction between receiving requests on the two different ports? For example, say you want requests to `http://[HOST]:443/a` to be processed by an **SSL Validation** policy, while requests for `http://[HOST]:8080/a` to be handled by a standard **Schema Validation** policy. How can you achieve this?

The addition of a new HTTP Services Group can resolve this issue. The new `SSL HTTP Services Group` opens a single HTTPS Interface that listens on port 443, and is configured with a Relative Path of `/a` to handle requests on this path. The configuration is summarized in the following table:

Services Group	HTTP Port	Relative Path
HTTP Services Group	8080	/a
SSL HTTP Services Group	443	/a

With this configuration, when you receive a request on `http://[HOST]:8080/a`, it is handled by the **Schema Validation Policy**. But when you get a request to the SSL port on `http://[HOST]:443/a` it is processed by the **SSL Validation Policy**. Using HTTP Services Groups in this way, you can configure the Process to dispatch requests received on the same path (for example, `/a`) to different policies depending on the port on which the request was accepted.

Default HTTP Service Groups

By default, the Enterprise Gateway ships with two pre-configured HTTP Services Groups (`Default Services` and `Management Services`). By default, `Management Services` are not visible in the Policy Studio, but can be made visible using the **Preferences** menu in the Policy Studio. The `Default Services` Group contains some general purpose default policies for use with an out-of-the-box installation of the Enterprise Gateway. In addition to these two pre-configured Service Groups, you can add new HTTP Services Groups for situations like the one described above where you want to dispatch requests to different policies, based on the port on which the requests are received.

For more details on the default `Management Services` Group, see the [Note on Management Services](#).

Adding an HTTP Service Group

To add a Service Group, right-click the Process, and select the **Add HTTP Services** menu option. Enter a name for the group in the **HTTP Services** dialog. If you want to monitor traffic processed by this Service Group using Service Monitor, check the **Include in real time monitoring** checkbox. For details on configuring the Enterprise Gateway to store real-time monitoring data, which can be used to produce graphical reports in a web-based interface, see the [Service Monitor Install Guide](#).

When a Service Group has been created, the following types of HTTP Services can be associated with it.

HTTP and HTTPS Interfaces

An HTTP Interface defines the address and port that the Enterprise Gateway Process listens on. There are two types of Interface: HTTP and HTTPS. The HTTP interface handles standard non-authenticated HTTP requests, while the HTTPS interface can accept mutually authenticated SSL connections.

To create an HTTP Interface, right-click the name of the HTTP Service Group (for example, `Default Services`) in the tree view of the Policy Studio, and select **Add Interface** from the menu options. To create an HTTP Interface, select the **HTTP** menu option. To create an HTTPS Interface, select the **HTTPS** menu option.

Configuring Network Settings

The following fields on the **Network** tab are common to both the **HTTP Interface** and **HTTPS Interface** dialogs, and must be configured for both types of interface:

Port:

The port number that the Enterprise Gateway Process listens on for incoming HTTP requests.

Address:

The IP address or host of the network interface on which the Enterprise Gateway Process listens. For example, you can configure the Process to listen on port 80 on the external IP address of a machine, while having a Web server running on the same port but on the internal IP address of the same machine. By entering *, the Process listens on all interfaces available on the machine hosting the Enterprise Gateway.

Protocol Version:

Select the Internet Protocol version that this Interface uses. You can select IPv4, IPv6, or both of these protocol versions.

Backlog:

When the Process is busy handling concurrent requests, the operating system can accept additional incoming connections. In such cases, a backlog of connections can build up while the operating system waits for the Process to finish handling current requests.

The specified **Backlog** value is the maximum number of connections that the Process allows the operating system to accept and queue up until the Process is ready to read them. There is a trade off: the larger the backlog, the larger the memory usage; the smaller the backlog, the greater the potential for dropped connections.

Idle Timeout:

The Enterprise Gateway supports the use of HTTP 1.1 persistent connections. The **Idle Timeout** is the time that the Process waits after sending a response over a persistent connection before it closes the connection.

Typically, a client tells the Process that it wants to use a persistent connection. The Process acknowledges this instruction and decides to keep the connection open for a certain amount of time after sending the response to the client. If the client does not reuse the connection by sending up another request within the **Idle Timeout** period, the Process closes the connection.

Active Timeout:

When the Process receives a large HTTP request, it reads the request off the network as it becomes available. If the time between reading successive blocks of data exceeds the **Active Timeout**, the Process closes the connection.

This guards against a client closing the connection while in the middle of sending data. Imagine the client's network connection is pulled out of the machine while in the middle of sending data to the Process. When the Process has read all the available data off the network, it waits the **Active Timeout** period of time before closing the connection.

Maximum Memory per Request:

The maximum amount of memory that the Process allocates to each request, not including the HTTP body.

Trace level:

The level of trace output. The possible values in order of least verbose to most verbose output are:

- FATAL
- ERROR
- INFO
- DEBUG
- DATA

The default trace level is read from the system settings.

Transparent Proxy (allow bind to foreign address):

Enables the use of the Enterprise Gateway as a *transparent proxy* on Linux systems with the `TPROXY` kernel option set. When selected, the value in the **Address** field can specify any IP address, and incoming traffic for the configured address/port combinations is handled by the gateway. For more details and an example, see [Configuring a Transparent Proxy](#).

Configuring Traffic Monitor Settings

The fields on the **Traffic Monitor** tab are common to both the **HTTP Interface** and **HTTPS Interface** dialogs. To override the system-level settings at the HTTP or HTTPS interface level, select **Override settings for this port**, and configure the following options:

Record inbound transactions	Select whether to record inbound message transactions received by the Enterprise Gateway. This is enabled by default.
Record outbound transactions	Select whether to record outbound message transactions sent from the Enterprise Gateway to remote hosts. This is enabled by default.
Record circuit path	Select whether to record the circuit path for the message transaction, which shows the filters that the message passes through. This is enabled by default.
Record trace	Select whether to record the trace output for the message transaction. This is enabled by default.

For more details, see [Configuring Traffic Monitor Settings](#).

Configuring Conditions for an HTTP Interface

You can configure the Enterprise Gateway to bring down an active HTTP Interface if certain *conditions* fail to hold. For example, the HTTP Interface can be brought down if a Remote Host is not available or if a physical network interface on the machine on which the Enterprise Gateway is running loses connectivity to the network.

For more information on configuring *conditions* for HTTP Interfaces, see the [Configuring Conditions for HTTP Interfaces](#) help page.

HTTPS Interfaces Only

You must complete the same fields for an HTTPS Interface on the **Network** tab as for an HTTP Interface, with the addition of the following configuration options:

X.509 Certificate:

Click the **X.509 Certificate** button to select the certificate that the Enterprise Gateway uses to authenticate itself to clients during the SSL handshake. The list of certificates currently stored in the **Certificate Store** is displayed. Select a single certificate from this list.

SSL Server Name Identifier (SNI):

You can specify the hostnames that are requested by clients in the **SSL Server Name Identifier (SNI)** table. SNI is an optional TLS feature whereby the client indicates to the server the hostname used to resolve the server's address. This enables a server to present different certificates for clients to ensure the correct site is contacted.

For example, the server IP address is 192.168.0.1. The DNS is consulted by clients to resolve a hostname to an address, and the server address is contacted using TCP/IP. If both `www.acme.com` and `www.anvils.com` resolve to 192.168.0.1, without SNI, the server does not know which hostname the client uses to resolve the address, because it is not party to the client's DNS name resolution. The server may be able to certify itself as either service, but when the connection is established, it does not know which hostname the client connects to.

With SNI, the client provides the name of the host (for example, `www.anvils.com`) in the initial SSL exchange, before

the server presents its certificate in its distinguished name (for example, CN=www.anvils.com). This enables the server to certify itself correctly as providing a service for the client's requested hostname.

To specify an SNI, perform the following steps:

1. Click the **Add** button to configure a server hostname in the **SSL Server Name Identifier (SNI)** dialog.
2. Specify the server hostname in the **Client requests server name** field.
3. Click the **Server assumes identity** button to import a Certificate Authority certificate into the Trusted Certificate Store.
4. **Click OK.**

Ciphers:

You can specify the ciphers that the server supports in the **Ciphers** field. The server selects the highest strength cipher (that is also supported by the client) from this list as part of the SSL handshake. For more information on the syntax of this setting, please see the [OpenSSL documentation](http://www.openssl.org/docs/apps/ciphers.html) [http://www.openssl.org/docs/apps/ciphers.html].

SSL Session Cache Size:

Specifies the number of idle SSL sessions that can be kept in memory. You can use this setting to improve performance because it caches the slowest part of establishing the SSL connection. A new connection does not need to go through full authentication if it finds its target in the cache. Defaults to 32. If there are more than 32 simultaneous SSL sessions, this does not prevent another SSL connection from being established, but means that no more SSL sessions are cached. A cache size of 0 means the cache size is unlimited.

Mutual Authentication:

You can configure clients to authenticate to the Enterprise Gateway on the **Mutual Authentication** tab. The following options are available:

- **Ignore Client Certificates**
The Enterprise Gateway ignores client certificates if they are presented during the SSL handshake.
- **Accept Client Certificates**
Client certificates are accepted when presented to the Enterprise Gateway, but clients that do not present certificates are not rejected.
- **Require Client Certificates**
The Enterprise Gateway only accepts connections from clients that present a certificate during the SSL handshake.

Client certificates are typically issued by a Certificate Authority (CA). In most cases, the CA includes a copy of its certificate in the client certificate so that consumers of the certificate can decide whether or not to trust the client based on the issuer of the certificate.

It is also possible that a *chain* of CAs were involved in issuing the client certificate. For example, a top-level organization-wide CA (for example, Company CA) may have issued department-wide CAs (for example, Sales CA, QA CA, and so on), and each department CA would then be responsible for issuing all department members with a client certificate. In such cases, the client certificate may contain a chain of one or more CA certificates.

You can use the **Maximum depth of client certificate chain** field to configure how many CA certificates in a chain of one or more are trusted when validating the client certificate. By default, only one issuing CA certificate is used, and this certificate must be checked in the list of trusted root certificates. If more than one certificate is used, only the top-level CA must be considered trusted, while the intermediate CA certificates are not.

Root Certificate Authorities trusted for mutual authentication

Select the root CA certificates that the Enterprise Gateway considers trusted when authenticating clients during the SSL handshake. Only certificates signed by the CAs selected here are accepted.

Relative Paths

A Relative Path binds policies to a specific relative path location (for example `/healthcheck`). When the Enterprise Gateway receives a request on this relative path, it invokes the specified policies. For details on how to configure policies, see [Policy Configuration](#).

To configure a Relative Path for a given HTTP Service Group, right-click the Service Group in the tree view, and select **Add Relative Path**. Complete the following settings on the **Configure Relative Path** dialog:

Enable this path resolver:

You can specify whether to enable listening on the specified path. This is enabled by default.

Policy:

On the **Policy** tab, specify the relative path and the policies that are called. The Enterprise Gateway invokes the selected policies when it receives a request on the specified path. You can specify a single policy or a chain of policies. Policies are called in the order displayed on this tab. Complete the following fields:

When a request arrives that matches the path:	Enter a relative path (for example, <code>/test</code>) for the selected HTTP Service Group. Requests received on this relative path are processed by the policies selected on this tab.
Global Request Policy	If a global request policy is configured, when you select this setting, the global request policy is called first in the policy chain. For more details, see Configuring Global Policies .
Path Specific Policy	To configure a path-specific policy, select the checkbox, and click the browse button to select a policy from the dialog.
Global Response Policy	If a global response policy is configured, when you select this setting, the global response policy is called last in the policy chain. For more details, see Configuring Global Policies .

When you select multiple policies to form a policy chain, the behavior is the same as for a policy shortcut chain filter. Policies are only evaluated when selected, and when the policy can be reached. If any reachable policy fails, the chain fails, and no more policies are evaluated.

Audit:

The **Audit** tab enables you to configure the logging level for all filters executed on the relative path, and to configure when message payloads are logged.

You can configure the following **Logging Level** settings on all filters executed on the specified relative path:

<i>Logging Level</i>	<i>Description</i>
Fatal	Logs Fatal log points that occur on all filters executed.
Failure	Logs Failure log points that occur on all filters executed. This is the default logging level.
Success	Logs Success log points that occur on all filters executed.

For more details on logging levels, and on how to configure logging for a specific filter, see the [Log Level and Message](#) topic.

You can configure the following **Payload Logging** settings on the specified relative path:

<i>Payload Logging</i>	<i>Description</i>
On request	Log the message payload when a request arrives.
On response	Log the message payload before the response is sent back.
On remote send	Log the message payload before the request is sent using any Connection or Connect to URL filters deployed in any policies executed.
On remote receive	Log the message payload when the response is received using any Connection or Connect to URL filters deployed in any policies executed.

For details on how to log message payloads at any point in a specific policy, see the [Log Message Payload](#) topic.

Web Service Resolvers

A Web Service Resolver is used to identify messages destined for a Web Service, and to map them to the **Service Handler (Web Service Filter)** for that Web Service. When you import a WSDL file, a new Web Service Resolver node is created for each imported Web Service in the **Services** tree view. You can edit the Web Service Resolver settings by right-clicking this tree node, and selecting **Edit**.

The following settings are available in the **Web Service Resolver** dialog:

Enable this web service resolver:

Specify whether to enable this Web Service resolver. This is enabled by default.

Name:

You can edit the name of the Web Service Resolver.

Web Service:

Click the browse button to select a Web Service to resolve to. Defaults to the Web Service imported into the Web Services Repository when this resolver was created.

Policy:

On the **Policy** tab, select the path and the policies to use for the Web Service. You can specify a single policy or a chain of policies. Policies are called in the order displayed on this tab. The global request policy, the policy automatically generated when the WSDL file is imported, and the global response policy are all selected in a chain by default. Complete the following fields:

matches the paths in the WSDL:	Select this option if you want the Web Service Resolver to use the paths specified in the WSDL file. This is the default.
matches this path:	Select this option if you want to specify a different path from the WSDL file, and enter the path.
Global Request Policy	If a global request policy is configured, when you select this setting, the global request policy is called first in the policy chain. For more details, see Configuring Global Policies .
Path Specific Policy	To configure a path-specific policy, select the checkbox, and click the browse button to select a policy from the dialog.
Global Response Policy	If a global response policy is configured, when you select this setting, the global response policy is called last in the

	policy chain. For more details, see Configuring Global Policies .
--	---

Policies are only evaluated when selected, and when the policy can be reached. If any selected policy fails, the chain fails, and no more policies are evaluated.

Audit:

The **Audit** tab enables you to configure the logging level for all filters executed on the Web Service, and to configure when message payloads are logged. The default logging level for all filters on the Web Service is *Failure*. These logging settings are the same as those already described for the **Audit** tab used for the Relative Path. For more details, see [Relative Paths](#).

Editing Service Handler Options

You also edit options for the **Service Handler** for the Web Service. Right-click the Web Service Resolver node, and select **Quick-Edit Policy** to display a dialog that enables you to configure the following options:

Validation	If you wish to use a dedicated validation policy for all messages sent to the Web Service, select this checkbox, and click the browse button to configure a policy in the dialog. For example, this enables you to delegate to a custom validation policy used by multiple Web Services.
Routing	If you wish to use a dedicated routing policy to send all messages on to the Web Service, select this checkbox, and click the browse button to configure a policy in the dialog. For example, this enables you to delegate to a custom routing policy used by multiple Web Services.
WSDL Access Options	Select whether to make the WSDL for this Web Service available to clients. The Allow the Enterprise Gateway to publish WSDL to clients checkbox is selected by default. The published WSDL represents a virtualized view of the Web Service. Clients can retrieve the WSDL from the Enterprise Gateway, generate SOAP requests, and send them to the Enterprise Gateway, which routes them on to the Web Service.

These options enable you to configure the underlying auto-generated Service Handler (**Web Service Filter**) without navigating to it in the **Policies** tree. These are the most commonly modified **Web Service Filter** options after importing a WSDL file. Changes made in this dialog are visible in the underlying **Web Service Filter**. For more details, see the [Web Service Filter](#) topic.

Static Content Provider

A *Static Content Provider* can be used in conjunction with an HTTP Interface to serve static content from a specified directory. A relative path is associated with each Static Content Provider so that requests received on this path are dispatched directly to the Provider and are not mapped to a Policy in the usual manner.

For example, you can configure a Static Content Provider to serve content from the `c:\docs` folder (on a Windows system) when it receives requests on the relative path `/docs`.

To add a Static Content Provider, right-click the Service Group under the Process, and select the **Static Content Pro-**

vider menu option. Complete the following fields on the **Static Content Provider** dialog:

Relative Path:

Enter the path that you want to receive requests for static content on.

Content Directory:

Enter or browse to the location of the directory that you want to serve static content from.

Index File:

Enter the name of the file that you want to use as the index file for content retrieved from the directory specified in the field above. This file is retrieved by default if no resource is explicitly specified in the request URI. For example, if the client requests `http://[HOST]:8080/docs` (with only a relative path specified as opposed to a specific resource), the file specified here will be retrieved. This file must exist in the directory specified in the previous field.

Allow Directory Listings:

If this option is selected, the Static Content Provider returns full directory listings for requests specifying a relative path only. For example, if this option is selected, and if a request is received for `http://[HOST]:8080/docs/samples`, the list of directories under this directory is returned, assuming that this directory exists on the file system. This option can be turned off to prevent attacks where a hacker can send up different request URIs in the hope that the server returns some information about the directory structure of the server.

Servlet Applications

Developers may wish to write their own Java servlets and deploy them under the Enterprise Gateway to serve HTTP traffic. Conversely, they may wish to remove some of the default servlets from the out-of-the-box configuration (for example, to remove the ability to view logging remotely). This pairing down of unwanted functionality may be required to further lock down the machine on which the Enterprise Gateway is running.

Note:

Adding and removing Servlet Applications should be performed only by developers with very specific requirements and under strict guidance from the Oracle Support team. These instructions simply outline how to configure the fields on the dialogs used to set up Servlet Applications. For more detailed instructions, please contact the Oracle [Support Team](#).

There are a few default Servlet Applications available under the **Management Services** group. By default, this services group is not visible, but can be displayed using the **Preferences** dialog on the Policy Studio. The following table describes the default servlets:

<i>Servlet Application</i>	<i>Use</i>
/configuration/	Updates configuration information for the Enterprise Gateway.
/runtime/	Deploys to the server after an update has been made to a policy.
/monitoring/	Displays real-time monitoring of server transactions.

Please note that deleting any of these Servlet Applications may prevent the Enterprise Gateway from functioning correctly. Default Servlet Applications should only be deleted under strict supervision of the Oracle Support team.

To add a new Servlet Application, right-click the Services Group that you wish to add the servlet to, and select **Add Servlet Application**. Configure the following fields on the **Servlet Application** dialog:

Relative Path:

Enter the servlet *context* in this field. You can add multiple servlets under this context, where each servlet is mapped to a unique URI.

Session Timeout:

Enter the timeout in seconds after which an inactive session is closed. Click **OK**.

The new Servlet Application now appears in the Policy Studio tree view. To add a new servlet, right-click the new Servlet Application, and select **Add Servlet**. Configure the following fields on the **Servlet** dialog:

URI:

The path entered here maps incoming requests on a particular request URI to the Java servlet class entered in the field below. This path must be unique across all Servlets that are added under this Servlet Application (servlet context).

Class:

Enter the fully qualified class name of the servlet class. This class can be added to the server runtime by adding the JAR, class file, or package hierarchy to the `[VINSTDIR]/ext/lib` folder, where `VINSTDIR` points to the root of your Enterprise Gateway installation.

Read Timeout:

Specify the time in seconds that the servlet should wait before closing an idle connection.

Servlet Properties:

You can configure properties for each servlet by clicking the **Add** button, and entering a name and value in the fields provided on the **Properties** dialog.

Packet Sniffers

Packet Sniffers are a type of passive service. Rather than opening up a TCP port and *actively* listening for requests, the Packet Sniffer *passively* reads raw data packets off the network interface. The Sniffer assembles these packets into complete messages that can then be passed into an associated policy.

Because the Packet Sniffer operates passively (does not listen on a TCP port) and, therefore, completely transparently to the client, it is most useful for monitoring and managing Web Services. For example, the Sniffer can be deployed on a machine running a Web Server acting as a container for Web Services. Assuming that the Web Server is listening on TCP port 80 for traffic, the Packet Sniffer can be configured to read all packets destined for port 80 (or any other port, if necessary). The packets can then be marshaled into complete HTTP/SOAP messages by the Sniffer and passed into a policy that logs the message to a database, for example.

Important Note:

On Linux and Solaris platforms, the Enterprise Gateway must be started by the root user to gain access to the raw packets.

Because Packet Sniffers are mainly for use as passive monitoring agents, they are usually created in their own HTTP Service Group. For example, you can create a new Service Group for this purpose by right-clicking the Process, selecting the **Add HTTP Services** menu option, and entering **Packet Sniffer Group** on the **HTTP Services** dialog.

You can then add a Relative Path Service to this Group by right-clicking the **Packet Sniffer Group**, and selecting the **Add Relative Path** menu option. Enter a path in the field provided, and select the policy that you want to dispatch messages to when the Packet Sniffer detects a request for this path (after it assembles the packets). For example, if the Relative Path is configured as */a*, and the Packet Sniffer assembles packets into a request for this path, the request is dispatched to the policy selected in the Relative Path service.

Finally, you can add the Packet Sniffer itself by right-clicking the **Packet Sniffer Group** node, selecting **Packet Sniffer**, and then the **Add** menu option. Complete the following fields on the **Packet Sniffer** dialog:

Device to Monitor:

Enter the name or identifier of the network interface that the Packet Sniffer will monitor. The default entry is *any*, but it is important to note that this is only valid on Linux. On UNIX-based systems, network interfaces are usually identified using names like *eth0*, *eth1*, and so on. On Windows, these names are more complicated (for example, `\Device\NPF_{00B756E0-518A-4144 ... }`).

Filter:

You can configure the Packet Sniffer to only intercept certain types of packets. For example, it can ignore all UDP packets, only intercept packets destined for port 80 on the network interface, ignore packets from a certain IP address, listen for all packets on the network, and so on.

The Packet Sniffer uses the `libpcap` library filter language to achieve this. This language has a complicated but powerful syntax that enables you to *filter* what packets are intercepted and what packets are ignored. As a general rule, the syntax consists of one or more expressions combined with conjunctions, such as `and`, `or`, and `not`. The following table lists a few examples of common filters and explains what they filter:

<i>Filter Expression</i>	<i>What does it filter?</i>
port 80	Capture only traffic for the HTTP Port (80).
host 192.168.0.1	Capture traffic to and from IP address 192.168.0.1.
tcp	Capture only TCP traffic.
host 192.168.0.1 and port 80	Capture traffic to and from port 80 on IP address 192.168.0.1.
tcp portrange 8080-8090	Capture all TCP traffic destined for ports from 8080 through to 8090.
tcp port 8080 and not src host 192.168.0.1	Capture all TCP traffic destined for port 8080 but not from IP address 192.168.0.1.

The default `tcp` filter simply captures all TCP packets arriving on the network interface. For more information on how to configure filter expressions like these, see the [tcpdump man pages](http://www.tcpdump.org/tcpdump_man.html) [http://www.tcpdump.org/tcpdump_man.html].

Promiscuous Mode:

When listening in promiscuous mode, the Packet Sniffer captures all packets on the same Ethernet network, regardless of whether or not the packets are addressed to the network interface that the Sniffer is monitoring.

Management Services

The **Management Services** group exposes a number of services that are used for remote configuration and monitoring. By default, this group is not visible in the Policy Studio tree view. However, you can view it by selecting the **Settings -> Preferences** main menu option, and selecting the **Show Management Services** checkbox. Click the **Apply** button to view the services. This setting also displays the **Management Services** under the **Policies** node in the Policy Studio tree view.

By default, the **Management Services** group consists of the following:

HTTP Interface: Port 8090

By default, the Enterprise Gateway exposes all its management services on port 8090 so that they can be configured remotely. At startup, the Policy Studio can connect to this port to read and write Enterprise Gateway configuration data. For more details, see [Changing the Management Services Port](#) below.

Relative Path: /

The `/` Relative Path is mapped to a default management policy called **Protect Management Interface**, which is available under the **Management Services** Policy Container. This policy performs HTTP Basic Authentication and passes control to the **Call Internal Service** filter. This is a special filter that dispatches a message to a Servlet Application or Static Content Provider based on the path on which the request was received.

For example, with the default configuration, assume that a request is received on `http://localhost:8090/configuration`. The following steps summarize the request processing cycle:

1. When a Relative Path of `/` is configured, it matches all incoming requests, and requests are dispatched to whatever policy the Relative Path is mapped to. In this case, the Relative Path is mapped to the **Protect Management Interface** policy, and so the request is passed to this policy.
2. The **Protect Management Interface** policy performs HTTP basic authentication on the originator of the request. Authentication is necessary because all configuration operations are considered privileged operations and should only be carried out by those with the authority to do so. If the originator can be successfully authenticated, the **Call Internal Service** filter is invoked.
3. The **Call Internal Service** filter is a special filter that passes messages to a Servlet Application or Static Content Provider. In this case, because the message is received on the management interface (port 8090), the filter attempts to match the Relative Path on which the request was received against all the Servlets and Content Providers configured in the same **Services Group** as this interface.
4. The configured Servlets and Content Providers for the **Management Services** group include `/configuration/`, `/runtime/`, and `/monitoring/`. Because the request is received on the `/configuration/` path, this matches the `/configuration/` Servlet Application, which is invoked.

Servlet Application: `/configuration/`

The Policy Studio running on a different host to the Enterprise Gateway can connect to this URL to remotely configure the Enterprise Gateway. For example if the Enterprise Gateway is running on a host called `SERVER`, the Policy Studio can connect to `http://SERVER:8090/configuration/` on startup so that it can remotely configure policies running at the Enterprise Gateway on the `SERVER` host.

Servlet Application: `/runtime/`

This application is used to receive remote requests to deploy to the server (for example, by the Policy Studio). When policy updates are made by the Policy Studio, it is necessary to deploy them to the server so that the policy updates can take effect immediately. Policy deployment requests are sent to this application.

Static Content: `/monitoring/`

You can remotely view real-time traffic analysis of the Enterprise Gateway. To do this, open a browser, and point it at `http://SERVER:8090/monitoring/`, where the Enterprise Gateway is running on a host named `SERVER`. The `/monitoring/` Static Content Provider then serves a HTML page containing an embedded flash component that displays real-time monitoring.

Important Note:

Changing the Interfaces, Relative Path, Servlet Applications, or Static Content Provider exposed under the **Management Services** group may prevent the Enterprise Gateway from functioning correctly. Because of this, the **Management Services** group is hidden by default, and should only be modified under strict supervision from the Oracle Support team.

Changing the Management Services Port

The default Management Services port is 8090. To specify a different port, perform the following steps:

1. In the Policy Studio main menu, select **Settings -> Preferences**, and ensure that the **Show Management Services** checkbox is selected.
2. On the Policy Studio **Services** tab, right-click the **Management Service** HTTP Interface, and select **Edit**.
3. Specify an updated value in the **Port** field (for example, `8091`), and click **OK**.
4. Click the **Deploy** button in the Policy Studio toolbar, or press F6 to deploy the update.
5. Restart Policy Studio. You must restart Policy Studio when Management Services are updated.
6. If you were connected to the Policy Center server, you can use the same URL to reconnect Policy Studio. If you were connected to the Enterprise Gateway server, you must use the updated port number in the URL to reconnect Policy Studio (for example, `http://localhost:8091/configuration/deployments/DeploymentService`).
7. In the Policy Studio, on the **Oracle Enterprise Gateway Dashboard**, the Enterprise Gateway Process is displayed

as uncontactable because it still uses the original port number.

8. Click the **Manage Processes** button, select the Enterprise Gateway Process in the **Processes** list, and click the **Connection** tab.
9. Specify the updated port number in the **URL** field, click **Update**, and click **OK**.
10. Wait for the server to be notified about the process URL change (for example, 60 seconds), and then refresh the **Oracle Enterprise Gateway Dashboard**.
11. The Process is now displayed as deployed, and you can edit its configuration.

Important Note:

As stated earlier, you should only modify **Management Services** under strict advice and supervision from the Oracle Support team.

Configuring SMTP Services

Overview

The Enterprise Gateway provides support for Simple Mail Transfer Protocol (SMTP), which enables the Enterprise Gateway to receive email and to act as a mail relay. The Enterprise Gateway can accept incoming email messages using the SMTP protocol, and then forward them on to a configured mail server. You can also use the Policy Studio to configure optional policy circuits for specific SMTP commands (for example, `HELO/EHLO`, `AUTH`, `MAIL FROM`, and so on).

When an SMTP command is configured in the Policy Studio, each time the SMTP command is accepted by the Enterprise Gateway, the appropriate policy circuit is executed. When the circuit completes successfully, the SMTP conversation resumes. This topic shows how to configure SMTP services, interfaces, and handler circuits using the Policy Studio.

Adding an SMTP Service

To add an SMTP service to enable the Enterprise Gateway to accept SMTP connections, perform the following steps in the Policy Studio:

1. On the **Services** tab on the left, select a Process tree node (for example, the default **Oracle Enterprise Gateway**).
2. Right-click, and select **Add SMTP Services**.
3. In the **SMTP Services** dialog, specify a unique name for the SMTP service in the **Name** field.
4. In the **Outgoing Server Settings** section, complete the following settings:

Host	Host name or IP address of the remote mail server. This is the server to which the Enterprise Gateway forwards incoming SMTP commands (for example, <code>smtp.gmail.com</code>). You can also specify a mail server running locally on the same machine as the Enterprise Gateway using an address of <code>localhost</code> or <code>127.0.0.1</code> .
Port	Port on which to connect to the remote mail server. Defaults to port 25.

5. In the **Security** section, complete the following settings:

Connection Security	Select the type of security used for the connection to the remote mail server. Defaults to <code>None</code> . Other possible values are <code>SSL</code> and <code>STARTTLS</code> .
Trusted Certificates	Use this tab to select the trusted CA certificates used in the security handshake for the connection to the remote mail server. This field is mandatory if <code>SSL</code> or <code>STARTTLS</code> connection security is selected.
Client SSL Authentication	Use this tab to specify the trusted client certificates used in the security handshake for the connection to the remote mail server. This field is optional if <code>SSL</code> or <code>STARTTLS</code> connection security is selected.
Advanced	Use this tab to specify a list of ciphers to use during the security handshake for the connection to the remote mail server. Defaults to <code>DEFAULT</code> . For more details, see the OpenSSL ciphers manpage. This field is optional if <code>SSL</code> or <code>STARTTLS</code> connection security is selected.

6. In the **Authentication** section, complete the following settings:

Username	Specify the username used to authenticate the Enterprise Gateway with a remote SMTP server using the <code>AUTH</code> SMTP command. For more details, see the section on SMTP Authentication .
Password	Specify the password used to authenticate the Enterprise Gateway with a remote SMTP server using the <code>AUTH</code> SMTP command. For more details, see the section on SMTP Authentication .

7. Select the **Include in real time monitoring** checkbox to monitor the SMTP services using the Enterprise Gateway real-time monitoring and Service Monitor tools.
8. **Click OK**. This creates a tree node for the SMTP service under the selected process in the **Services** tree.

Adding an SMTP Interface

When you have configured the outbound SMTP protocol, you must then set up an inbound interface to accept client connections. You can choose from the following interface types:

TCP	Non-secure connection. All traffic is sent in-the-clear.
SSL	SSL handshake is performed at connection time, so the entire SMTP conversation is secure.
STARTTLS	Initial connection is in the clear. The Enterprise Gateway advertises STARTTLS during the initial SMTP <code>HELO/EHLO</code> handshake. If the client supports this, it can send a STARTTLS command to the Enterprise Gateway, which in turn promotes connection security, and upgrades the connection to SSL/TLS.

Because the SSL and STARTTLS interface types have the potential to be secure (STARTTLS starts off non-secure, but can be upgraded during the SMTP conversation), a common configuration screen is used for both protocols in the Policy Studio.

To configure an inbound interface, perform the following steps in the Policy Studio:

1. On the **Services** tab on the left, select the SMTP tree node under the Process.
2. Right-click, and select **Add Interface** -> interface type (**TCP**, **SSL**, or **STARTTLS**).
3. Complete the settings on the relevant dialog. For full details on these settings, see the [Configuring HTTP Services](#) topic.
4. **Click OK**.

Configuring Circuit Handlers for SMTP Commands

You can use the Policy Studio to configure optional policy circuit handlers for each of the following SMTP commands:

- `HELO/EHLO`
- `AUTH`

- MAIL FROM
- RCPT TO
- DATA

The next sections explain how to configure circuit handlers for each command.

Adding an HELO/EHLO Circuit Handler

The **HELO/EHLO** policy circuit handler is invoked when a **HELO/EHLO** SMTP command is received from a client. This handler enables you to modify the **HELO/EHLO** greeting and the client domain. You can configure the greeting message sent back to the client from the Enterprise Gateway during the **HELO/EHLO** handshake as required. You can also configure a circuit to replace the value of `smtp.helo.greeting`. The domain specified by the connected client in the **HELO/EHLO** command can be modified before forwarding on to the remote mail server. You can also configure a circuit to replace the value of `smtp.helo.domain`.

To configure a policy circuit handler for the **HELO/EHLO** command, perform the following steps:

1. On the **Services** tab, select the SMTP tree node under the Process.
2. Right-click, and select **Add Circuit Handler -> HELO/EHLO**.
3. In the **Configure HELO Host** dialog, specify the **Greeting** to be sent back to the client as part of the **HELO/EHLO** handshake. Defaults to `Hello ${smtp.helo.domain}`.
4. In the **Policy** tree, select the policy that you wish to handle the **HELO/EHLO** command.
5. Click **OK**.

Message attributes

The following message attributes are generated during processing:

Message Attribute	Description
<code>smtp.helo.domain</code>	The domain specified by the connecting client in its HELO/EHLO SMTP command.
<code>smtp.helo.greeting</code>	The HELO greeting to be sent back to the client after HELO/EHLO processing is performed. The default value is: <code>Hello \${smtp.helo.domain}</code> .
<code>monitoring.enabled</code>	true if monitoring is enabled for the protocol, otherwise false.
<code>message.monitoring.enabled</code>	true if message-monitoring is enabled for the protocol, otherwise false.

Adding an AUTH Circuit Handler

The **AUTH** policy circuit handler is invoked when an **AUTH** SMTP command is received from a client. You can use the **AUTH** handler to run a circuit to perform user authentication checks. For example, during the Authentication phase of the SMTP conversation, the client-supplied username and password can be verified against an Authentication Repository using a circuit containing an **Attribute Authentication** filter. For details on possible authentication scenarios, see the section on [SMTP Authentication](#).

To configure a circuit handler for the **AUTH** command, perform the following steps:

1. On the **Services** tab, select the **SMTP** tree node under the Process.
2. Right-click, and select **Add Circuit Handler -> AUTH**.

3. In the **Configure AUTH** dialog, in the **Policy** tree, select the policy that you wish to handle the **AUTH** command.
4. **Click OK.**

Message attributes

The following message attributes are generated during processing:

<i>Message Attribute</i>	<i>Description</i>
authentication.subject.id	The username supplied by the client.
authentication.subject.password	The password supplied by the client.
monitoring.enabled	true if monitoring is enabled for the protocol, otherwise false.
message.monitoring.enabled	true if message-monitoring is enabled for the protocol, otherwise false.

Adding a MAIL Circuit Handler

The **MAIL** policy circuit handler is invoked when a **MAIL FROM SMTP** command is received from a client. Emails can be rejected based on wildcard matching of the supplied sender address in the **MAIL FROM SMTP** command. For example, email addresses containing **GMAIL.COM** (fromAddress of ***@gmail.com**) as the domain could be accepted using a simple **True** filter. Whereas, email addresses containing **YAHOO.COM** (fromAddress of ***@yahoo.com**) could be rejected using a simple **False** filter.

To configure a policy circuit handler for the **MAIL FROM** command, perform the following steps:

1. On the **Services** tab, select the **SMTP** tree node under the **Process**.
2. Right-click, and select **Add Circuit Handler -> MAIL**.
3. In the **Configure MAIL Address** dialog, you must specify the **From Address**. This is an email address used to filter addresses specified in the **MAIL FROM SMTP** command. You can specify this as a wildcard. The following are some example values:

<i>From Address</i>	<i>Description</i>
*	Runs the circuit for any email address received.
*@gmail.com	Runs the circuit for all email addresses with the domain gmail.com.
S*@oracle.*	Runs the circuit for all email addresses with any oracle domain, and beginning with the letter s.

The circuit selection is performed on a best-match basis.

4. In the **Policy** tree, select the policy that you wish to handle the **MAIL FROM** command.
5. **Click OK.**

You can configure multiple **MAIL** handlers so that different filter circuits are executed, depending on the received mail address.

Message attributes

The following message attributes are generated during processing:

<i>Message Attribute</i>	<i>Description</i>
smtp.mail.from	The email address specified in the MAIL FROM SMTP command received from the client.
monitoring.enabled	true if monitoring is enabled for the protocol, otherwise false.
message.monitoring.enabled	true if message-monitoring is enabled for the protocol, otherwise false.

Adding a RCPT Circuit Handler

The **RCPT** policy circuit handler is invoked when a RCPT TO SMTP command is received from a client. You can use this handler to filter addresses specified in the RCPT TO SMTP command. Recipients can be rejected based on wildcard matching of the supplied recipient address in the RCPT SMTP command. For example, recipient addresses containing GMAIL.COM (toAddress of *@gmail.com) as the domain could be accepted using a simple **True** filter. Whereas, addresses containing YAHOO.COM (toAddress of *@yahoo.com) could be rejected using a simple **False** filter. You can configure multiple **RCPT** handlers so that different filter circuits are executed, depending on the received email address.

To configure a circuit handler for the RCPT TO command, perform the following steps:

1. On the **Services** tab, select the SMTP tree node under the Process.
2. Right-click, and select **Add Circuit Handler -> RCPT**.
3. In the **Configure Recipient Address** dialog, you must specify the **To Address**. This is an email address used to filter addresses specified in the RCPT TO SMTP command. You can specify this as a wildcard. The following are some example values:

<i>To Address</i>	<i>Description</i>
*	Runs the circuit for any email address received.
*@oracle.com	Runs the circuit for all email addresses with the domain oracle.com.
d*@yahoo.*	Runs the circuit for all email addresses with any yahoo domain, and beginning with the letter d.

The circuit selection is performed on a best-match basis.

4. In the **Policy** tree, select the policy that you wish to handle the MAIL FROM command.
5. **Click OK**.

Message attributes

The following message attributes are generated during processing:

<i>Message Attribute</i>	<i>Description</i>
smtp.rcpt.to	The email address specified in the RCPT TO SMTP command received from the client.
monitoring.enabled	true if monitoring is enabled for the protocol, otherwise false.
message.monitoring.enabled	true if message-monitoring is enabled for the protocol, otherwise false.

Adding a DATA Circuit Handler

The **DATA** policy circuit handler is invoked when a **DATA** SMTP command is received from a client. For example, for emails that contain SOAP/XML content, you can add an XML signature to the XML data, stored in the `content.body` message attribute, using an **XML Signature Generation** filter. For emails containing attachments, the attached mail data can be run through one of the Enterprise Gateway anti-virus filters. Alternatively, you can use **SMIME Encrypt** or **SMIME Decrypt** filters to encrypt or decrypt emails (including attachments) passing through the Enterprise Gateway. You can also digitally sign emails using an **SMIME Sign** filter, or verify signatures on already digitally signed emails using an **SMIME Verify** filter.

To configure a policy circuit handler for the **DATA** command, perform the following steps:

1. On the **Services** tab, select the **SMTP** tree node under the Process.
2. Right-click, and select **Add Circuit Handler -> DATA**.
3. In the **Policy** tree, select the policy that you wish to handle the **DATA** command.
4. **Click OK**.

Message attributes

The following message attributes are added during processing:

<i>Message Attribute</i>	<i>Description</i>
<code>content.body</code>	The stream representing the body of the mail. Note that the <code>content.body</code> does not include MIME headers.
<code>monitoring.enabled</code>	true if monitoring is enabled for the protocol, otherwise false.
<code>message.monitoring.enabled</code>	true if message monitoring is enabled for the protocol, otherwise false.

SMTP Authentication

The SMTP protocol supports Extended SMTP (ESMTP) PLAIN authentication. The following matrix shows the possible authentication scenarios and actions based on the SMTP Services configuration:

<i>Scenario</i>	<i>AUTH Handler</i>	<i>AUTH name</i> <i>User- and Password</i>	<i>Mail Server ad- vertises AUTH</i>	<i>Enterprise Gateway ad- vertises AUTH</i>	<i>Proxy client AUTH</i>	<i>Authenticate Enterprise Gateway to Server</i>
1	No	No	No	No	No	No
2	No	No	Yes	Yes	Yes	No
3	No	Yes	No	No	No	No
4	No	Yes	Yes	No	No	Yes
5	Yes	No	No	Yes	No	No
6	Yes	No	Yes	Yes	No	No
7	Yes	Yes	No	Yes	No	No
8	Yes	Yes	Yes	Yes	No	Yes

These authentication scenarios are described as follows:

1. No authentication username and password are specified so the Enterprise Gateway does not attempt to authenticate with the server. The server does not support authentication anyway. The mail server does not advertise authentication so the Enterprise Gateway does not advertise AUTH to the client. The client authentication is not proxied because the server does not support it.
2. No authentication username and password are specified so the Enterprise Gateway does not attempt to authenticate with the server. The server does not support authentication anyway. The mail server advertises AUTH, so the Enterprise Gateway advertises AUTH to the client. No AUTH handler is configured, so the client authentication details are proxied to the server.
3. Same as 1 above.
4. The authentication username and password are specified so the Enterprise Gateway authenticates with the server. The mail server advertises AUTH, but because a username and password are specified, the Enterprise Gateway does not advertise AUTH to the client because the Enterprise Gateway authenticates with the server using the configured credentials. This also implies no client authentication proxying.
5. No authentication username and password are specified so the Enterprise Gateway does not attempt to authenticate with the server. The server does not support authentication anyway. AUTH handler configured, which implies the Enterprise Gateway performs authentication, so advertise AUTH to the client.
6. Same as 5 above.
7. AUTH handler configured, which implies the Enterprise Gateway performs authentication, so advertise AUTH to the client. No proxying occurs because the Enterprise Gateway performs the authentication. No authentication is performed with the server because the server does not support it.
8. AUTH advertised to the client because the Enterprise Gateway performs authentication (and the mail server supports it). AUTH handler configured, which implies the Enterprise Gateway performs authentication. No proxying occurs because the Enterprise Gateway performs the authentication. Authentication is performed with the server because the server supports AUTH and a username and password is configured.

SMTP Content-Transfer-Encoding

The SMTP protocol supports automatic Content-Transfer-Encoding/Decoding. For `DATA` SMTP commands, the content of the incoming mail body may be encoded. To enable circuit filters to view and/or manipulate the raw body data, the contents are automatically decoded before circuit execution, and re-encoded afterwards (before being forwarded on to the configured outbound mail server).

Supported encodings

The following encodings are supported:

- Base64
- 7-bit
- 8-bit
- quoted-printable
- binary

However, Base64 is the only encoding that results in decoding/re-encoding of the mail data.

Multi-part MIME content, generally used for sending attachments in SMTP, is also supported. Each separate body in the multi-part is checked for a Content-Transfer-Encoding, and the decoding/re-encoding is performed as appropriate.

Deployment Example

This section provides a step-by-step example of how to configure and deploy SMTP services using the Enterprise Gateway. In this example, the Enterprise Gateway acts as a relay between a Thunderbird email client and the Google Gmail service.

Configuring the Enterprise Gateway SMTP Services

The Enterprise Gateway connects to the Gmail STARTTLS interface, which is available at `smtp.gmail.com`, and listening on port 587. To configure the SMTP Services, perform the following steps in the Policy Studio:

1. On the **Services** tab on the left, select a Process tree node (for example, the default **Oracle Enterprise Gateway**).
2. Right-click, and select **Add SMTP Services**.
3. Enter `smtp.gmail.com` for the **Host**.
4. Enter 587 for the **Port**.
5. Select **STARTTLS** from the **Connection Security** drop-down list. This is selected because `smtp.gmail.com:587` exposes the Gmail STARTTLS SMTP interface.
6. Because STARTTLS has the potential to be upgraded to a secure connection, you must also select some **Trusted Certificates**.
7. Accept all other defaults, and click **OK** to add the SMTP services.

Configuring the SMTP Client Interface

To configure a STARTTLS client interface, perform the following steps in the Policy Studio:

1. Right-click the **SMTP Services** node, and select **Add Interface -> STARTTLS**.
2. Enter a **Port** (for example, 8026). This is the port on which the Enterprise Gateway's incoming SMTP traffic is accepted. You can enter any port that is not already in use.
3. Because STARTTLS has the potential to be upgraded to a secure connection, you must configure a trusted certificate. Click the **X.509 Certificate** button.
4. Select a certificate in the **Select Certificate** dialog.
5. Click **OK** to return to the **Configure STARTTLS Interface** dialog.
6. When the certificate has been configured, accept all other defaults, and click **OK** to add the incoming STARTTLS interface.

When the SMTP services and STARTTLS client interface have been configured, you must deploy the changes to the Enterprise Gateway.

Configuring Thunderbird Client Settings

This example uses Thunderbird as the email client. However, you can use any standard email client that supports SMTP. Thunderbird is available as a free download from <http://www.mozillamessaging.com/>.

To configure a STARTTLS outgoing server in your Thunderbird client, perform the following steps:

1. Launch the Thunderbird email client.
2. From the main menu, select **Tools -> Account Settings**.
3. Expand the **Local Folders** tree node in the left pane.
4. Select the **Outgoing Server** node to create a new outgoing server configuration.
5. Click **Add** to display the **SMTP Server** dialog.
6. Enter `Oracle Enterprise Gateway [STARTTLS]` in the **Description** field.
7. Enter `localhost` (or the IP Address of the machine on which the Enterprise Gateway service is running) in the **Server Name** field.
8. Enter 8026 in the **Port** field. This will send SMTP traffic to the STARTTLS interface configured above, so the ports must match.
9. Select **STARTTLS** from the **Connection security** drop-down list. Traffic on this connection may be upgraded to secure during the SMTP conversation.
10. Select **Normal Password** from the **Authentication method** drop-down list. This indicates that Authentication will be performed.
11. Enter a valid Gmail user-name for the **User Name**.

12. Click **OK** to add the new outgoing server configuration.

Configuring Certificates in Thunderbird

To enable Thunderbird to successfully negotiate the STARTTLS conversation with the Enterprise Gateway, you must import a CA certificate into Thunderbird. This is also because a certificate was already generated and imported into the Enterprise Gateway when configuring its STARTTLS client interface.

To configure a STARTTLS outgoing server in your Thunderbird client, perform the following steps:

1. From the Thunderbird main menu, select **Tools -> Options**.
2. Select the **Certificates** tab.
3. Click the **View Certificates** button, to display the **Certificate Manager** dialog.
4. Click **Import**, and import the appropriate CA certificate.
5. Click **OK** when finished.

Testing the STARTTLS Client Interface

To test the STARTTLS client interface using Thunderbird, perform the following steps:

1. Launch the Thunderbird email client, and create a new mail message.
2. Enter a valid Gmail address in the **To** field.
3. Enter Enterprise Gateway Test as the **Subject**.
4. Enter This mail has been sent using Oracle Enterprise Gateway in the mail body.
5. To specify the appropriate outgoing mail server, select **Tools -> Account Settings** from the main menu.
6. Select Oracle Enterprise Gateway [STARTTLS] - localhost from the **Outgoing Server** drop-down list.
7. Click **OK**.
8. Send the mail.

The following example from the Enterprise Gateway trace shows the SMTP commands that occur. Commands marked in **bold text** shows traffic from the Thunderbird client to the Enterprise Gateway and vice versa. Commands marked in **bold italics** shows traffic from the Enterprise Gateway to the Gmail server at smtp.gmail.com:587, and vice versa.

```
DEBUG 14:46:46:546 [14b4] incoming call on interface *:8026 from 127.0.0.1:1487
DEBUG 14:46:46:546 [14b4] new connection 08133248, settings source incoming interface
(force 1.0=no, idleTimeout=60000, activeTimeout=60000)
DATA 14:46:46:546 [14b4] snd 0018: <220 doejOracle>
DATA 14:46:46:562 [14b4] rcv 18: <EHLO [127.0.0.1]>
DEBUG 14:46:46:562 [14b4] 080BE260: new connection cache set SMTP Client
DEBUG 14:46:46:562 [159c] idle connection monitor thread running
DEBUG 14:46:46:562 [14b4] new endpoint smtp.gmail.com:587
DEBUG 14:46:46:640 [14b4] Resolved smtp.gmail.com:587 to:
DEBUG 14:46:46:640 [14b4] 209.85.227.109:587
DEBUG 14:46:46:718 [14b4] connected to 209.85.227.109:587
DEBUG 14:46:46:718 [14b4] new connection 08135BA0, settings source service-wide defaults
(force 1.0=no, idleTimeout=15000, activeTimeout=30000)
DATA 14:46:46:765 [14b4] rcv 44: <220 mx.google.com ESMTP v11sm7979387weq.40>
DATA 14:46:46:765 [14b4] snd 0018: <ehlo [127.0.0.1]>
DATA 14:46:46:812 [14b4] rcv 125: <250-mx.google.com at your service, [87.198.245.194]
250-SIZE 35651584
250-8BITMIME
250-STARTTLS
250 ENHANCEDSTATUSCODES>
DATA 14:46:46:812 [14b4] snd 0010: <starttls>
DATA 14:46:46:843 [14b4] rcv 30: <220 2.0.0 Ready to start TLS>
DEBUG 14:46:46:843 [14b4] push SSL protocol on to connection
```

```

DEBUG 14:46:46:906 [14b4] No SSL host name provided: using default certificate for interface
DEBUG 14:46:46:906 [14b4] verifyCert: preverify=1, depth=2, subject /C=US/O=Equifax/OU=Equifax
Secure Certificate Authority, issuer /C=US/O=Equifax/OU=Equifax Secure Certificate Authority
DEBUG 14:46:46:906 [14b4] ca cert? 1
DEBUG 14:46:46:906 [14b4] verifyCert: preverify=1, depth=1, subject /O=Google Inc/CN=Google
Internet Authority, issuer /C=US/O=Equifax/OU=Equifax Secure Certificate Authority
DEBUG 14:46:46:906 [14b4] verifyCert: preverify=1, depth=0, subject
/C=US/ST=California/L=Mountain View/O=Google Inc/CN=smtp.gmail.com,
issuer /C=US/O=Google Inc/CN=Google Internet Authority
DEBUG 14:46:46:952 [14b4] negotiated SSL cipher "RC4-MD5", session 00000000 (not reused)
DATA 14:46:46:952 [14b4] snd 0018: <ehlo [127.0.0.1]>
DATA 14:46:46:999 [14b4] rcv 140: <250-mx.google.com at your service, [87.198.245.194]
250-SIZE 35651584
250-8BITMIME
250-AUTH LOGIN PLAIN XOAUTH
250 ENHANCEDSTATUSCODES>
DATA 14:46:46:999 [14b4] snd 0109: <250-OracleEnterprise Gateway Hello [127.0.0.1]
250-SIZE 35651584
250-8BITMIME
250-STARTTLS
250 ENHANCEDSTATUSCODES>
DEBUG 14:46:46:999 [14b4] delete transaction 0B95D2C0 on connection 08133248
DATA 14:46:46:999 [14b4] rcv 10: <STARTTLS>
DATA 14:46:46:999 [14b4] snd 0014: <220 Go ahead>
DEBUG 14:46:46:999 [14b4] push SSL protocol on to connection
DEBUG 14:46:46:999 [14b4] Servername CB: SSL host name: localhost, not in host map -
using default certificate for interface
DEBUG 14:46:47:031 [14b4] negotiated SSL cipher "AES256-SHA", session 00000000 (not reused)
DATA 14:46:47:031 [14b4] rcv 18: <EHLO [127.0.0.1]>
DATA 14:46:47:031 [14b4] snd 0018: <ehlo [127.0.0.1]>
DATA 14:46:47:077 [14b4] rcv 140: <250-mx.google.com at your service, [87.198.245.194]
250-SIZE 35651584
250-8BITMIME
250-AUTH LOGIN PLAIN XOAUTH
250 ENHANCEDSTATUSCODES>
DATA 14:46:47:077 [14b4] snd 0124: <250-OracleEnterprise Gateway Hello [127.0.0.1]
250-SIZE 35651584
250-8BITMIME
250-AUTH LOGIN PLAIN XOAUTH
250 ENHANCEDSTATUSCODES>
DEBUG 14:46:47:077 [14b4] delete transaction 0B95D2C0 on connection 08133248
DATA 14:46:47:077 [14b4] rcv 41: <AUTH PLAIN ADGzaHllaDe0SHFlex2r82Su555=>
DATA 14:46:47:077 [14b4] snd 0041: <auth PLAIN ADGzaHllaDe0SHFlex2r82Su555=>
DATA 14:46:47:718 [14b4] rcv 20: <235 2.7.0 Accepted>
DATA 14:46:47:718 [14b4] snd 0020: <235 2.7.0 Accepted>
DATA 14:46:47:718 [14b4] rcv 45: <MAIL FROM:<john.doe@oracle.com> SIZE=444>
DATA 14:46:47:718 [14b4] snd 0036: <mail from:<john.doe@oracle.com>>
DATA 14:46:47:765 [14b4] rcv 33: <250 2.1.0 OK v11sm7979387weq.40>
DATA 14:46:47:765 [14b4] snd 0033: <250 2.1.0 OK v11sm7979387weq.40>
DATA 14:46:47:765 [14b4] rcv 30: <RCPT TO:<test@gmail.com>>
DATA 14:46:47:765 [14b4] snd 0030: <rcpt to:<test@gmail.com>>
DATA 14:46:47:812 [14b4] rcv 33: <250 2.1.5 OK v11sm7979387weq.40>
DATA 14:46:47:812 [14b4] snd 0033: <250 2.1.5 OK v11sm7979387weq.40>
DATA 14:46:47:812 [14b4] rcv 6: <DATA>
DATA 14:46:47:812 [14b4] snd 0006: <data>
DATA 14:46:48:609 [14b4] rcv 34: <354 Go ahead v11sm7979387weq.40>
DATA 14:46:48:609 [14b4] snd 0008: <354 OK>
DATA 14:46:48:609 [14b4] rcv 447: <Message-ID: <4CB85B46.4060205@oracle.com>
Date: Fri, 15 Oct 2010 14:46:46 +0100
From: John Doe <john.doe@oracle.com>
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.2.9) Gecko/20100915
Thunderbird/3.1.4
MIME-Version: 1.0
To: test@gmail.com
Subject: Enterprise Gateway Test

```

Content-Type: text/plain; charset=ISO-8859-1; format=flowed
 Content-Transfer-Encoding: 7bit

This mail has been sent via Oracle Enterprise Gateway

```
.>
DATA      14:46:48:609 [14b4] snd 0442: <Message-ID: <4CB85B46.4060205@oracle.com>
Date: Fri, 15 Oct 2010 14:46:46 +0100
From: John Doe <john.doe@oracle.com>
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.2.9) Gecko/20100915
Thunderbird/3.1.4
MIME-Version: 1.0
To: test@gmail.com
Subject: Enterprise Gateway Test
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit
```

This mail has been sent via Oracle Enterprise Gateway>

```
DATA      14:46:48:609 [14b4] snd 0005: <.>
DATA      14:46:49:874 [14b4] rcv 44: <250 2.0.0 OK 1287150409 v11sm7979387weq.40>
DATA      14:46:49:874 [14b4] snd 0044: <250 2.0.0 OK 1287150409 v11sm7979387weq.40>
DEBUG     14:46:49:874 [14b4] delete transaction 0B95D2C0 on connection 08133248
DATA      14:46:49:874 [14b4] rcv 6: <QUIT>
DATA      14:46:49:874 [14b4] snd 0006: <quit>
DATA      14:46:49:921 [14b4] rcv 49: <221 2.0.0 closing connection v11sm7979387weq.40>
DEBUG     14:46:49:921 [14b4] delete transaction 08040BD8 on connection 08135BA0
DATA      14:46:49:921 [14b4] snd 0049: <221 2.0.0 closing connection v11sm7979387weq.40>
DEBUG     14:46:49:921 [14b4] delete transaction 0B95D2C0 on connection 08133248
DEBUG     14:46:49:921 [14b4] delete connection 08133248, current transaction 00000000
```

Policy Execution Scheduling

Overview

You can configure a policy execution scheduler at the level of the Enterprise Gateway process. This enables you to schedule the execution of any policy on a specified date and time in a recurring manner. The Enterprise Gateway provides a pre-configured library of schedules to select from when creating a policy execution scheduler. You can also add your own schedules to the globally available library in the Policy Studio.

You can use policy execution scheduling in any policy circuit (for example, to perform a message health check). This feature is also useful when polling a service to enforce a Service Level Agreement (for example, to ensure the response time is less than 1000 ms, and if not, to send an alert).

Cron Expressions

In the Policy Studio, policy execution schedules are based on cron expressions. A cron expression is a string that specifies a time schedule for executing a policy. It consists of six required fields and one optional field, each separated by a space, which together specify when run the policy. For example, the following expression specifies to run at 10:15am every Monday, Tuesday, Wednesday, Thursday and Friday in 2011:

```
0 15 10 ? * MON-FRI 2011
```

Syntax

The following table shows the syntax used for each field:

Field	Values	Special Characters
Seconds	0-59	, - * /
Minutes	0-59	, - * /
Hours	0-23	, - * /
Day of Month	1-31	, - * ? / L W
Month	1-12 or JAN-DEC	, - * /
Day of Week	1-7 or SUN-SAT	, - * ? / L #
Year (optional)	empty or 1970-2199	, - * /

Special Characters

The special characters are explained as follows:

Special Character	Description
,	Separates values in a list (for example, MON,WED,SAT means Mondays, Wednesdays and Saturdays only).
-	Specifies a range of values (for example, 2011-2015 means every year between 2011 and 2015 inclusive).
*	Specifies all values of the field (for example, every minute).

Special Character	Description
?	Specifies no value in the Day of Month and Day of Week fields. This enables you to specify a value in one field, but not in the other.
/	Specifies time increments (for example, in the Minutes field, 0/15 means minutes 0, 15, 30, and 45, while 5/15 means minutes 5, 20, 35, and 50). Specifying * before the / is the same as specifying 0 as the start value. The / character enables you to turn on every nth value in the set of values for the specified field. For example, 7/6 in the month field only turns on month 7, and does not mean every 6th month.
L	Specifies the last value in the Day of Month and Day of Week fields. In the Day of Month field, this means the last day of the month (for example, January 31, or February 28 in non-leap years). In the Day of Week field, when used alone, this means 7 or SAT. When used after another value, it means the last xxx day of the month (for example, 5L means the last Thursday of the month). When using the L character, do not specify lists or ranges because this can give confusing results.
W	Specifies the weekday (Monday-Friday) nearest the given day. For example, 15W means the nearest weekday to the 15th of the month. If the 15th is a Saturday, the policy runs on Friday 14th. If the 15th is a Sunday, the policy runs on Monday 16th. If the 15th is a Tuesday, it runs on Tuesday 15th. However, if you specify 1W, and the 1st is a Saturday, the policy runs on Monday 3rd to avoid crossing the month boundary. You can only specify the W character for a single day, and not a range or list of days.
#	Specifies the nth xxx weekday of the month in the Day of Week field. For example, a value of FRI#2 means the second Friday of the month. However, if you specify #5, and there are not 5 of the specified Day of the Week in the month, no policy is run that month. When the # character is specified, there can only be one expression in the Day of Week field (for example, 2#1,6#4 is not valid because there are two expressions).

Examples

The following are some of the cron expressions provided in the **Schedule Library** in the Policy Studio:

Cron Expression	Description
0 15 10 ? * *	Run at 10:15am every day.
0 15 10 ? * 6L 2011-2015	Run at 10:15am on every last Friday of every month during the years 2011, 2012, 2013, 2014, and 2015.
0 15 10 ? * 6#3	Run at 10:15am on the third Friday of every month.
0 0 10 1,15 * ?	Run at 10am on the 1st and 15th days of the month.
0 10,44 14 ? 3 WED	Run at 2:10pm and at 2:44pm every Wednesday in the month of March.

Cron Expression	Description
0,30 * * ? * SAT,SUN	Run every 30 seconds but only on Weekends (Saturday and Sunday).
0 0/5 14,18 * * ?	Run every 5 minutes starting at 2pm and ending at 2:55pm, and every 5 minutes starting at 6pm and ending at 6:55pm, every day.
0 0-5 14 * * ?	Run every minute starting at 2pm and ending at 2:05pm, every day.

Important Notes:

Please note the following:

- Support for specifying both a Day of Week and a Day of Month value is not complete. You must use the ? or * character in one of these fields.
- Overflowing ranges with a larger number on the left than the right are supported (for example, 21-2 for 9pm until 2am , or OCT-MAR). However, overuse may cause problems with daylight savings (for example, 0 0 14-6 ? * FRI-MON).

Adding a Schedule

To add a schedule to the globally available library in the Policy Studio, perform the following steps:

1. Click the **Policies** tab on the left.
2. Select the **Schedule Library** node.
3. Click the **Add** button at the bottom of the **Schedules** screen.
4. In the **Schedules** dialog, enter a **Name** (for example, Run every 30 seconds).
5. Enter a **Cron expression** (for example, 0/30 * * * * ?).
6. Click **OK**.

You can also edit or delete a selected schedule using the appropriate button.

Adding a Policy Execution Scheduler

To add a policy execution scheduler in the Policy Studio, perform the following steps:

1. Click the **Services** tab on the left.
2. Right-click the Process node (for example, the default **Oracle Enterprise Gateway**), and select **Add policy execution scheduler**.
3. Click the button next to the **Schedule** field, select a cron expression in the dialog, and click **OK**.
4. Click the button next to the **Policy** field, select a policy in the dialog, and click **OK**.
5. Click **OK**.

Directory Scanner

Overview

The Directory Scanner allows you to scan a certain directory on the file system for files containing XML messages. Once the messages have been read, they can be passed into the core message pipeline where the full collection of message processing filters can act on them.

The Directory Scanner is typically used in cases where an external application is dropping XML files (perhaps by FTP) on to the file system so that they can be validated, modified, and potentially routed onwards over HTTP, JMS, or simply stored back to another directory where the application can pick them up again.

This sort of protocol mediation is very useful in cases where legacy systems are involved. Instead of making drastic changes to the legacy system by adding an HTTP engine to it, for example, the Enterprise Gateway can be used to pull the files from the file system and then route them onwards over HTTP to another back-end system. The added benefit here, of course, is that messages are exposed to the full compliment of message processing filters made available by the Enterprise Gateway. This ensures that only properly validated messages will be routed on to the target system.

To add a new Directory Scanner, right-click the name of the Process in the tree view of the Policy Studio (for example, "Oracle Enterprise Gateway"), and select the **Directory Scanner -> Add** menu option. The following sections describe how to configure the fields on the **Directory Scanner Settings** dialog.

Directory to Scan

The fields configured in this section determine what files to scan, where to scan for them, and when to scan.

Name:

Enter or browse to the directory that the Enterprise Gateway will scan for XML files.

File Name Pattern:

If you only want to scan certain files based on some pattern, you can enter the pattern here. For example, if you only want to scan for files with a particular file extension, you can enter the file extension here. Similarly, if a particular naming scheme is used when dropping the files into the directory specified above, you can enter a pattern here to only scan for these files.

Poll Rate:

The poll rate entered here (in milliseconds) determines how often the Enterprise Gateway scans the directory for new XML files.

Directory for Output

Once the Directory Scanner has finished scanning the files it will move them to another directory to avoid processing them again. The fields configured in this section determine where processed files will be placed and what they will be called.

Name:

Enter the name of the directory to place the processed files. This may either be the response from the Web Service (after the policy has routed it onwards and received a response) or a modified message in cases where the policy has inserted a security token into the message or converted the message using an XSLT conversion, for example.

File Prefix:

If you would like to save processed files into the directory above with a prefix added to the filename, enter the prefix here. For example, you may want to prepend "_PROCESSED" to all processed files.

File Suffix:

Similarly you can add a suffix to the output files by entering the suffix in this field.

Completed Directory

Processed files will be removed from the source directory and placed into this directory post-processing to avoid re-processing the same files over and over again.

Working Directory

Files will be moved to the "temporary" directory specified here during processing. This is necessary in cases where the poll rate is quite low and there is a chance that the file may be scanned again before it is processed fully and moved to the completed directory.

Policy to Use

The messages contained within the scanned XML files will be passed into the policy selected here. If the policy routes messages to a Web Service, the response from the service will be placed in the output directory specified above. Similarly, if the policy modifies the request by signing it or adding a security token to it, for example, the updated message will also be placed in the output directory.

Packet Sniffers

Overview

Packet Sniffers are a type of Passive Service. Rather than opening up a TCP port and *actively* listening for requests, the Packet Sniffer *passively* reads raw data packets off the network interface. The Sniffer assembles these packets into complete messages that can then be passed into an associated policy.

Because the Packet Sniffer operates passively (i.e. does not listen on a TCP port) and, therefore, completely transparently to the client, it is most useful for monitoring and managing Web Services. For example, the Sniffer can be deployed on a machine running a Web Server acting as a container for Web Services. Assuming that the Web Server is listening on TCP port 80 for traffic, the Packet Sniffer can be configured to read all packets destined for port 80 (or any other port, if necessary). The packets can then be marshalled into complete HTTP/SOAP messages by the Sniffer and passed into a policy that logs the message to a database, for example.

Important Note:

Please note that on Linux and Solaris platforms, the Enterprise Gateway must be started by the root user in order to gain access to the raw packets.

Configuration

Since Packet Sniffers are mainly for use as passive monitoring agents, they are usually created within their own HTTP Service Group. So, for example, a new Service Group can be created for this purpose by right-clicking on the Process, selecting the **Add HTTP Services** menu option, and then entering "Packet Sniffer Group" on the **HTTP Services** dialog.

We can then add a Relative Path Service to this Group by right-clicking on the "Packet Sniffer Group" and selecting the **Add Relative Path** menu option. Enter a path in the field provided and select the policy that you want to dispatch messages to when the Packet Sniffer detects a request for this path (after it assembles the packets). So, for example, if the Relative Path is configured as "/a", and the Packet Sniffer assembles packets into a request for this path, the request will be dispatched to the policy selected in the Relative Path Service.

Finally, we can add the Packet Sniffer itself by right-clicking on the "Packet Sniffer Group" node, selecting **Packet Sniffer**, and then the **Add** menu option. Complete the following fields on the **Packet Sniffer** dialog:

Device to Monitor:

Enter the name or identifier of the network interface that the Packet Sniffer will monitor. The default entry here is "any", but it is important to note that this is only valid on Linux. On UNIX-based systems, network interfaces are usually identified using names like "eth0", "eth1", and so on. On Windows, these names are more complicated, for example, "\Device\NPF_{00B756E0-518A-4144 ... }".

Filter:

The Packet Sniffer can be configured to only intercept certain types of packets. For example, it can ignore all UDP packets, only intercept packets destined for port 80 on the network interface, ignore packets from a certain IP address, listen for all packets on the network, and so on.

The Packet Sniffer uses the **libpcap** library filter language to achieve this. This language has a complicated but powerful syntax that allows you to *filter* what packets are intercepted and what packets are ignored. As a general rule, the syntax consists of one or more expressions combined with conjunctions, such as "and", "or", and "not". The following table lists a few examples of common filters and explains what they filter:

<i>Filter Expression</i>	<i>What does it filter?</i>
port 80	Capture only traffic for the HTTP Port (i.e. 80).
host 192.168.0.1	Capture traffic to and from IP address 192.168.0.1.
tcp	Capture only TCP traffic.

host 192.168.0.1 and port 80	Capture traffic to and from port 80 on IP address 192.168.0.1.
tcp portrange 8080-8090	Capture all TCP traffic destined for ports from 8080 through to 8090.
tcp port 8080 and not src host 192.168.0.1	Capture all TCP traffic destined for port 8080 but not from IP address 192.168.0.1.

The default filter of "tcp" simply captures all TCP packets arriving on the network interface. For more information on how to configure filter expressions like these, please refer to the man pages of **tcpdump** man page, which can be found [here](http://www.tcpdump.org/tcpdump_man.html) [http://www.tcpdump.org/tcpdump_man.html].

Promiscuous Mode:

When listening in "promiscuous mode", the Packet Sniffer will capture all packets on the same Ethernet network, regardless of whether or not the packets are addressed to the network interface that the Sniffer is monitoring.

Messaging System

Overview

A *messaging system* is a loosely coupled, peer-to-peer facility where clients can send messages to, and receive messages from any other client. In a messaging system, a client sends a message to a messaging agent. The recipient of the message can then connect to the same agent and read the message. However, the sender and recipient of the message do not need to be available at the same time to communicate (for example, unlike HTTP). The sender and recipient need only know the name and address of the messaging agent to talk to.

The Java Messaging System (JMS) is an implementation of such a messaging system. It provides an API for creating, sending, receiving, and reading messages. Java-based applications can use it to connect to other messaging system implementations. A *JMS provider* can deliver messages synchronously or asynchronously, which means that the client can fire and forget messages or wait for a response before resuming processing. Furthermore, the JMS API ensures different levels of reliability in terms of message delivery. For example, it can ensure that the message is delivered once and only once, or at least once.

The Enterprise Gateway uses the JMS API to connect to other messaging systems that expose a JMS interface, including Oracle WebLogic Server, IBM MQSeries, JBoss Messaging, TIBCO EMS, IBM WebSphere Server, and Progress SonicMQ. As a consumer of a JMS queue or topic, the Enterprise Gateway can read XML messages and pass them into a policy for validation. These messages can then be routed on over HTTP or dropped on to another JMS queue or topic.

Important Note:

You must add the JMS provider's JAR files to the Enterprise Gateway classpath for this feature to function correctly. Copy the provider JAR files to the `INSTALL_DIR/ext/lib` folder, where `INSTALL_DIR` points to the root of your Enterprise Gateway installation.

Configuring a JMS Service

You can configure a global JMS service on the **External Connections** tab in Policy Studio by right-clicking the **JMS Services** node, and selecting **Add a JMS Service**. The details entered in the **JMS Service** dialog are used by the Enterprise Gateway to drop messages on to a JMS queue or topic. The **Messaging System** Connection filter uses JMS Services configured here to do this.

Alternatively, you can configure a JMS service at the Process level and configure the Enterprise Gateway to consume a JMS queue or topic. Right-click the Process on the **Services** tab in Policy Studio, and select **JMS Wizard**.

Configure the following fields on the **JMS Service** dialog:

Name:

Enter a descriptive name for the JMS Provider in the **Name** field.

Provider URL:

Enter the URL of the JMS provider (for example, `jnp://localhost:1099`).

Initial Context Factory:

The Enterprise Gateway uses a connection factory to create a connection with a JMS provider. A connection factory encapsulates a set of connection configuration parameters that have been defined by the administrator. For example, the initial context factory class for the JBoss application server is `org.jnp.interfaces.NamingContextFactory`.

Connection Factory:

Enter the name of the connection factory to use when connecting to the JMS provider. The name of the connection factory is vendor-specific. For example, the connection factory used for the JBoss application server is `org.jnp.interfaces:javax.jnp`.

Username:

If a user name is required to connect to this JMS provider, enter it in this field.

Password:

Enter the password for this user.

Custom Message Properties:

You can add JNDI context settings by clicking the **Add** button, and adding name and value properties in the fields.

When the JMS Service has been configured, you can configure the Enterprise Gateway to drop messages on to a queue or topic exposed by this service by selecting it from the **JMS Service** field on the **Messaging System** Connection filter dialog when configuring a policy. For more details, see the [Messaging System Filter](#) topic.

You can also configure JMS sessions for the newly added JMS Service at the Process level. For more details, see the next section.

Configuring a JMS Session

JMS services have JMS sessions, which can be shared by multiple JMS consumers, or used by a single JMS consumer only. To configure a JMS session, right-click the Process on the **Services** tab in the Policy Studio, and select **Messaging System** -> **Add JMS Session**. Alternatively, you can configure a JMS session using the JMS Wizard.

Configure the following fields on the JMS Session screen:

Service

Select a pre-configured JMS Service from the drop-down list. For more details, see [Configuring a JMS Service](#).

Allow Duplicates

Typically, a JMS session operates in *auto* mode, meaning that messages received by the session are automatically acknowledged to guarantee once-only message delivery. By selecting the **Allow Duplicates** checkbox, you are configuring the JMS session to operate in *duplicates okay* mode. This means that messages are acknowledged lazily by the JMS session with the result that duplicate messages are possible. If messages are not automatically acknowledged, the client has no way of telling whether the JMS consumer received the message, and so may re-send the message. Running in this mode reduces the overhead associated with the guarantee of once-only message delivery offered by *auto* mode.

Listener Count

Specify the number of listeners permitted for this JMS session. Defaults to 1.

Configuring a JMS Consumer

You can configure multiple JMS consumers under a single JMS session, which share that session. Alternatively, you can configure a single JMS consumer per JMS session. Consumers sharing a JMS session access that session serially. Each consumer blocks until a response (if any) is received. Consumers with their own session do not encounter this problem, which may improve performance.

You can configure JMS consumers using the JMS Wizard, or by right-clicking an existing JMS session, and selecting **Add JMS Consumer**.

Configure the following fields on the JMS consumer screen:

Source:

Enter the name of the queue or topic from which you want to consume JMS messages.

Selector:

The entered expression specifies the messages that the consumer is interested in receiving. Using a selector, the task of filtering the messages is performed by the JMS provider instead of by the consumer. The selector is a string that specifies an expression whose syntax is based on the SQL-92 conditional expression syntax. The Process only receives messages whose headers and properties match the selector.

Extraction Method:

Specify how to extract the data from the JMS message from the drop-down list:

- Create a `content.body` attribute based on the SOAP over JMS draft specification (the default)
- Insert the JMS message directly into the attribute named below
- Populate the attribute below with the value inferred from message type to Java

Attribute Name:

The name of the message attribute that holds the data extracted from the JMS message. Defaults to the `jms.message` message attribute.

Policy:

Select the checkbox for the appropriate policy to run on the JMS message after it has been consumed by the Enterprise Gateway.

Configuring the JMS Wizard

You can use the **JMS Wizard** to configure a Process to consume JMS messages from a JMS queue or topic. When a message has been consumed by the Enterprise Gateway, it can be dispatched to a specified policy where the full complement of message filters can run on the message. The message can then be routed onwards over HTTP or dropped back on to a JMS queue or topic.

To launch the **JMS Wizard**, right-click the Process on the **Services** tab in the Policy Studio, and select **Messaging System -> JMS Wizard**. The wizard includes the following configuration screens:

JMS Service Provider

The first screen in the wizard enables you to configure connection details to the JMS provider that produces the JMS messages that are consumed by the Enterprise Gateway. For details on configuring the fields on this screen, see the [Configuring a JMS Service](#) section.

JMS Session Configuration

The second screen in the wizard enables you to configure the **Allow Duplicates** option for the JMS session that is established with the JMS provider. For details on configuring this option, see the [Configuring a JMS Session](#) section.

JMS Consumer Configuration

The third screen in the wizard enables you to configure JMS consumer settings. For details on configuring the fields on this screen, see the [Configuring a JMS Consumer](#) section.

Remote Host Settings

Overview

You can use the **Remote Host Settings** to configure the way in which the Enterprise Gateway connects to a specific external server or routing destination. For example, typical use cases for configuring Remote Hosts with the Enterprise Gateway are as follows:

- Forcing the Enterprise Gateway to send *only* HTTP 1.0 requests to a destination server because that server supports only HTTP 1.0.
- Resolving inconsistencies in the way the destination server supports HTTP.
- Mapping a hostname to a specific IP address or addresses (for example, if a DNS server is unreliable or unavailable).
- Setting the timeout, session cache size, input/output buffer size, and other connection-specific settings for a destination server (for example, if the destination server is particularly slow, you can set a longer timeout).
- Stop accepting inbound connections on the HTTP Interface when the Enterprise Gateway loses connectivity to the remote host.

You can add Remote Hosts *per-process* by right-clicking the Process in the Policy Studio tree view, and selecting **Add Remote Host**. The tabs in the **Remote Host Settings** configuration screen are described in the next sections.

General Settings

You can configure the following settings on the **General** tab:

Host Name:

The host name or IP address of the Remote Host to connect to. If the host name entered in a **Static Router** filter matches this host name, the connection-specific settings configured on the **Remote Host** dialog are used when connecting to this host. This also includes any IP addresses listed on the **Addresses** tab, which override the default network DNS server mappings, if configured.

Port:

The TCP port on the Remote Host to connect to.

Maximum Connections:

The maximum number of connections to open to a Remote Host. If the maximum number of connections has already been established, the Enterprise Gateway Process waits for a connection to drop or become idle before making another request. The default maximum is 128 connections.

Force HTTP 1.0:

In cases where the Enterprise Gateway is routing on to a Remote Host that does not fully support the HTTP 1.1 protocol, anomalies may occur during the connection. To prevent this, you can force the Enterprise Gateway to use the HTTP 1.0 protocol.

Include Content Length in Request:

When this option is selected, the Enterprise Gateway includes the Content-length HTTP header in all requests to this Remote Host.

Include Content Length in Response:

When this option is selected, if the Enterprise Gateway receives a response from this Remote Host that contains a Content-length HTTP header, it returns this length to the client.

Send Server Name Indication TLS extension to server:

Adds a field to outbound TLS/SSL calls that shows the name that the client used to connect. For example, this can be useful if the server handles several different domains, and needs to present different certificates depending on the name

the client used to connect.

Verify server's certificate matches requested hostname:

Ensures that the certificate presented by the server matches the name of the remote host being connected to. This prevents host spoofing and man-in-the-middle attacks. This setting is selected by default.

Address Settings

You can configure the following settings on the **Addresses** tab:

Addresses to use instead of DNS lookup:

You can add a list of IP addresses that the Enterprise Gateway uses instead of attempting a DNS lookup on the host name provided. This is useful in cases where a DNS server is not available or is unreliable. By default, connection attempts are made to the listed IP addresses on a round-robin basis.

For example, if a **Static Router** filter is configured to route to `www.webservice.com`, it first checks if any **Remote Hosts** have been configured with a **Host Name** entry matching `www.webservice.com`. If it finds a **Remote Host** with matching **Host Name**, it resolves the hostname to the IP addresses listed here. In addition, it uses all the connection-specific settings configured on the **Remote Host** dialog when routing messages to these IP addresses. If it can not find a matching host, the **Static Router** filter uses whatever DNS server has been configured for the network on which the Enterprise Gateway is running.

To add a list of IP addresses for a Remote Host, perform the following steps:

1. In the **Addresses to use instead of DNS lookup** box, select a priority group (for example, **Highest Priority**).
2. Click **Add**.
3. Enter an IP address in the **Configure IP Address** dialog.
4. Click **OK**.
5. Repeat these steps to add more IP addresses as appropriate.

Load balancing:

The **Load Balancing Algorithm** drop-down box enables you to specify whether load balancing is performed on a simple round-robin basis or weighted by response time. **Simple Round Robin** is the default algorithm. Connection attempts are made to the listed IP addresses on a round-robin basis in each priority group. The **Weighted by response time** algorithm compares the request/reply response times for the server address in each priority group. This is the simplest way of estimating the relative load of the address. This algorithm works as follows:

1. The address with the least response time is selected to send the next message to.
2. If the address fails to send the message, it ignores that address for a period of time and selects another address in the same way.
3. If all addresses in a given group fail to accept a connection, addresses in the next group in ascending order of priority are used in the same way.
4. Only when all addresses in all priorities have failed to accept connections is delivery of the message abandoned, and an error raised.

The response times used by this algorithm decline over time. You can specify the rate of exponential decline by specifying a **Period to wait before response time is halved**. The default is 10,000 ms (10 sec). This enables addresses that were heavily loaded for a period of time to eventually resume accepting messages after the load subsides. For example, server A takes 100 ms to reply, and the other servers in the same priority group reply in 25 ms. A **Period to wait before response time is halved** of 10,000 ms (10 sec) means that after 20 seconds server A is retried along with the other servers. In this case, the response time has been halved twice ($100 \text{ ms} / 2 / 2 = 25 \text{ ms}$).

Advanced Settings

The options available on the **Advanced** tab are used when creating sockets for connecting to the Remote Host. Default

values are provided for all fields, which should only be modified under advice from the [Oracle Support Team](#).

You can configure the following configuration options on the **Advanced** tab:

Active Timeout:

When the Enterprise Gateway receives a large HTTP request, it reads the request off the network when it becomes available. If the time between reading successive blocks of data exceeds the **Active Timeout**, the Enterprise Gateway closes the connection. This prevents a Remote Host from closing the connection while sending data. For example, the Remote Host's network connection is pulled out of the machine while sending data to the Enterprise Gateway. When the Enterprise Gateway has read all the available data off the network, it waits the **Active Timeout** period of time before closing the connection.

Idle Timeout:

The Enterprise Gateway supports HTTP 1.1 persistent connections. The **Idle Timeout** is the time that Enterprise Gateway waits after sending a message over a persistent connection to the Remote Host before it closes the connection. Typically, the Remote Host tells the Enterprise Gateway that it wants to use a persistent connection. The Enterprise Gateway acknowledges this, and keeps the connection open for a specified period of time after sending the message to the host. If the connection is not reused by within the **Idle Timeout** period, the Enterprise Gateway closes the connection.

Input Buffer Size:

The maximum amount of memory allocated to each request.

Output Buffer Size:

The maximum amount of memory allocated to each response.

Cache Addresses For:

The period of time to cache addressing information after it has been received from the naming service (for example, DNS).

SSL Session Cache Size:

Specifies the size of the SSL session cache for connections to the remote host. This controls the number of idle SSL sessions that can be kept in memory. You can use this setting to improve performance because it caches the slowest part of establishing the SSL connection. A new connection does not need to go through full authentication if it finds its target in the cache. Defaults to 32. If there are more than 32 simultaneous SSL sessions, this does not prevent another SSL connection from being established, but means that no more SSL sessions are cached. A cache size of 0 means the cache size is unlimited.

At **DEBUG** level or higher, the Enterprise Gateway outputs trace when an entry goes into the cache, for example:

```
DEBUG 09:09:12:953 [0d50] cache SSL session 11AA3894 to support.acme.com:443
```

If the cache is full, the output is as follows:

```
DEBUG 09:09:12:953 [0d50] enough cached SSL sessions 11AA3894 to support.acme.com:443
already
```

Configuring Watchdogs

You can configure an HTTP Interface to shut down based on certain *conditions*. One such condition is dependent on the Enterprise Gateway being able to contact a particular back-end Web Service running on a Remote Host. To do this, you can configure an **HTTP Watchdog** for a Remote Host to poll the endpoint. If the endpoint cannot be reached, the HTTP Interface is shut down.

To configure the Enterprise Gateway to shut down an HTTP Interface based on the availability of a Remote Host, perform the following steps:

1. Configure an **HTTP Watchdog** for the Remote Host.
2. Configure a **Requires Endpoint** condition on the HTTP Interface.
3. When configuring this condition, select the Remote Host configured in step 1 (the host with the associated Watchdog).

Note:

When **Load Balancing** is configured as **Weighted by response time**, and Remote Host Watchdogs are configured, the watch dog polling also contributes to the load balancing calculations.

For more information on adding a watchdog to a Remote Host, see [Configuring an HTTP Watchdog](#). For more information on adding Conditions to an HTTP Interface, see [Configuring Conditions for HTTP Interfaces](#).

Configuring an HTTP Watchdog

Overview

An HTTP Watchdog can be added to a Remote Host configuration in order to periodically poll the Remote Host to check its availability. The idea being that if the Remote Host becomes unavailable for some reason, a HTTP Interface can be brought down and will stop accepting requests. Once the Remote Host comes back online, the HTTP Interface will be automatically started up and will start accepting requests again.

To learn more about the reasons for shutting down an HTTP Interface if certain *conditions* do not hold, please refer to the help page on [Configuring Conditions for HTTP Interfaces](#).

To configure an HTTP Watchdog, right-click on a previously configured Remote Host in the tree view of the Policy Studio and select the **Watchdog -> Add** menu option. Configure the following sections on the **Configure HTTP Watchdog** dialog.

Configuration

Valid HTTP Response Code Ranges:

You can use this section to specify the HTTP response codes that you will regard as proof that the Remote Host is available. For example, if a 200 OK HTTP response is received for the poll request, the Remote Host can be considered available.

To specify a range of HTTP status codes, click the **Add** button and enter the **Start** and **End** of the range of HTTP response codes in the fields provided. An exact response code can be specified by entering the response code in both fields, e.g. "200".

HTTP Request for Polling:

The fields in this section allow you to configure the type and URI of the HTTP request to use to poll the Remote Host with. The default option is to use the *Options* HTTP command with a URI of "*", which is typically used to retrieve status information about the HTTP server.

If you wish to use an alternative HTTP request to poll the Remote Host, select an HTTP request method from the **Method** dropdown and then specify the URI to use in the **URI** field.

Remote Host Polling:

The settings in this section determine when and how the HTTP Watchdog polls the Remote Host. The **Poll Frequency** determines how often the Watchdog is to send the polling request to the Remote Host.

By default, the Watchdog uses "real" HTTP requests to the Remote Host to determine its availability. In other words, if the Enterprise Gateway is sending a batch of requests to the Remote Host it will use the response codes from these requests to decide whether or not the Remote Host is up. Therefore, the Watchdog effectively "polls" the Remote Host by sending real HTTP requests to it.

If you want to configure the Watchdog to send poll requests during periods when it is not sending requests to and receiving responses from the Remote Host, you should select the **Poll if up** checkbox. In this case the Watchdog will use "real" HTTP requests to poll the Remote Host as long as it is sending them, but will start sending "test" poll requests when it is not sending HTTP requests to the Remote Host in order to test its availability.

It is important to note that once a Remote Host is deemed to be down (i.e. an "invalid" HTTP response code was received) the Watchdog will continue to poll it at the configured **Poll Frequency** until it comes back up again (i.e. until a "valid" HTTP response code is received).

Configuring Conditions for HTTP Interfaces

Overview

In certain cases, it may be desirable to pull down the HTTP Interface that accepts traffic for the Enterprise Gateway. For example, if the back-end Web Service is unavailable or if the physical interface on the machine loses connectivity to the network, it is possible to shut down the HTTP Interface so that it stops accepting requests.

A typical scenario where this functionality proves useful is as follows:

- A load balancer sits in front of several running instances of the Enterprise Gateway and round-robins requests between them all.
- A client sends SSL requests through the load balancer, which forwards them opaquely to one of the Enterprise Gateway instances.
- The Enterprise Gateway terminates the SSL connection, processes the message with the configured policy, and forwards the request on to the back-end Web Service.

In this deployment scenario, the load balancer does not want to keep sending requests to an instance of the Enterprise Gateway if it has either lost connectivity to the network or if the back-end Web Service is unavailable. If either of these *conditions* hold, the load balancer should stop attempting to route requests through this instance of the Enterprise Gateway and use the other instances instead.

So then, how can the load balancer determine the availability of the Web Service and also the connectivity of the machine hosting the Enterprise Gateway to the network on which the Web Service resides? Given that the request from the client to the Enterprise Gateway is over SSL, the load balancer has no way of decrypting the encrypted SSL data to determine whether or not a SOAP Fault, for example, has been returned from the Enterprise Gateway to indicate a connection failure.

The solution is to configure certain *conditions* for each HTTP Interface, which must hold in order for the HTTP Interface to remain available and accept requests. If any of the associated conditions fail, the Interface will be brought down and will not accept any more requests until the failed condition becomes true and the HTTP Interface is restarted. Once the load balancer receives a connection failure from the Enterprise Gateway (which it will when the HTTP Interface is down) it will stop sending requests to this Enterprise Gateway and will choose to round-robin requests amongst the other instances instead.

The following conditions can be configured on the HTTP Interface:

- *Requires Endpoint:*
The HTTP Interface will remain up only if the Remote Host is available. The Remote Host is polled periodically to determine availability so that the HTTP Interface can be brought back up automatically when the Remote Host becomes available again.
- *Requires Link:*
The HTTP Interface will remain up only if a named physical interface has connectivity to the network. As soon as a "down" physical interface regains connectivity, the HTTP Interface will automatically come back up again.

Conditions can be configured for an HTTP Interface by right clicking on the HTTP Interface (e.g. "":8080") node under the Process node in the tree view of the Policy Studio. Select the **Add Condition** menu option and then either the **Requires Endpoint** or **Requires Link** option depending on your requirements. The sections below describe how to configure these conditions.

Requires Endpoint Condition

A **Requires Endpoint** Condition can be configured in cases where you only want to keep the HTTP Interface up if the

back end Web Service (i.e. the Remote Host) is available. An HTTP Watchdog can be configured for the Remote Host, which is then responsible for polling the Remote Host periodically to ensure that the Web Service is available. Take a look at the [Remote Hosts](#) and [Configuring HTTP Watchdogs](#) help pages for more information.

Remote Host:

The HTTP Interface will be shut down if the Remote Host selected here is deemed to be unavailable. The Remote Host can be continuously polled so that the Interface can be brought up again when the Remote Host becomes available again.

Requires Link Condition

The **Requires Link** Condition is used to bring down the HTTP Interface if a named physical network interface is no longer connected to the network. For example, if the cable is removed from the ethernet switch, the dependent HTTP Interface will be brought down immediately. The HTTP Interface will only start listening again once the physical interface is connected to the network again (i.e. when the ethernet cable is plugged back in).

Important Note:

The **Requires Link** Condition is only available on Linux and Solaris platforms.

Interface Name:

The HTTP Interface will be brought down if the physical network interface named here is no longer connected to the network. On Unix platforms, physical network interfaces are usually named "eth0", "eth1", and so on. On Solaris machines, interfaces are named according to the vendor of the network card, for example, "bge0", "bge1", etc.

POP Client

Overview

The POP Client allows you to poll a POP mail server and read email messages from it. Once the messages have been read, they can be passed into the core message pipeline where the full collection of message processing filters can act on them.

Configuration

The POP Client configuration is available by right-clicking on the Process node in the tree view of the Policy Studio and selecting the **POP Client -> Add** menu option. The following fields should then be configured on the **POP Mail Server** dialog:

Server Name:

Enter the hostname or IP address of the POP mail server.

Port:

Enter the port on which the POP server is listening. By default, POP servers listen on port 110.

User Name:

Enter the user name of a configured mail user for this POP server.

Password:

Enter the password for this user.

Poll Rate:

The Process will poll the mail server at the rate (in milliseconds) specified here.

Delete Message from Server:

Select this option if you want the POP server to delete email messages after they have been read by the Process.

Policy to Use:

Select the policy that you want to use to process messages that have been read from the POP server.

TIBCO Integration

Overview

The Enterprise Gateway ships with in-built support for TIBCO Enterprise Messaging System (EMS) and TIBCO Rendezvous enterprise-level products. The Enterprise Gateway can both produce and consume messages for both systems.

The remaining sections describe how to integrate with both TIBCO EMS and TIBCO Rendezvous. The reader is advised to follow the relevant links for more information on each of the steps involved.

TIBCO Rendezvous Integration

The Enterprise Gateway can act as both a producer and consumer of TIBCO Rendezvous messages. In both cases, a Rendezvous daemon must be configured, which is responsible for communicating with other Rendezvous programs on the network.

Producing TIBCO Rendezvous Messages:

The following steps must be configured in order to produce messages and send them to another Rendezvous program. Follow the links below to learn more about how to configure each of the steps involved.

1. [Configure the TIBCO Rendezvous Daemon](#)
2. [Configure the TIBCO Rendezvous Routing Filter](#)

Consuming TIBCO Rendezvous Messages:

A TIBCO Rendezvous Listener can be configured at the Process level in order to consume Rendezvous messages. The following steps must be followed in order to consume Rendezvous messages. Take a look at the help pages for the steps below for more complete information.

1. [Configure the TIBCO Rendezvous Daemon](#)
2. [Configure the TIBCO Rendezvous Listener](#)

TIBCO Enterprise Messaging System Integration

The Enterprise Gateway can also be configured to produce and consume messages for TIBCO Enterprise Messaging System (EMS) product. The first step in configuring either a producer or consumer of EMS messages is to configure a TIBCO Connection. The producer or consumer can then be configured.

Producing EMS Messages:

The following steps must be configured in order to send messages to an EMS Server. Follow the links below to learn more about how to configure each of the steps involved.

1. [Configure the TIBCO EMS Connection](#)
2. [Configure the TIBCO EMS Routing Filter](#)

Consuming TIBCO EMS Messages:

A TIBCO EMS Consumer can be configured at the Process level in order to consume messages from a queue or topic on an EMS Server. The following steps must be followed in order to consume EMS messages. Take a look at the help pages for the steps below for more complete information.

1. [Configure the TIBCO EMS Connection](#)
2. [Configure the TIBCO EMS Consumer](#)

Real-Time Monitoring Settings

Overview

The real-time monitoring settings are configured at the Process level. For example, these enable you to specify monitoring of the message path and message content. The Real-time Monitoring Console and Service Monitor use this data to display graphical reports in their web-based interfaces. In addition, for Service Monitor, you can also specify the database to which the Process writes metrics data.

Configuring Monitoring Settings

To configure real-time monitoring settings, right-click the Process (for example, the default **Oracle Enterprise Gateway**), and select **Monitoring** -> **Metrics**. The following fields are available on the **Real time monitoring** dialog.

Name:

Enter a descriptive name for these monitoring settings in this field.

Enable real time monitoring:

This setting enables real-time monitoring globally for the Enterprise Gateway. This checkbox must be selected to monitor traffic processed by the Enterprise Gateway, and is enabled by default. To disable real-time monitoring, unselect this checkbox.

Important Note:

Enabling real-time monitoring has a negative impact on performance. If you wish to maximize performance, do not enable this setting.

Configuring Service Monitor Settings

Store real time monitoring data for charts/reports:

If you wish real-time monitoring data to be written to a database, select this checkbox. This enables the following fields.

Use the following database:

If you have already configured the database that you wish to store the metrics data in, you can select it from this drop-down list. You can add databases as global configuration items under the **External Connections** node in the Policy Studio tree view. For more information on configuring database connections, see the [Database Connection](#) topic.

Note:

Service Monitor must be configured to read metrics data from the database that the Enterprise Gateway is configured to write the metrics data to. For more details, see the [Service Monitor Installation Guide](#).

Time window to store:

This setting determines the time period over which metrics are accumulated and enables you to view messages processed over this interval (for example, the number of messages processed over a 5 second interval). There are three time window sizes available to specify the granularity of the monitoring data: 5 seconds, 5 minutes, and 1 hour. When storing metrics into the database, metrics for a 5 second time window are stored by default. To store metrics for a longer time window size, select a longer value. It is important to note that storing metrics with shorter time window sizes can fill the database quickly and may cause reports rendered from the database to be slow.

The time window selected determines when reports are generated and viewed. Metrics for all time window sizes are always stored and made available to Service Monitor at the end of the time window interval. All time windows start at 0:00:00, and the count from there uses the specified window: hourly metrics at hour intervals, 5 minutes at 5 minute intervals, and 5 seconds at 5 second intervals. For example, if the Enterprise Gateway is configured to store hourly metrics, this data is not available to Service Monitor until the hour comes to an end.

Finally, it is important to note that if you have configured the Enterprise Gateway to store metrics using hourly time periods, you can not view metrics for 5 second or 5 minute time windows in Service Monitor. On the other hand, if you have

configured 5 second time periods, you can only view metrics for 5 second and hourly time windows in Service Monitor, but not 5 minutes.

Configuring Traffic Monitoring

Overview

The options on the **Traffic Monitor Settings** dialog enable you to configure the web-based Traffic Monitor tool and its message traffic log. For example, you can configure where the data is stored and what message transaction details are recorded in the log.

To access the **Traffic Monitor Settings** dialog, right-click the Enterprise Gateway process on the **Services** tab, and select **Monitoring** -> **Traffic**. You can access the Traffic Monitor tool on the **Enterprise Gateway Welcome Page**, available from <http://localhost:8090/index.html>. For details on how to use the Traffic Monitor tool, see the topic on [Monitoring Traffic](#).

Configuration

You can configure the following settings:

Enable Traffic Monitor:

Select whether to enable the web-based Traffic Monitor tool. This is enabled by default.

Data Directory:

Enter the directory that stores the traffic monitoring data files and database (`data.sdb`). Enter the location relative to the directory in which the Enterprise Gateway is installed. Defaults to `conf/opsdb.d`. If this directory and/or the database do not exist when the Enterprise Gateway starts up, they are recreated automatically.

Data Persistence Settings:

You can configure the following data persistence settings:

Record inbound transactions	Select whether to record inbound message transactions received by the Enterprise Gateway. This is enabled by default.
Record outbound transactions	Select whether to record outbound message transactions sent from the Enterprise Gateway to remote hosts. This is enabled by default.
Record circuit path	Select whether to record the circuit path for the message transaction, which shows the filters that the message passes through. This is enabled by default.
Record trace	Select whether to record the trace output for the message transaction. This is enabled by default.

Note:

These settings are global for all traffic passing through the Enterprise Gateway. You can override these persistence settings at the port level when configuring an HTTP or HTTPS interface. For more details, see [Configuring HTTP Services](#).

Data Expiry Settings:

You can configure the following data expiry settings:

Max. database entries	Enter the maximum number of rows stored in the traffic monitoring database. Defaults to 1000000 rows.
Max. database size	Enter the maximum size of the traffic monitoring database file, and select the database size units from the list. De-

	faults to 1 GB.
Purge data older than	Enter the period of time to store data, and select the time units from the list. Data older than the specified time is purged from the database. Defaults to 1 day.

Cryptographic Acceleration

Overview

The Enterprise Gateway uses OpenSSL to perform cryptographic operations, such as encryption and decryption, signature generation and validation, and SSL tunneling. OpenSSL exposes an *Engine API*, which makes it possible to plug in alternative implementations of some or all of the cryptographic operations implemented by OpenSSL. When configured appropriately, OpenSSL calls the engine's implementation of these operations instead of its own.

For example, a particular engine may provide improved implementations of the asymmetric operations RSA and DSA. This engine can then be plugged into OpenSSL so that whenever OpenSSL needs to perform either an RSA or DSA operation, it calls out to the engine's implementation of these algorithms rather than call its own.

Typically, OpenSSL engines provide a hardware implementation of specific cryptographic operations. The hardware implementation usually offers improved performance over its software-based counterpart, which is known as *cryptographic acceleration*.

Cryptographic acceleration can be configured at the process level in the Enterprise Gateway. To configure the Enterprise Gateway process to use an OpenSSL engine instead of the default OpenSSL implementation, right-click the process in the tree-view in **Policy Studio**, and select the **Cryptographic Acceleration -> Add OpenSSL Engine**.

General Configuration

The **OpenSSL Engine Configuration** dialog:

The dialog displays the name of the engine, the algorithms that it implements, together with any initialization and cleanup commands required by the engine. Complete the following fields:

Name:

Enter an appropriate name for the engine in this field.

Provides:

Enter a comma-separated list of cryptographic operations to be performed by the engine instead of OpenSSL. The engine must implement the listed operations, otherwise the default OpenSSL operations are used. The following operations are available:

<i>RSA</i>	RSA (Rivest Shamir Adleman) asymmetric algorithm
<i>DSA</i>	DSA (Digital Signature Algorithm) asymmetric algorithm
<i>RAND</i>	Random number generation
<i>DH</i>	Diffie-Hellman anonymous key exchange algorithm
<i>ALL</i>	Engine's implementation of all cryptographic algorithms

For example, if you want to configure the Enterprise Gateway to use the engine's implementation of the RSA, DSA, and DH algorithms only, enter the following in the **Provides** field:

RSA, DSA, DH

Commands:

The OpenSSL engine framework allows a number of control commands to be invoked at various stages in the loading and unloading of a specific engine library. These commands can be issued before and/or after the initialization of the engine, and also before and/or after the engine is un-initialized. Control commands are based on text name-value pairs.

Typical uses for control commands include specifying the path to a driver library, logging configuration information, a password to access a protected devices, a configuration file required by the engine, and so on.

OpenSSL control commands can be added by clicking the **Add** button. The **OpenSSL Engine Command**:

Enter the name of the command in the **Name** field, and its value in the **Value** field. This command *must* be supported by the engine.

Use the **When** drop-down list to select when the command is to be run. The options available are as follows:

<i>preInit</i>	Command is run before the engine is initialized (before the call to <code>ENGINE_init()</code>).
<i>postInit</i>	Command is run after the engine is initialized (after the call to <code>ENGINE_init()</code>).
<i>preShutdown</i>	Command is run before the engine shuts down (before the call to <code>ENGINE_finish()</code>).
<i>postShutdown</i>	Command is run after the engine shuts down (after the call to <code>ENGINE_finish()</code>).

Conversations for Crypto Engines

A Hardware Security Module (HSM) protects the private keys that it holds using a variety of mechanisms, including physical tokens, passphrases, and other methods. When use of the private key is required by an agent, it must authenticate itself with the HSM, and be authorized to access this data.

For information on how the Enterprise Gateway interacts with the HSM, see the [Cryptographic Acceleration - Conversation](#) topic.

Cryptographic Acceleration - Conversation - Request:Response

Conversations for Crypto Engines

HSMs protect the private keys they hold using a variety of mechanisms, including physical tokens, passphrases, and other methods. When use of the private key is required by some agent, they need to authenticate themselves with the HSM and be authorized to access this data.

Whatever the mechanism protecting the keys on the HSM, this will commonly require some interaction with the agent. The most common form of interaction required is for the agent to present a passphrase. The intent is generally that this is carried out by a real person, rather than produced mechanically by the agent. Other forms of interaction may include prompting the operator to insert a specific card into a card reader, for example.

The requirement for an operator to enter a passphrase renders automated startup of services using the HSM impossible, however. Although weaker from a security standpoint, the server is able to conduct an automated dialog with a HSM when it requires access to a private key, presenting specific responses to specific requests, including feeding passphrases to it. (This will, of course, be futile if the dialog calls for the insertion of a physical token in a device.)

The dialogue for different keys on the same device will often be the same. For example, a number of keys on an nCipher HSM may require the server to present an operator passphrase for a pre-inserted card in a card-reader. The specific dialogues are therefore associated with the cryptographic engine.

Each dialogue consists of a set of expected request/generated response pairs. The "expected request" takes the form of a regular expression. When the cryptographic device prompts for input, the text of this prompt is compared against each expected request in the conversation, until a match is found. Once matched, the corresponding "generated response" is delivered to the HSM.

In the simplest case, consider a HSM producing the following prompt

```
Enter passphrase for operator card Operator1:
```

We can identify this, for example with the following regex:
"passphrase.*Operator1"

In the configured conversation, we can make the expected response to this prompt the passphrase for the specific card, for example:
"tellNoOne"

The server is somewhat at the mercy of the HSM for how this dialog continues: if the HSM continues to prompt for requests, the server can only attempt to respond. You may set the "maximum expected challenges" setting on the conversation to indicate a maximum number of prompts to expect from the HSM, at which point the server will do its best to terminate the conversation, almost certainly failing to load the affected key.

TIBCO Rendezvous Daemon

Overview

TIBCO Rendezvous® is the leading low latency messaging product for real-time high throughput data distribution applications. A message can be sent from the TIBCO daemon running on the local machine to a single TIBCO daemon running on a separate host machine or it can be broadcast to several daemons running on multiple machines. Each message has a subject associated with it, which acts as the *destination* of the message.

A listener, which is itself a TIBCO daemon, can declare an interest in a subject on a specific daemon. Whenever a message is delivered to this subject on the daemon the message is delivered to the listening daemon.

The Enterprise Gateway can act as a listener on a specific subject at a TIBCO daemon, in which case it said to be acting as a *consumer* of TIBCO messages. Similarly, it can also send messages to a TIBCO daemon, effectively acting as a *producer* of messages. In both cases, the local TIBCO daemon must be configured to talk to the TIBCO daemons running on the remote machines.

For more information on consuming and producing messages to and from TIBCO Rendezvous, please refer to the following help pages:

- [TIBCO Integration](#)
- [TIBCO Rendezvous Listener](#)
- [TIBCO Rendezvous Router](#)

The remainder of this page describes how to configure a TIBCO Rendezvous Daemon. For a more detailed description of how to configure the fields on this dialog please refer to your TIBCO Rendezvous documentation.

Configuration

You can configure **TIBCO Rendezvous Daemons** on the **External Connections** tab in the Policy Studio. Right-click the **TIBCO Rendezvous Daemons node**, and select **Add a TIBCO Rendezvous Daemon**. Configure the following fields on the **TIBCO Daemon Settings** dialog:

Name:

Enter a friendly name for this TIBCO Rendezvous Daemon. When configured, this name is available for selection when configuring a **TIBCO Rendezvous Listener** and a **TIBCO Rendezvous Connection** filter.

Service:

Communication between TIBCO Rendezvous daemons takes place using Pragmatic General Multicast (PGM) or Universal Datagram Protocol (UDP) services. The specified *service parameter* configures the local TIBCO Rendezvous daemon to use this type of service when sending or broadcasting messages to other TIBCO Rendezvous daemons who are also using this service.

You can specify the service in the following ways:

- **By Service Name:**
If your network administrator has added an entry for TIBCO Rendezvous in a network database such as NIS (for example, `rendezvous 7500/udp`), you can enter the name of the service (for example, `rendezvous`) in this field.
- **By Port Number:**
Alternatively, you can enter the port number on which the TIBCO Rendezvous daemon is listening (for example, `7500`).
- **Default Option:**
If you leave this field blank, a default service name of `rendezvous` is assumed. For this reason, administrators should add an entry in the network database with this name (for example, `rendezvous 7500/udp`). This enables

you to leave this field blank so that this default service is used.

Network:

If the machine on which the TIBCO Rendezvous daemon is running has more than one network interface, you can specify what interface to use for all communications with other daemons. Each TIBCO Rendezvous daemon can only communicate on a single network, meaning that separate daemons must be configured for each network you want the daemon to communicate on.

For simplicity, you can leave this field blank, in which case the primary network interface is used for communication with other daemons. For more information on how to configure different networks and multicast groups, please see the TIBCO Rendezvous documentation.

Daemon:

The value entered here tells the Enterprise Gateway where it can find the TIBCO Rendezvous daemon, which is responsible for communicating with all other daemons on the network. This daemon can be local or remote.

For local daemons you need only specify the port number that the daemon is running on (for example 6500). Alternatively, you can leave this field blank to connect to the daemon on the default port.

To connect to a remote daemon, you must specify both the host and port number of the daemon in this field (for example `daemon_host:6500`).

TIBCO Rendezvous Listener

Overview

TIBCO Rendezvous® is the leading low latency messaging product for real-time high throughput data distribution applications. A message can be sent from the TIBCO daemon running on the local machine to a single TIBCO daemon running on a separate host machine or it can be broadcast to several daemons running on multiple machines. Each message has a subject associated with it, which acts as the *destination* of the message.

A listener, which is itself a TIBCO daemon, can declare an interest in a subject on a specific daemon. Whenever a message is delivered to this subject on the daemon the message will be delivered to the listening daemon.

The Enterprise Gateway can act as a listener on a specific subject at a TIBCO daemon, in which case it said to be acting as a *consumer* of TIBCO messages. Similarly it can also send messages to a TIBCO daemon, effectively acting as a *producer* of messages. For more information on how to send messages to other TIBCO Rendezvous programs, see the [TIBCO Rendezvous Routing](#) filter

Configuration

A TIBCO Rendezvous Listener is configured at the Process level in the Policy Studio. To add a listener, right-click the Oracle Enterprise Gateway Process in the tree view of the Policy Studio. Select the **TIBCO -> Rendezvous Listener -> Add** option from the context menu. Configure the following fields on the **TIBCO Rendezvous Listener** dialog:

TIBCO Settings Tab:

Enter the name of the subject that you want this consumer to listen for in the **Subject** field. Only messages addressed with this subject are consumed by the listener.

Select the TIBCO Rendezvous Daemon to use to communicate with other TIBCO programs. You can configure TIBCO Rendezvous Daemons globally on the **External Connections** tab in the Policy Studio. For more information, please see the [TIBCO Rendezvous Daemon](#) topic.

Policy to Use:

When messages with the specified subject have been consumed they must be passed into a policy where they can be processed accordingly. Select the policy that you want to use to process consumed messages from the tree.

TIBCO Enterprise Messaging System Consumer

Overview

TIBCO Enterprise Messaging System™ (EMS) provides a distributed message bus with native support for JMS (Java Messaging Service) and TIBCO Rendezvous, along with other protocols.

In general, TIBCO EMS clients *produce* messages and send them to the TIBCO EMS Server. Similarly, TIBCO EMS clients can connect to the TIBCO EMS Server and declare an interest in a particular queue or topic on that server. In doing so, it can *consume* messages that have been produced by another TIBCO EMS client.

The Enterprise Gateway can act as a message producer by sending messages to the TIBCO EMS Server and as a message consumer by listening on a queue or topic at the server. Both configurations require a connection to the TIBCO EMS Server. For more information on consuming and producing messages to and from TIBCO EMS, please refer to the following pages:

Configuration

TIBCO EMS Consumers are added at the Process level in the Policy Studio. To add a consumer, right-click the "Oracle Enterprise Gateway" node under "Processes" in the tree view of the Policy Studio. Select the **TIBCO -> Enterprise Messaging Consumer Service -> Add** options from the context menus. The following tabs and fields should be configured on the **TIBCO Enterprise Messaging Service Consumer** dialog.

Connection Tab:

Select a previously configured TIBCO EMS Connection for this consumer to connect to. TIBCO EMS Connections are configured globally under the "External Connection" node in the tree view of the Policy Studio. Take a look at the [TIBCO EMS Connection](#) page for more information.

Settings Tab:

Configure the following fields on the **Settings** tab:

Destination Type:

Select whether this consumer will read messages off a queue or topic.

Queue/Topic Name:

Enter the name of the queue or topic here.

Selector:

Enter a filter to restrict the messages that are read off the queue or topic.

Do Not Receive Local Messages:

Check this option if you do not want to consume messages that have been produced by the Enterprise Gateway. For example, if you have configured a TIBCO EMS Routing filter to place messages on to a queue and have also configured a TIBCO EMS Consumer to read messages from the same queue, you can check this option to ensure that the consumer will ignore these locally generated messages.

Extraction Method:

The option selected here determines how the Enterprise Gateway will serialize the JMS message consumed from the queue or topic so that it can be passed into the policy selected on the **Policy** tab. The following options are available:

- *Create a content.body attribute based on the SOAP over JMS draft specification:*
If this option is selected, messages are formatted according to the [SOAP over JMS](http://www.w3.org/TR/soapjms/) [http://www.w3.org/TR/soapjms/] recommendation, and stored in the `content.body` message attribute.
- *Insert the JMS message directly into the attribute named below:*
Select this option to simply store the JMS message directly into the attribute specified in the **Attribute Name** field

below.

- *Populate the attribute below with the value inferred from message type to Java:*
Select this option if you wish to infer the data type of the JMS message from the underlying TIBCO EMS data type. In this case a TIBCO EMS TextMessage, BytesMessage, and MapMessage, will be converted into a java.lang.String, a byte[], and a java.lang.Map, respectively, while a JMS ObjectMessage will be deserialized into the attribute specified in the **Attribute Name** field below.

Attribute Name:

Once the message has been consumed it will be stored in the Oracle message attribute specified here. The **Extraction Method** selected above will determine how the raw JMS message is deserialized to the specified attribute. The consumed message can be processed at any stage hereafter in the policy selected on the **Policy** tab by accessing this attribute. By default the message is stored in the `ems.message` attribute.

Policy Tab:

Select a previously configured policy that you want to pass messages to after consuming the messages from the queue or topic configured on the **Settings** tab.

Oracle Security Service Module Settings

Overview

An Oracle Security Service Module (SSM) integrates a secured application (in this case, the Enterprise Gateway) with an Oracle Entitlements Server (OES) so that security administration (for example, roles, resources, and policies) is delegated to the Oracle Entitlements Server. An SSM must be installed on the machine hosting the application to be secured by the Oracle Entitlements Server. The SSM runs in-process with the secured application, which improves performance and on-the-wire security.

In the Policy Studio, on the **Services** tab, you can right-click the Enterprise Gateway process, and select **Oracle -> Security Service Module (SSM) Settings**. The **Oracle Security Service Module Settings** dialog enables you to configure the Enterprise Gateway to act as a Java SSM. For more details Oracle Entitlements Server and SSMs, see the [Oracle Entitlements Server](http://www.oracle.com/technetwork/middleware/oes/overview/index.html) [http://www.oracle.com/technetwork/middleware/oes/overview/index.html] website.

Note:

Oracle Entitlements Server was previously known as BEA AquaLogic Enterprise Security (ALES). Some items, such as schema objects, paths, and so on, may still use the ALES name.

Settings

Configure the following fields on the **Settings** tab:

Enable Oracle Security Service Module:

Select whether to enable the Enterprise Gateway process to act as an SSM. This setting is disabled by default.

Application Configuration Name:

Enter the Application Configuration name for the SSM instance. This is the name of your runtime application used by OES (for example, for monitoring purposes).

Configuration Name:

Enter the OES Configuration name for the SSM instance to be stored in the OES Configuration Repository. Configuration names share the same name as their Policy Domain names.

Application Configuration Properties:

Click **Add** to specify optional configuration properties as name-value pairs. Enter a **Name** and **Value** in the **Properties** dialog. Repeat to specify multiple properties.

Policy Domain Name:

Enter the OES Policy Domain name for the SSM instance. Policy Domains contain policy definitions (target resource, permission set, and policy). Policy Domain names share the same name as their Configuration names.

Name Authority Definition

Configure the following field on the **Name Authority Definition** tab:

Name Authority Definition File:

Click the **Browse** button at the bottom right to configure the Name Authority Definition file for the SSM. This is an XML file that specifies the naming authority definition required for the Enterprise Gateway. For example, a specified XML file named `gatewayNameAuthorityDefinition.xml` file should contain the following settings:

```
<AuthorityConfig>
  <AuthorityDefinition name="gatewayResource" delimiters="/\">
    <Attribute name="protocol" type="MULTI_TOKEN" authority="URLBASE" />
  </AuthorityDefinition>
```

```
<AuthorityDefinition name="gatewayAction" delimiters="/">
  <Attribute name="action" type="SINGLE_VALUE_TERMINAL"/>
</AuthorityDefinition>
</AuthorityConfig>
```

Further Information

When you have configured the settings in the **Oracle Security Service Module Settings** dialog, you can use the following filters to integrate the Enterprise Gateway with Oracle Entitlements Server:

- [Oracle Entitlements Server Authorization](#)
- [Get Roles from Oracle Entitlements Server](#)

Certificate Store

Overview

For the Enterprise Gateway to trust X.509 certificates issued by a specific Certificate Authority (CA), you must import that CA's certificate into the Enterprise Gateway's Trusted Certificate Store. For example, if the Enterprise Gateway is to trust secure communications (SSL connections or XML Signature) from an external SAML Policy Decision Point (PDP), you must import the PDP's certificate or the issuing CA's certificate into the Enterprise Gateway.

Configuring a Certificate and Private Key

To view the list of certificates stored in the Certificate Store, click the **Certificates** tab on the left of the Policy Studio. The certificates are listed in a table in the main panel of the Policy Studio.

To create a certificate and private key, click the **Create/Import** button on the **Certificates** screen in the **Policy Studio**. The **Configure Certificate and Private Key** dialog is displayed. The next sections explain how to use this dialog.

X.509 Certificate

The following configuration options are available on the **X.509 Certificate** tab:

- **Subject:**
Click the **Edit** button to configure the *Distinguished Name* (DName) of the subject.
- **Public Key:**
Click the **Import** button to import the subject's public key (usually from a PEM or DER-encoded file).
- **Version:**
This read-only field displays the X.509 version of the certificate.
- **Issuer:**
This read-only field displays the distinguished name of the CA that issued the certificate.
- **Validity Period:**
The dates specified here define the validity period of the certificate.
- **Alias Name:**
This mandatory field enables you specify a friendly name (or alias) for the certificate.
- **Use Distinguished Name:**
Select this option to view the DName of the certificate in the text box instead of the certificate alias.
- **Import Certificate:**
Click this button to import a certificate from a file.
- **Export Certificate:**
Use this option to export the certificate to a file.
- **Sign Certificate:**
Click this button to sign the certificate. The certificate can either be self-signed, or it can be signed by the private key belonging to a trusted CA whose key pair has been stored in the **Certificate Store**.

Private Key

Use the **Private Key** tab to configure details of the private key. By default, private keys are stored locally in the Certificate Store. They can also be stored on a Hardware Security Module (HSM), if required.

Private Key Stored Locally:

Select the **Private key stored locally** radio button. The following configuration options are available for keys that are stored locally in the Certificate Store:

- **Private Key:**

This read-only field displays details of the private key.

- **Import Private Key:**
Click the **Import Private Key** button to import the subject's private key (usually from a PEM or DER-encoded file).
- **Export Private Key:**
Click this button to export the subject's private key to a PEM or DER-encoded file.

Private key stored on HSM:

If the private key that corresponds to the public key stored in the certificate resides on a HSM, select the **Private key stored on HSM** radio button. Configure the following fields to associate a key stored on a HSM with the current certificate:

- **Engine Name:**
Enter the name of the OpenSSL Engine to use to interface to the HSM. All vendor implementations of the OpenSSL Engine API are identified by a unique name. Please refer to your vendor's HSM or OpenSSL Engine implementation documentation to find out the name of the engine.
- **Key ID:**
The value entered is used to uniquely identify a specific private key from all others that may be stored on the HSM. On completion of the dialog, this private key will be associated with the certificate that you are currently editing.

Global Options

The following global configuration options apply to both the **X.509 Certificate** and **Private Key** tabs:

- **Import Certificate + Key:**
Use this option to import a certificate and a key from a file.
- **Export Certificate + Key:**
Use this option to export a certificate and a key to a file.

Click **OK** when you have finished configuring the certificate and/or private key.

Managing certificates

On the main **Certificates** screen, you can edit an existing certificate using the **Edit** button. You can also view the details of an existing certificate using the **View** button. Similarly, you can remove a certificate from the Certificate Store using the **Remove** button.

You can also export a certificate to a Java keystore. You can do this by selecting the certificate in the table, and then clicking the **Export to Keystore** button. Choose the name and location of the new keystore file, and enter a passphrase for this keystore when prompted.

Similarly, you can import certificates and keys from a Java keystore into the Certificate Store. To do this, click the **Keystore** button on the main Certificates screen. On the **Keystore** screen, browse to the location of the keystore by clicking the button beside the **Keystore** field.

The certificates/keys in the keystore are listed in the table. To import any of these keys to the Certificate Store, select the box next to the certificate or key that you want to import, and then click the **Import to Trusted Certificate Store** button. If the key is protected by a password, you are prompted for this password.

You can also use the **Keystore** screen to view and remove existing entries in the keystore. You can also add keys to the keystore and to create a new keystore. Use the appropriate button to perform any of these tasks.

Enterprise Gateway Users

Overview

The Enterprise Gateway User Store contains the configuration data for managing Enterprise Gateway user information. This topic introduces the concepts of Enterprise Gateway Users, Groups, and Attributes. It explains how to manage these components on the **Users** tab in the Policy Studio.

Users

Enterprise Gateway Users specify the user identity in the User Store. This includes details such as the user name, password, and X.509 certificate. Enterprise Gateway Users must be a member of at least one User Group. In addition, Users can specify optional Attributes, and inherit Attributes at the Group level.

To view all existing Users, click the **Users** tab on the bottom left of the Policy Studio, and select the **Users** tree node. The Users are listed in the table on the main panel of the Policy Studio. You can find a specific User by entering a search string in the **Filter** field.

Adding Users

You can create Enterprise Gateway Users on the **Users** page in the Policy Studio. Click the **Add** button on the right to view the **Add User** dialog.

Adding User Details

To specify the new user details, complete the following fields on the **General** tab:

- **User Name**
Enter a name for the new user.
- **Password**
Enter a password for the new user.
- **Confirm Password**
Re-enter the user's password to confirm.
- **X.509 Cert**
Click the **X.509 Cert** button to load the user's certificate from the **Certificate Store**.

Adding User Attributes

You can specify optional User Attributes on the **Attributes** tab, which is explained in the next section.

Attributes

You can specify Attributes at the User level and at the Group level on the **Attributes** tab. Attributes specify user configuration data (for example, attributes used to generate SAML attribute assertions).

Adding Attributes

The **Attributes** tab enables you to configure user attributes as simple name-value pairs. The following are examples of user attributes:

- `role=admin`
- `email=niall@oracle.com`
- `dept=eng`
- `company=oracle`

You can add user attributes by clicking the **Add** button. Enter the attribute name, type, and value in the fields provided. The `Encrypted` type refers to a string value that is encrypted using a well-known encryption algorithm or cipher.

Groups

Enterprise Gateway User Groups are containers that encapsulate one or more Users. You can specify Attributes at the Group level, which are inherited by all Group members. If a User is a member of more than one Group, that User inherits Attributes from all Groups (the superset of Attributes across the Groups of which the User is a member).

To view all existing Groups, click the **Users** tab on the bottom left of the Policy Studio, and select the **Groups** tree node. The User Groups are listed in the table on the main panel of the Policy Studio. You can find a specific Group by entering a search string in the **Filter** field.

Adding Groups

You can create User Groups on the **Groups** page in the Policy Studio. Click the **Add** button on the right to view the **Add Group** dialog.

Adding Group Details

To specify the new group details, complete the following fields on the **General** tab:

- **Group Name**
Enter a name for the new group.
- **Members**
Click the **Add** button to display the **Add Group Member** dialog, and select the members to add to the group.

Adding Group Attributes

You can specify optional Attributes at the Group level on the **Attributes** tab. For more details, see the [Attributes](#) section.

Updating Users or Groups

To edit details for a specific User or Group, select it in the list, and click the **Edit** button on the right. Enter the updated details in the **Edit User** or **Edit Group** dialog.

To delete a specific User or Group, select it in the list, and click the **Remove** button on the right. Alternatively, to delete all Users or Groups, click the **Remove All** button. You are prompted to confirm all deletions.

System Alerting

Overview

This tutorial shows the Enterprise Gateway's enhanced logging capabilities. System alerts are usually sent when a filter fails, but they can also be used for notification purposes. The Enterprise Gateway can send system alerts to several alert destinations, including a Windows Event Log, UNIX/Linux syslog, NMP Network Management System, Check Point Firewall-1, email recipient, or Twitter.

There are two main steps involved in configuring the Enterprise Gateway to send system alerts:

1. Configuring an alert destination
2. Configuring an **Alert** filter

Configuring an Alert Destination

The first step in configuring the Enterprise Gateway to send alerts is to configure an *alert destination*. The Enterprise Gateway can send alerts to the following destinations:

- [UNIX/Linux Syslog](#)
- [Windows Event Log](#)
- [Check Point FireWall-1](#)
- [SNMP Network Management System](#)
- [Email Recipient](#)
- [Twitter](#)

You can configure these alert destinations using the **External Connections** tab on the left of the Policy Studio.

UNIX/Linux Syslog

Many types of UNIX and Linux provide a general purpose logging utility called *syslog*. Both local and remote processes can send logging messages to a centralized system logging daemon, known as *syslogd*, which in turn writes the messages to the appropriate log files. You can configure the level of detail at which *syslog* logs information. This enables administrators to centrally manage how log files are handled, rather than separately configuring logging for each process.

Each type of process logs to a different syslog *facility*. There are facilities for the kernel, user processes, authorization processes, daemons, and a number of place-holders that can be used by site-specific processes. For example, the Enterprise Gateway enables you to log to facilities such as *auth*, *daemon*, *ftp*, *local0-7*, and *syslog* itself.

remote syslog

To configure a remote syslog alert destination, perform the following steps:

1. Right-click **Alerts** on the left in the **External Connections** tree, and select **Add -> Syslog Remote**.
2. The **Syslog Server** dialog enables you to specify details about the machine on which the syslog daemon is running. The Enterprise Gateway connects to this daemon and logs to the specified facility when the alert event is triggered. Complete the following fields on the **Syslog Server** dialog:
 - **Name:**
Enter a name for this alert destination.
 - **Host:**
Enter the host name or IP address of the machine where the syslog daemon is running.
 - **Facility:**
Select the facility that the Enterprise Gateway sends alerts to from the drop-down list.

3. Click **OK**.

local syslog (UNIX only)

To configure a local syslog alert destination, perform the following steps:

1. Right-click **Alerts** on the left in the **External Connections** tree, and select **Add -> Syslog Local (UNIX only)**.
2. The **Syslog Server** dialog enables you to specify where the alert is sent when the alert event is triggered. Complete the following fields on the **Syslog Server** dialog:
 - **Name:**
Enter a name for this alert destination.
 - **Facility:**
Select the facility that the Enterprise Gateway sends alerts to from the drop-down list.
3. Click **OK**.

Windows Event Log

This alert destination enables alert messages to be written to the local or a remote Windows Event Log. To add a Windows Event Log alert destination, perform the following steps:

1. Right-click **Alerts** on the left in the **External Connections** tree, and select **Add -> Windows Event Log**.
2. The **Windows Event Log Alerting** dialog enables you to specify the machine of the Event Log the Enterprise Gateway sends alerts to. Complete the following fields on this dialog:
 - **Name:**
Enter a name for this alert destination.
 - **UNC Server name:**
Enter the UNC (Universal Naming Code) of the machine where the event log resides. For example, to send alerts to the event log running on a machine called `\\NT_SERVER`, enter `\\NT_SERVER` as the UNC name for this host.
3. Click **OK**.

Check Point FireWall-1

The Enterprise Gateway complies with OPSEC (Open Platform for Security). OPSEC compliance is awarded by Check Point Software Technologies to products that have been successfully integrated with at least one of their products. In this case, the Enterprise Gateway has been integrated with the Check Point FireWall-1 product.

FireWall-1 is the industry leading firewall that provides network security based on a security policy created by an administrator. Although OPSEC is not an open standard, the platform is recognized worldwide as the standard for interoperability of network security, and the alliance contains over 300 different companies. OPSEC integration is achieved through a number of published APIs, which enable third-party vendors to interoperate with Check Point products.

To configure a FireWall-1 alert destination, perform the following steps:

1. Right-click **Alerts** on the left in the **External Connections** tree, and select **Add -> OPSEC**.
2. The **OPSEC Alerting** dialog enables you to specify details about the machine on which FireWall-1 is installed, the port it is listening on, and how to authenticate to the firewall. The Enterprise Gateway connects to the specified firewall when the alert event is triggered and prevents further requests for the particular client that triggered the alert. The following configuration settings must be set:
 - **sam_server_auth_port:**
The port number used to establish SIC (Secure Internal Communications) based connections with the firewall.
 - **sam_server_auth_type:**
The authentication method used to connect to the firewall.
 - **sam_server_ip:**
The host name or IP address of the machine that hosts the Check Point Firewall.
 - **sam_server_opsec_entity_sic_name:**

- The firewall's SIC name.
 - **opsec_sic_name:**
The OPSEC application's SIC Name, which is the application's full DName as defined by the VPN-1 SmartCenter Server.
 - **opsec_sslca_file:**
The name of the file containing the OPSEC application's digital certificate.
3. Click **OK**.

You can store configuration information in a file and then load it using the **Browse** button. Alternatively, you can use the **Template** button to load the required settings into the text area, and add the configuration values manually.

For the Enterprise Gateway to establish the SSL connection to the firewall, the `opsec_sslca_file` specified must be uploaded to the Enterprise Gateway machine. You can do this by clicking the **Add** button at the bottom of the screen.

For more information on OPSEC settings, see the documentation for your OPSEC application.

SNMP Network Management System

This alert destination enables the Enterprise Gateway to send SNMP (Simple Network Management Protocol) traps to an NMS (Network Management System).

To configure an SNMP alert destination, perform the following steps:

1. Right-click **Alerts** on the left in the **External Connections** tree, and select **Add -> SNMP**.
2. The **SNMP Alerting** dialog enables you to specify details about the NMS that the Enterprise Gateway should send an alert to. Complete the following fields:
 - **Host:**
The host name or IP address of the machine on which the NMS system resides.
 - **Port:**
The port on which the NMS system is listening.
 - **Timeout:**
The timeout in seconds for connections from the Enterprise Gateway to the NMS system.
 - **Retries:**
The number of retries that should be attempted whenever a connection failure occurs.
 - **SNMP Version:**
Select the version of SNMP that you wish to use for this alert.
3. Click **OK**.

Email Recipient

This alert destination enables alert messages to be sent by email. To add a Windows Event Log alert destination, perform the following steps:

1. Right-click **Alerts** on the left in the **External Connections** tree, and select **Add -> Email**.
2. The **Email Alerting** dialog enables you to configure how the email alert is to be sent. Complete the following fields:
 - **Name:**
Enter a name for this alert destination.
 - **Email Recipient (To):**
Enter the recipient of the alert mail in this field. Use a semicolon-separated list of email addresses to send alerts to multiple recipients.
 - **Email Sender (From):**
Email alerts appear *from* the sender email address specified here. It is important to note that some mail servers do not allow relaying mail when the sender in the From field is not recognized by the server.
3. In the **SMTP Server Settings**, specify the following fields:
 - **Outgoing Mail Server (SMTP):**

- Specify the SMTP server that the Enterprise Gateway uses to relay the alert email.
- **Port:**
Specify the SMTP server port to connect to. Defaults to port 25.
- **Connection Security:**
Select the connection security used to send the alert email (SSL, TLS, or NONE). Defaults to NONE.
- 4. If you are required to authenticate to the SMTP server, specify the following fields in **Log on Using**:
 - **User Name:**
Enter the user name for authentication.
 - **Password:**
Enter the password for the user name specified.
- 5. Finally, you can select the **Email Debugging** checkbox to find out more information about errors encountered by the Enterprise Gateway when attempting to send email alerts. All trace files are written to the `/trace` directory of your the Enterprise Gateway installation.
- 6. Click **OK**.

Twitter

This alert destination enables the Enterprise Gateway to send tweet alerts to Twitter. Twitter uses the OAuth open authentication standard. To enable the Enterprise Gateway to send tweet alerts using the Twitter API, you first need to do the following:

- Create a Twitter account to represent you as the user
- Register a custom application for your Enterprise Gateway instance, which posts alerts on the user's behalf

Twitter requires that API calls are made for both the user and the application. The Twitter API requires the following credentials:

- Consumer Key of registered applications
- Consumer Secret Key of registered application
- Access Token allowing application to post on behalf of a user
- Access Token Secret to verify the Access Token

Twitter uses this information to determine which application is calling the API, and verifies that the Twitter user you are attempting to make API requests on behalf of has authorized access to their account using the specified application. Twitter identifies and authenticates all requests as coming from both the user performing the request and the registered Enterprise Gateway application working on the user's behalf.

Registering a client application

To use the Twitter API, you must create a Twitter account, and register a client application for the Enterprise Gateway. If you have not already created a Twitter account, register a new account using the instructions on <http://www.twitter.com>. When you have created the account, register a client application for the Enterprise Gateway as follows:

1. Go to <http://dev.twitter.com/>.
2. On the **Twitter** toolbar, select **Your apps**.
3. Click the **Register a new app** button.
4. Enter the details for your custom application. Some details are arbitrary, but you must specify the following values:
 - **Application Type:**
Select the **Client** radio button.
 - **Default Access Type:**
Select the **Read & Write** radio button.

Note: The **Application Name** may already be registered to another user, so you may need to specify a different

unique name.

5. Click **Register Application**. Each client application you register is provisioned a consumer key and consumer secret. These are used, in conjunction with the OAuth library, to sign every request you make to the API. Using this signing process, Twitter trusts that the traffic identifying itself as you is indeed you.
6. Select your registered application, and select **My Access Token**. This provides you with an access token and an access token secret. You must store these safely.

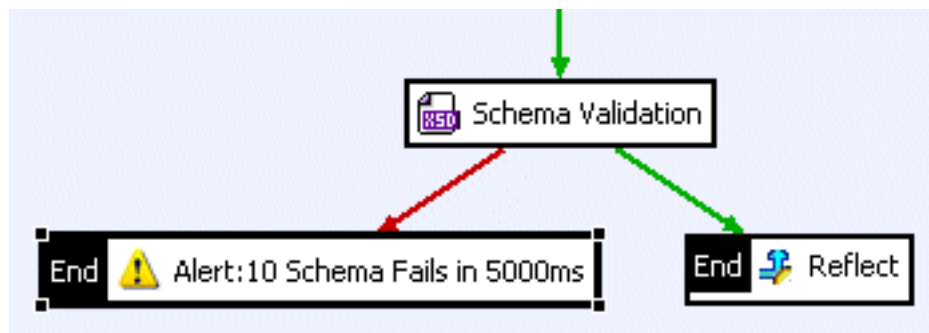
Configuring a Twitter alert destination

To configure a Twitter alert destination, perform the following steps:

1. Right-click **Alerts** on the left in the **External Connections** tree.
2. Select **Add -> Twitter**.
3. The **Twitter Alerting** dialog enables you to specify credentials for the Twitter user that the Enterprise Gateway uses to send an alert to. Complete the following fields on this dialog:
 - **Consumer Key:**
The Consumer Key of your registered application.
 - **Consumer Secret:**
The Consumer Secret of your registered application.
 - **Access Token:**
The Access Token that represents you.
 - **Access Token Secret:**
The Access Token Secret that represents you.

Configuring an Alert Filter

Typically, an **Alert** filter is placed on the failure path of another filter in the policy. For example, you could configure an alert if a schema validation fails 10 times within a 5000 millisecond period for a specified Web Service. In this case, you would place the **Alert filter** on the failure path from the **Schema Validation** filter, as shown in the following policy screen shot:



When editing policies, you can drag and drop the **Alert** filter from the **Monitoring** filter group. To configure an alert filter, specify the following settings on the **Alert filter** screen:

Name:

Enter a descriptive name for the filter.

Message:

Configure the following settings on the **Message** tab:

Alert Type	Select the severity level of the alert. This option is only relevant for alert destinations that support severity levels, such as the Windows Event Log.
-------------------	--

Alert Message	<p>Enter the text that appears in the alert. You can enter message attributes using properties, which are looked up and expanded to values at runtime. For example, instead of sending a generic alert stating Authentication Failed, you can use a message attribute to give the ID of the user that was not authenticated. To do this, use the following syntax:</p> <pre>\${name_of_attribute}</pre> <p>The following examples show how to use message attributes in alert messages:</p> <ul style="list-style-type: none"> • Authentication failure for user: <code>\${authentication.subject.id}</code>. • <code>{alert.number.failures}</code> authentication failures have occurred in <code>\${alert.time.period}</code> seconds. • <code>\${alert.number.failures}</code> exceptions have occurred in circuit <code>\${circuit.name}</code>. • The last exception was <code>\${circuit.exception}</code> with path <code>\${circuit.path}</code>. <p>Note that an alert message is not required for alerts sent to an OPSEC firewall.</p>
----------------------	--

Tracking:

Configure the following settings on the **Tracking** tab:

Accumulated number of messages	Enter the number of times this filter can be invoked before the alert is sent.
In time period (secs)	Enter the time period in which the accumulated number of messages can occur before and alert is triggered.
Track per client	Select this option if you want to record the accumulated number of messages in the specified time period for each client.

Destination:

All pre-configured alert destinations are displayed in the **Alert name** table. Select the destinations that the Enterprise Gateway sends alerts to if the criteria specified for this filter are met.

Global Schema Cache

Overview

The **Schema Cache** contains XML Schemas that can be used globally by **Schema Validation** filters. You can import XML Schemas from XML Schema files or from WSDL files. WSDL files often contain XML Schemas that define the elements that appear in SOAP messages. To facilitate this, the Policy Studio can import WSDL files from the file system, from a URL, or from a UDDI registry.

When the XML Schema has been imported into the cache and selected in a **Schema Validation** filter, the Enterprise Gateway can retrieve the schema from the cache instead of fetching it from its original location. This improves the runtime performance of the filter, and also ensures that an administrator has complete control over the schemas used to validate messages.

The **Schema Cache** is available as a top-level tree node on the Policy Studio **Policies** tab. The list schemas present in the Schema Cache are shown in the tree. You can view the contents of any of these schemas by clicking the schema. The schema contents are displayed in the **Source** tab on the right.

At any point, you can manually modify the contents of the schema in the **Source** tab. To save the modified contents to the cache, enter **Ctrl-S**.

Adding Schemas to the Cache

To add an XML Schema to the cache, right-click the **Schema Cache** in the tree, and select **Add Schema**. Alternatively, click the **Add Schema** link at the top of the **Schema Cache** screen on the right. The **Load Schema** dialog enables you to load a schema from an XML Schema file directly or from a WSDL file.

Select the **From XML Schema** radio button to load the schema directly from a schema file, and click **Next**. On the next screen, enter or browse to the location of the schema file using the field provided. You can also enter a full URL to pull the schema from a web location. Click the **Finish** button to import the schema into the cache.

Alternatively, if you want to load the schema file from a WSDL file, select the **From WSDL** radio button on the **Select Schema Source** screen, and click **Next**.

The WSDL file can be located from the file system, from a URL, or from a UDDI registry. Select the appropriate option and enter or browse to the location of the WSDL file in the fields provided. If you wish to retrieve the WSDL file from a UDDI registry, click the **WSDL from UDDI** radio button, and click the **Browse UDDI** button. The **Browse UDDI Server for WSDL** dialog enables you to connect to a UDDI and search it for a particular WSDL file. For more information on how to configure this dialog, see the [Retrieving WSDL Files from a UDDI Directory](#) topic.

Testing WSDL Files for WSI Compliance

Before loading the schema from a WSDL file, you can check the WSDL file for compliance with the WS-I Basic Profile. The Basic Profile consists of a set of assertions and guidelines on how to ensure maximum interoperability between different implementations of Web Services. For example, there are recommendations on the SOAP style to use (`document/literal`), how schema information is included in WSDL files, and how message parts are defined to avoid ambiguity for consumers of WSDL files.

The Policy Studio uses the Java version of the WS-Interoperability Testing Tools to test imported WSDL files for compliance with the recommendations in the Basic Profile. A report is generated showing which recommendations have passed and which have failed. While you can still import a WSDL file that does not comply with the Basic Profile, there is no certainty that consumers of the Web Service can use it without encountering problems.

Important Note:

Before you run the WS-I compliance test, you must ensure that the Java version of the Interoperability Testing Tools is installed on the machine on which the Policy Studio is running. You can download these tools from www.ws-i.org [ht-

tp://www.ws-i.org].

To configure the location of the WS-I testing tools, select **Window -> Preferences** from the Policy Studio main menu. In the **Preferences** dialog, select the **WS-I Settings**, and browse to the location of the WS-I testing tools. You must specify the *full path* to these tools (for example, C:\Program Files\WSI_Test_Java_Final_1.1\wsi-test-tools). For more details on configuring WS-I settings, see the [Policy Studio Preferences](#) topic.

Running the WS-I Compliance Test

To run the WS-I compliance test on a WSDL file, perform the following steps:

1. Select **Tools -> Run WS-I Compliance Test** from the Policy Studio main menu.
2. In the **Run WS-I Compliance Test** dialog, browse to the **WSDL File** or specify the **WSDL URL**.
3. Click **OK**. The WS-I Analysis tools run in the background in Policy Studio.

The results of the compliance test are displayed in your browser in a **WS-I Profile Conformance Report**. The overall result of the compliance test is displayed in the **Summary** section. The results of the WS-I compliance tests are grouped by type in the **Artifact: description** section. For example, you can access details for a specific port type, operation, or message by clicking the appropriate link in the **Entry List** table. Each **Entry** displays the results for the relevant WS-I Test Assertions.

Organizing Schemas with Schema Containers

If you intend to add large numbers of schemas to validate different types of requests, it makes sense to organize these types of schemas into different groups. For example, if you have a set of schemas that define types used in requests for a StockQuote Web Service and another set of schemas used to validate requests for a PurchaseOrder Web Service, it makes sense to organize each set of schemas into separate groups (for example, StockQuote Schemas and Purchase-Order Schemas).

The Policy Studio enables you to add *Schema Containers* for this purpose. To add a Schema Container, right-click the **Schema Cache** tree node, and select the **Add Schema Container** menu option. Enter a descriptive name for the container in the field provided on the **Schema Container** dialog.

You can add related schemas under this container by right-clicking the container and selecting the **Add Schema** menu option. You can then load the schema directly from an XML Schema file, or indirectly from a WSDL file in the usual manner.

Furthermore, you can create containers within containers to further organize your schemas. To do this, right-click an existing container, and select the **Add Schema Container** menu option.

A useful feature of Schema Containers is the ability to copy and paste schemas from one container to another. For example, you can use this to copy schemas to a **Test Schemas** container where you can modify them and test them against incoming requests. To do this, right-click a schema, and select the **Copy** menu option. To copy the schema to another container, right-click the destination container and select the **Paste** option.

Important Notes

- Only one schema is allowed per target namespace in any one container. This is because schemas are keyed in the cache using the `targetNamespace` in the schema. Therefore, all elements referenced by the imported schema must be in the same container as the schema. The Policy Studio automatically stores imported and included schemas in the same container as the top-level schema. However, deleting schemas from a container that contains elements referenced by other schemas in the container prevents the set of schemas from successfully validating incoming requests.
If a schema has no `targetNamespace` defined, it is keyed using the full path to the file on the file system.
- If you add a schema that *includes* other schemas (using the `<include>` element) to the Schema Cache, the included schemas are added inline to the top-level parent schema. Because schemas are keyed by target namespace, and because all included schemas *must* belong to the same namespace as the parent schema, it

makes sense to inline the included schemas.

Schema Validation

The **Schema Validation** filter is used to validate XML messages against schemas stored in the cache or in the **Web Services Repository**. This filter is found in the **Content Filtering** category of filters in the Policy Studio. For more information on configuring this filter, see the [Schema Validation](#) topic.

External Connections

Overview

The Enterprise Gateway can leverage your existing Identity Management infrastructure, thus avoiding the need to maintain separate silos of user information. For example, if you already have a database full of user credentials, the Enterprise Gateway can authenticate requests against this database, rather than using its own internal user store. Similarly, the Enterprise Gateway can authorize users, lookup user attributes, and validate certificates against third-party Identity Management servers.

You can add a connection to an external system as a global *External Connection* in the Policy Studio so that it can be reused across all filters and policies. For example, if you create a circuit that authenticates users against an LDAP directory, and then validates an XML signature by retrieving a public key from the same LDAP directory, it makes sense to create a global External Connection for that LDAP directory. You can then *select* the LDAP Connection in both the authentication and XML signature verification filters, rather than having to reconfigure them in both filters.

You can also use External Connections in cases where you want to configure a group of related URLs. This is most useful when you want to round-robin between a number of related URLs to ensure high availability. When the Enterprise Gateway is configured to use a *URL Connection Set* (instead of a single URL), it round-robins between the URLs in the set.

You can configure External Connections by right-clicking the appropriate node (for example, **Database Connections**) on the **External Connections** tab in the Policy Studio. This topic introduces the different types of External Connection and shows where to obtain more details.

Authentication Repository Profiles

The Enterprise Gateway can authenticate users against external databases and LDAP repositories, in addition to its own local user store. You can also use a number of bespoke authentication connectors to enable the Enterprise Gateway to authenticate against specific third-party Identity Management products.

Connection details for these authentication repositories are configured at a global level, making them available for use across authentication (and authorization) filters. This saves the administrator from reconfiguring connection details in each filter.

For example, the available authentication repository types include the following:

- CA SiteMinder Repositories
- Database Repositories
- Entrust GetAccess Repositories
- LDAP Repositories
- Local Repositories (for example, Local User Store)
- Oracle Access Manager Repositories
- Oracle Entitlements Server Repositories
- RADIUS Repositories
- RSA Access Manager Repositories
- Tivoli Repositories

For details how to configure the various authentication repository types, see the [Authentication Repository](#) topic.

Connection Sets

Connection Sets are used by the Enterprise Gateway to round-robin between groups of external servers (for example, RSA Access Manager). You can reuse these global groups when configuring connections to external servers in the

Policy Studio. For this reason, Connection Sets are available on the **External Connections** tab according to the filter from which they are available. For example, Connection Sets under the **RSA ClearTrust Connection Sets** node are available in the **RSA Access Manager** filter.

At runtime, the Enterprise Gateway can round-robin between the servers in the group to ensure that if one of the servers becomes unavailable, the Enterprise Gateway can use one of the other servers in the group.

To add a Connection Set for a particular category of filters, right-click the appropriate node under the **Connection Sets** node on the **External Connections** tab. Select **Add a Connection Set** to display the **Connection Group** dialog. For more details, see the [Configuring Connection Groups](#) topic.

Database Connections

The Enterprise Gateway typically connects to databases to authenticate or authorize users using the Enterprise Gateway's numerous Authentication and Authorization filters. Similarly, the Enterprise Gateway can retrieve user attributes from a database (for example, which can then be used to generate SAML attribute assertions later in the policy).

You can configure database connections globally on the **External Connections** tab, making them available to the various filters that require a database connection. This means that an administrator can reuse the same database connection details across multiple authentication, authorization, and attribute-based filters.

The Enterprise Gateway maintains a JDBC pool of database connections to avoid the overhead of setting up and tearing down connections to service simultaneous requests. This pool is implemented using *Jakarta DBCP (Database Connection Pools)*. The settings in the **Advanced** section of the **Configure Database Connection** dialog are used internally by the Enterprise Gateway to initialize the connection pool. The table at the end of this section shows how the fields correspond to specific configuration DBCP settings.

To configure details for a global database connection, right-click the **Database Connections** node on the **External Connections** tab. Select the **Add a Database Connection** menu option, and configure the fields on the **Configure Database Connection** dialog. For details on configuring these fields, see the [Database Connection](#) topic.

Kerberos Connections

You can configure global **Kerberos Clients**, **Kerberos Services**, and **Kerberos Principals** on the **External Connections** tab in the Policy Studio. When a Kerberos item is configured, it is available for selection in all Kerberos-related configuration screens that require this item. This enables you to share Kerberos configuration items across multiple filters.

For more details on configuring Kerberos clients, services, principals and filters, see the [Kerberos Index](#) page.

LDAP Connections

In the same way that database connections can be configured globally in the Policy Studio (and then reused across individual filters), LDAP connections are also managed globally in the Policy Studio. LDAP connections are used by authentication, authorization, and attribute filters. Filters that require a public key (from a public-private key pair) can also retrieve the key from an LDAP source.

When a filter that uses an LDAP directory is run for the first time, it binds to the LDAP directory using the connection details configured on the **Configure LDAP Server** dialog. Usually the connection details include the username and password of an administrator user who has read access to all users in the LDAP directory for whom you wish to retrieve attributes or authenticate.

To configure a global LDAP connection, right-click the **LDAP Connections** node on the **External Connections** tab. Select the **Add an LDAP Connection** menu option, and configure the following fields on the **Configure LDAP Server** dialog:

Name:

Enter a name for the LDAP connection in the **Name** field.

URL:

Enter the location of the LDAP directory in the **URL** field. An LDAP URL is a combination of the protocol (LDAP), the IP address or host name of the LDAP server, and the port number for the LDAP service. By default, port 389 is reserved for LDAP connections. The following is an example of a typical LDAP directory URL:

```
ldap://192.168.0.45:389
```

Authentication Type:

If the configured LDAP directory requires clients to authenticate to it, you must select the appropriate authentication method in the **Type** drop-down list. When the Enterprise Gateway connects to the LDAP directory, it is authenticated to it using the method selected here. The Enterprise Gateway can authenticate to an LDAP directory using the following methods:

No Authentication

No authentication credentials need be submitted to the LDAP server. The client connects anonymously to the server. Typically, a client is only allowed to perform read operations when connected anonymously to the LDAP server. You do not need to enter any authentication details for this authentication method.

Simple Authentication

Simple authentication involves sending a user name and corresponding password in clear text to the LDAP server. Because the password is passed in clear text, you should connect over an encrypted channel (for example, SSL).

You do not need to specify a **Realm** when using Simple authentication. The realm is only used when a hash of the password is supplied (for Digest-MD5). However, when the LDAP server contains multiple realms, and the specified user name is present in more than one of these realms, it is at the discretion of the specific LDAP server as to which user name binds *bind* to it.

Select the **SSL Enabled** checkbox to force the Enterprise Gateway to connect to the LDAP server over SSL. To successfully establish SSL connections with the LDAP server, you must import its certificate into the Enterprise Gateway's certificate store. You can do this using the global [Certificates](#) screen.

Digest-MD5 Authentication

With Digest-MD5 authentication, the server generates some data and sends it to the client. The client encrypts this data with its password according to the MD5 algorithm. The LDAP server then uses the client's stored password to decrypt the data and hence authenticate the user.

The **Realm** field is optional, but may be necessary if the LDAP server contains multiple realms. If a realm is specified, the LDAP server attempts to authenticate the user for the specified realm only.

Test Connection:

When you have entered all the necessary connection details, you can test the configuration by clicking the **Test Connection** button. This enables you to detect configuration errors at design time, rather than at runtime.

OCSP Connections

The Enterprise Gateway can use OCSP (Online Certificate Status Protocol) to validate a certificate against an OCSP responder or group of responders. OCSP requests for certificate validation can be signed by a key from the **Certificate Store**, and the response from the OCSP responder can be optionally validated.

An OCSP Connection typically comprises a *group* of OCSP Responder URLs, together with options to sign OCSP requests and validate OCSP responses. To configure a global OCSP Connection, right-click the **OCSP Connection** node on the **External Connections** tab. Select the **Add an OCSP Connection** option to display the **Certificate Validation - OCSP** dialog. For more details, see the [OCSP Certificate Validation](#) topic.

Important Note:

When an OCSP Connection is added in this manner, it is available for selection in the **OCSP Certificate Validation** filter, which is found under the **Certificate** category of filters in the Policy Studio.

Proxy Servers

You can configure proxy servers on the **External Connections** tab, which can then be specified in the **Connection** and **Connect To URL** filters. When configured, the filter connects to the proxy server, which routes the message to the destination server.

To configure a proxy server, click the **External Connections** tab, and select **Proxy Servers** -> **Add a Proxy Server**. For details on how to configure the settings the **Proxy Server Settings** dialog, see the [Proxy Server](#) topic.

RADIUS Clients

The Remote Authentication Dial In User Service (RADIUS) protocol provides centralized authentication and authorization for clients connecting to remote services.

To configure a client connection to a remote server over the RADIUS protocol, click the **External Connections** tab, and select **RADIUS Clients** -> **Add a RADIUS Client**. For details on how to configure the settings the **RADIUS Client** dialog, see the [RADIUS Clients](#) topic.

For details on how to configure a RADIUS Authentication Repository, see the [Authentication Repository](#) topic.

SiteMinder

To add a CA SiteMinder connection, right-click the **SiteMinder/SOA Security Manager** node on the **External Connections** tab, and select **Add a SiteMinder Connection** to display the **SiteMinder Connection Details** dialog. For details on configuring the fields on this dialog, see the [SiteMinder/SOA Security Manager Connection](#) topic.

SMTP Servers

SMTP Servers are global configuration items that can be configured on the **External Connections** tab, and then referenced by the **SMTP** filter. Right-click the **SMTP Servers** node on the **External Connections** tab, and select **Add an SMTP Server**. You must configure the following fields:

SMTP Server Hostname:

Enter the hostname or IP address of the SMTP server.

Port:

Enter the SMTP port on the specified server hostname. By default, SMTP servers run on port 25.

Connection Security:

Select the security required for the connection to the SMTP server (NONE, SSL, or TLS). Defaults to NONE.

User Name:

Enter the user name of a registered user that can access the SMTP server.

Password:

Enter the password for this user.

SOA Security Manager

To add a CA SiteMinder connection, right-click the **SiteMinder/SOA Security Manager** node on the **External Connections** tab, and select **Add a SOA Security Manager Connection** to display the **SOA Security Manager Connection Details** dialog. For details on configuring the fields on this dialog, see the [SiteMinder/SOA Security Manager Connection](#) topic.

Syslog Servers

You can configure Syslog Servers globally, and then select them as a customized logging destination for a Process. Right-click the **Syslog Servers** node on the **External Connections** tab, and select the **Add a Syslog Server** menu option. Complete the following fields on the **Syslog Server** dialog:

Name:

Enter a name for the Syslog server.

Host:

Specify the host on which the Syslog daemon is running.

Facility:

Select the Syslog facility that you want to log to.

To configure a Process to log to this global Syslog Server, complete the following steps:

1. On the Policy Studio **Services** tab, right-click the name of the Process in the tree (for example, **Oracle Enterprise Gateway**).
2. Select the **Logging -> Custom** option.
3. On the **Configure Logging** dialog, open the **Syslog** tab, and select the globally configured Syslog Server from the drop-down list.
4. Ensure to select the **Enable logging to Syslog** checkbox.

TIBCO

You can add connections to the following TIBCO products:

- TIBCO Enterprise Messaging System (EMS)
- TIBCO Rendezvous Daemon

To add a TIBCO EMS connection, right-click the **TIBCO EMS Connection** node on the **External Connections** tab, and select **Add a TIBCO EMS Connection**. For details on configuring the fields on the dialog, see the [TIBCO Enterprise Messaging System Connection](#) topic.

To add a TIBCO Rendezvous Daemon, right-click the **TIBCO Rendezvous Daemon** node on the **External Connections** tab, and select **Add a TIBCO Rendezvous Daemon**. For details on configuring the fields on the dialog, see the [TIBCO Rendezvous Daemon](#) topic.

Tivoli

You can create a connection to an IBM Tivoli server to enable integration between the Enterprise Gateway and Tivoli Access Manager. Tivoli connections can then be used by the Enterprise Gateway's Tivoli filter to delegate authentication and authorization decisions to Tivoli Access Manager, and to leverage existing Tivoli Access Manager policies.

To add a Tivoli connection, right-click the **Tivoli Connections** node on the **External Connections** tab, and select **Add a Tivoli Connection**. For details on configuring the fields on the **Tivoli Configuration** dialog, see the [Tivoli Integration](#) topic.

URL Connection Sets

URL Connection Sets are used by Enterprise Gateway filters to round-robin between groups of external servers (for example, Entrust GetAccess, OCSP, SAML PDP, or XKMS). These global groups can then be reused when configuring these filters in the Policy Studio. For this reason, URL Connection Sets are available on the **External Connections** tab according to the filters from which they are available. For example, URL sets under the **OCSP URL Sets** node are available in the **OCSP Certificate Validation** filter, while sets under the **XKMS URL Sets** are only available from the **XKMS Certificate Validation** filter.

At runtime, the Enterprise Gateway can round-robin between the servers in the group to ensure that if one of the servers becomes unavailable, the Enterprise Gateway can use one of the other servers in the group.

To add a URL Connection Set for a particular category of filters, right-click the appropriate node under the **URL Connection Sets** node on the **External Connections** tab. Select the **Add a URL Set** option to display the **URL Group** dialog. For more details, see the [Configuring URL Groups](#) topic.

XKMS Connections

The Enterprise Gateway can also validate certificates against an XKMS (XML Key Management Service) responder or group of responders. An XKMS Connection consists of a group of XKMS responders to validate certificates against, coupled with the signing key to use for signing requests to each of the responders in the group.

To add a global XKMS Connection, right-click the **XKMS Connection** node on the **External Connections** tab, and select the **Add an XKMS Connection** option to display the **Certificate Validation - XKMS** dialog. For more details, see the [XKMS Certificate Validation](#) topic.

All global XKMS Connections are available for selection when configuring the **Certificate Validation - XKMS** filter. This saves the administrator from reconfiguring XKMS connection details across multiple filters.

Global Caches

Overview

In cases where a Web Service is serving the same request (and generating the same response) over and over again, it makes sense to use a caching mechanism. When a cache is employed, a unique identifier for the request is cached together with the corresponding response for this request. If an identical request is received, the response can be retrieved from the cache instead of forcing the Web Service to reprocess the identical request and generate the same response. The use of caching in this way helps divert unnecessary traffic from the Web Service and makes it more responsive to new requests.

For example, assume you have deployed a Web Service that returns a list of cities in the USA from an external database, which is then used by a variety of Web-based applications. Because the names and quantity of cities in the USA are relatively constant, if the Web Service is handling hundreds or thousands of requests every day, this represents a serious waste of processing time and effort, especially considering that the database that contains the relatively fixed list of city names is hosted on a separate machine to the service.

If you assume that the list of cities in the database does not change very often, it makes sense to use Oracle Enterprise Gateway to cache the response from the Web Service that contains the list of cities. Then when a request for this Web Service is identified by the Enterprise Gateway, the cached response can be returned to the client. This approach results in the following performance improvements:

- The Enterprise Gateway does not have to route the message on to the Web Service, therefore saving the processing effort required, and perhaps more importantly, saving the time it takes for the round trip.
- The Web Service does not have to waste processing power on generating the same list over and over again, therefore making it more responsive to requests for other services.
- Assuming a naive implementation of database retrieval and caching, the Web Service does not have to query the database (over the network) and collate the results over and over again for every request.

The caching mechanism used in the Enterprise Gateway offers full control over the size of the cache, the lifetime of objects in the cache, whether objects are cached to disk or not, and even whether caches can be replicated across multiple instances of Enterprise Gateways. The following sections describe how to configure both local and distributed caches in the Enterprise Gateway, concluding with a detailed example of how to configure a circuit to cache responses.

Local Caches

Local caches are used where a single Enterprise Gateway instance has been deployed. In such cases, you do not need to replicate caches across multiple running instances of the Enterprise Gateway.

In the Policy Studio, you can add a local cache, by selecting the top-level **Caches** tree item on the **External Connections** tab, and clicking the **Add** button at the bottom right of the screen. Select **Add Local Cache** from the menu. You can configure the following fields on the **Configure Local Cache** dialog:

Cache Name:

Enter a name for the cache.

Maximum Elements in Memory:

Enter the maximum number of objects that can be in memory at any one time.

Maximum Elements on Disk:

Sets the maximum number of objects that can be stored in the disk store at any one time. A value of zero indicates an unlimited number of objects.

Eternal:

If this option is selected, objects stored in the caches never expire and timeouts have no effect.

Overflow to Disk:

Select this option if you want the cache to overflow to disk when the number of objects in memory has reached the amount set in the **Maximum Elements in Memory** field above.

Note:

The following fields are optional:

Time to Idle:

Determines the maximum amount of time (in seconds) between accesses that an object can remain idle before it expires. A value of zero indicates that objects can idle for infinity, which is the default value. Note that if the **Eternal** field is selected, this setting is ignored.

Time to Live:

Sets the maximum time between when an object is created and when it expires. The default value is zero, which means that the object can live for infinity. Note that if the **Eternal** field is selected, this setting is ignored.

Persist to Disk:

If selected, the disk store is persisted between JVM restarts. This option is disabled by default.

Disk Expiry Interval:

Configures the number of seconds between runs of the disk expiry thread. The default is 120 seconds.

Disk Spool Buffer Size:

Indicates the size of memory (in MBs) to allocate the disk store for a spool buffer. Writes are made to this memory and then asynchronously written to disk. The default size is 30 MB. If you get `OutOfMemory` exceptions, you may consider lowering this value. However, if you notice poor performance, you should increase the value.

Eviction Policy:

Select the eviction policy that the cache uses to evict objects from the cache. The default policy is **Least Recently Used**. However, you can also use **First in First Out** and **Less Frequently Used**.

Distributed Caches

If you have deployed several Enterprise Gateways throughout your network, you need to employ a distributed cache. In this scenario, each Enterprise Gateway has its own local copy of the cache but registers a cache event listener that replicates messages to the other caches so that put, remove, expiry, and delete events on a single cache are duplicated across all other caches.

You can configure a distributed cache by selecting the top-level **Caches** tree item on the **External Connections** tab, and clicking the **Add** button at the bottom right of the screen. Select **Add Distributed Cache** from the menu, and configure the following fields on the **Configure Distributed Cache** dialog:

Note:

Many of the settings for the distributed cache are identical to those for the local cache. For details on how to configure these fields, see the [Local Caches](#) section. The following information refers to fields that are not displayed on both dialogs.

Event Listener Class Name:

Enter the name of the listener factory class that enables this cache to register listeners for cache events, such as put, remove, delete, and expire.

Properties Separator:

Specify the character to use to separate the list of properties.

Properties:

Specify the properties to pass to the `RMICacheReplicatorFactory`. The following properties are available:

- `replicatePuts=true | false`

- Determines whether new elements placed in a cache are replicated to other caches. Default is `true`.
- `replicateUpdates=true | false`
 Determines whether new elements that override (update) existing elements with the same key in a cache are replicated. Default is `true`.
- `replicateRemovals=true`
 Determines whether element removals are replicated. Default is `true`.
- `replicateAsynchronously=true | false`
 Determines whether replications are asynchronous (`true`) or synchronous (`false`). Default is `false`.
- `replicateUpdatesViaCopy=true | false`
 Determines whether new elements are copied to other caches (`true`) or a remove message is sent (`false`). Default is `true`.
- `asynchronousReplicationIntervalMillis=[number of ms]`
 The asynchronous replicator runs at a set interval of milliseconds. The default is 1000 and the minimum is 10. This property is only applicable if `replicateAsynchronously=true`.

Cache Bootstrap Class Name:

Specifies a `BootstrapCacheLoader` factory that the cache can call on initialization to pre-populate itself. The `RMIBootstrapCacheLoader` bootstraps caches in clusters where `RMICacheReplicators` are used.

Properties Separator:

The character entered here is used to separate the list of properties listed in the field below.

Properties:

The properties listed here are used to initialize the `RMIBootstrapCacheLoaderFactory`. The following properties are recognized:

- `bootstrapAsynchronously=true | false`
 Determines whether the bootstrap happens in the background after the cache has started (`true`), or if bootstrapping must complete before the cache is made available (`false`). Default is `true`.
- `maximumChunkSizeBytes=[integer]`
 Caches can potentially grow larger than the memory limits on the JVM. This property enables the bootstrapper to fetch elements in chunks. The default chunk size is 5000000 (5 MB).

Cache Settings

In a distributed cache, there is no master cache controlling all caches in the group. Instead, each cache is a peer in the group and needs to know where all the other peers in the group are located. *Peer Discovery* and *Peer Listeners* are two essential parts of any distributed cache system.

You can configure these by right-clicking the top-level **Caches** tree item on the **External Connections** tab, and selecting **Edit Cache Settings**. Configure the following fields on the **Cache Settings** dialog:

Peer Provider Class:

By default, the built-in peer discovery class factory is used:

```
net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory
```

Properties Separator:

Specify the token used as the separator for the list of properties in the next field.

Properties:

The properties listed here specify whether the peer discovery mechanism is automatic or manual. If the automatic mechanism is used, each peer uses TCP multicast to establish and maintain a multicast group. This is the default option because it requires minimal configuration and peers can be automatically added and removed from the group. Each peer pings the group every second. If a peer has not pinged any of the other peers after 5 seconds, it is dropped from the group, while a new peer is admitted to the group if it starts pinging the other peers. To use automatic peer discovery, en-

sure that the `peerDiscovery` setting is set to `automatic`. You can specify the multicast address and port using the `multicastGroupAddress` and `multicastGroupPort` settings. You can specify the time to live for multicast datagrams using the `timeToLive` setting.

Alternatively, you can configure a manual peer discovery mechanism, whereby each peer definitively lists the peers that it wants to communicate with. This should only be used in networks where there are problems propagating multicast datagrams. To use a manual peer discovery mechanism, make sure the `peerDiscovery` setting is set to `manual`. The list of RMI URLs of the other peers in the group must also be specified, for example:

```
rmiUrls=//server2:40001/sampleCache1|//server2:40001/sampleCache2 .
```

Peer Listener Class:

The peer listener class specified is responsible for listening for messages from peers in the group.

Properties Separator:

Specify the token used to separate the list of properties.

Properties:

The properties entered configure the way the listener behaves. Valid properties are as follows:

- **hostname (optional)**
Hostname of the machine on which the listener is listening. Note that, by default, this is set to `localhost`, which maps to the local loopback address of `127.0.0.1`, which is not addressable from another machine on the network. If you intend this cache to be used over the network, you should change this address to the IP address of the network interface on which the listener is listening.
- **port (mandatory)**
Specify the port on which the listener is listening, which by default is `4001`. This setting is mandatory.
- **socketTimeoutMillis (optional)**
Enter the number of seconds that client sockets wait when sending messages to this listener until they give up. The default is `2000` ms.

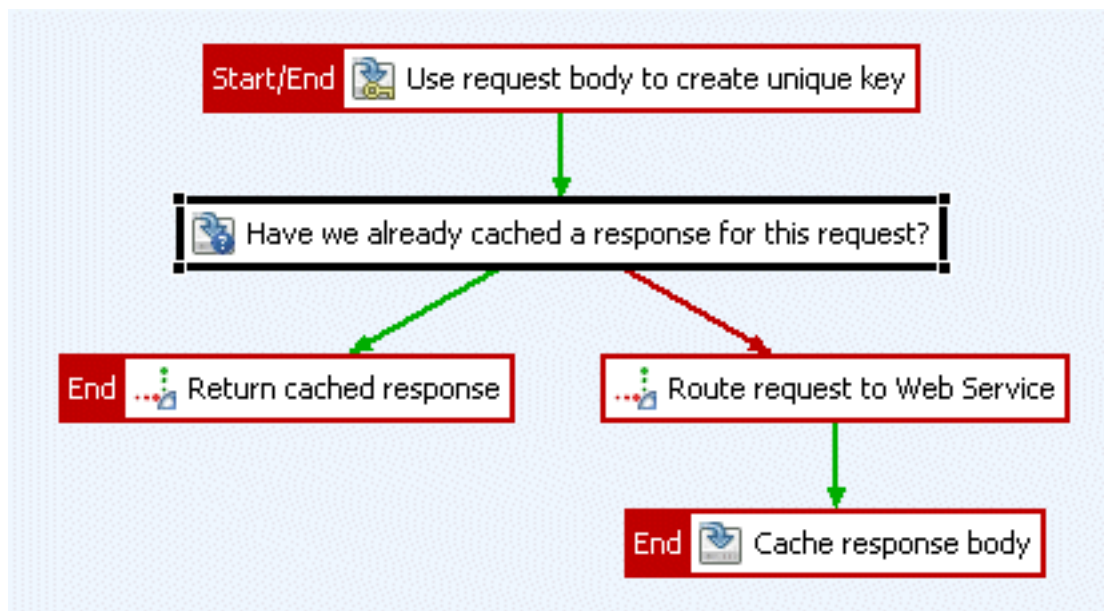
Notify replicators of removal of items during refresh:

A server refresh automatically purges all items from the cache (for example, when configuration updates are deployed to the Enterprise Gateway). If this checkbox is selected, the contents of each peer in the group are also purged. This avoids a situation where a single peer is refreshed (and has its contents purged), but the other peers in the group are not purged. If this option is not selected, the refreshed peer attempts to bootstrap itself to the other peers in the group, resulting in the cache items becoming replicated in the refreshed cache. This effectively negates the effect of the server refresh and may result in inconsistent behavior.

Example of Caching Response Messages

This simple example shows how to construct a circuit that caches responses from the Web Service. It uses the request body to identify identical successive requests. In other words, if the Enterprise Gateway receives two successive requests with an identical message body, it returns the corresponding response from the cache instead of routing the request to the Web Service.

The following diagram illustrates the complete circuit:



The logic of the circuit is summarized as follows:

1. The purpose of the first filter is to configure what part of the request you want to use to identify unique requests. This example uses the request body as the unique key, which is then used to look up the appropriate response message from the cache.
2. The second filter looks up the request body in the response cache to see if it contains the request body. If it does, the response message that corresponds to this request is returned to the client.
3. If it does not, the request is routed to the Web Service, which processes it (by connecting to a database over the network and running a SQL statement) and returns a response to the Enterprise Gateway.
4. The Enterprise Gateway then returns the response to the client and caches it in the response cache.
5. When the next identical request is received by the Enterprise Gateway, the corresponding response is located in the responses cache and returned immediately to the client.

You must configure the following caching filters to achieve this circuit. For convenience, the routing filters will not be included because the configuration options depend on your target Web Service.

Create Key Filter:

This filter is used to decide what part of the request is used in order for a request to be considered unique. Different parts of the request can be identified internally using Oracle message attributes, for example, `content.body` contains the request body. The following fields must be configured for this filter:

- **Name:** Use request body to create unique key
- **Attribute Name:** `content.body`

Is Cached?:

This filter looks up the cache to see if a response has been stored for the current request. It looks up the cache using a message attribute, which is `message.key` by default. The `message.key` attribute contains a hash of the request message, and can be used as the key for objects in the cache. If the key is found in the cache, the value of the key (cached response for this request) is written to the `content.body` attribute, which can be returned to the client using the **Reflect** filter. You must configure the following fields:

- **Name:** Is a response for this request already cached?

- **Cache containing key:** Response Cache (assuming you have created a cache of this name)
- **Attribute Containing Key:** `message.key`
- **Overwrite attribute name if found:** `content.body`

Reflect:

If the **Is Cached?** filter passes, it retrieves the response from the cache and stores it in the `content.body` message attribute. The **Reflect** filter is used to return the cached response to the client.

Routing:

If a response for this request could not be located in the cache, the Enterprise Gateway routes the request to the Web Service, and waits for a response. For more details on how to route messages, see the [Routing Configuration](#) tutorial.

Cache Attribute:

When the response has been received from the Web Service, it should be cached for future use. The **Cache Attribute** filter is used to configure the key used to look up the cache and which aspect of the response message is stored as the key value in the cache. Note that this example specifies the value of the `content.body` attribute to cache. Because this filter is configured *after* the routing filters, this attribute contains the response message. Note also that the value entered in the **Attribute Key** field should match that entered in the **Attribute containing key** field in the **Is Cached?** filter. You must configure the following fields:

- **Name:** Cache response body
- **Cache to use:** Response Cache
- **Attribute key:** `message.key`
- **Attribute name to store:** `content.body`

For more information on these filters, see the following topics:

<i>Filter</i>	<i>Topic</i>
Create Key	Create Key
Is Cached?	Is Cached?
Cache Attribute	Cache Attribute
Remove Cached Attribute	Remove Cached Attribute

Retrieve Attribute from Database

Overview

The Enterprise Gateway can retrieve user attributes from a specified database, or write user attributes to a specified database. It can do this by running an SQL query on the specified database, or by invoking a stored procedure call.

General Configuration

Configure the following field:

Name:

Enter an appropriate name for this filter.

Database

Configure the following fields on the **Database** tab:

Database Location:

The Enterprise Gateway searches the selected database for the user's attributes. The default option available from the drop-down list is the `Default Database Connection`.

You can configure database connection details on the **External Connections** tab in Policy Studio. To add a connection, right-click the **Database Connections** node, and select **Add a Database Connection**. For details on completing the **Configure Database Connection** dialog, see the [Database Connection](#) topic.

Database Statements:

The **Database Statements** table lists the currently configured SQL queries or stored procedure calls. These queries and calls retrieve certain user attributes from the database selected in the **Database Location** field.

You can edit and delete existing queries by selecting them from the drop-down list and clicking the **Edit** and **Delete** buttons. For more information on how to configure a **Database Query**, see the [Database Query Configuration](#) topic.

Advanced

On the **Advanced** tab, configure the following fields in the **User Attribute Extraction** section:

Place query results into user attribute list:

Select whether to place the database query results in a user attribute list using this checkbox (selected by default). When selected, the query results are placed in the `attribute.lookup.list` message attribute.

Associate attributes with user ID

When the **Place query results into message attribute list** checkbox is selected, you can select or enter a user ID to associate with the user attributes. For example, if the user name is stored as `admin` in the database, you must select the message attribute containing the value `admin`. The Enterprise Gateway then looks up the database using this name. By default, the user ID is stored in the `authentication.subject.id` message attribute.

Configure the following fields on the **Attribute Naming** section:

Prefix for message attribute names:

You can specify an optional prefix for message attribute names. The default prefix is `user.`

Attribute name for stored procedure out parameters:

You can also specify an attribute name for stored procedure out parameters. The default prefix is `out.param.value.`

Case for attribute names:

You can specify whether attribute names are in lower case or upper case. The default is lower case.

Retrieve Attributes from Directory Server

Overview

The Enterprise Gateway can leverage an existing directory server by querying it for user profile data. The **Retrieve from Directory Server** filter can lookup a user, retrieve that user's attributes, and set them to the `attribute.lookup.list` message attribute, which stores a map of name-value pairs.

General Configuration

The following fields are available on the **Retrieve From Directory Server** filter configuration screen:

Name:

Enter a name for this filter here.

LDAP Directory:

The Enterprise Gateway will query the LDAP directory selected here for user attributes. To configure an LDAP directory, click the **Add/Edit** button. Take a look at the [LDAP Configuration](#) help page for more information on how to do this. An LDAP connection will be retrieved from a pool of connections at run-time.

Retrieve Unique User Identity

Use this section to select the user whose profile the Enterprise Gateway will look up in the directory server. The user ID can be taken from a message attribute or looked up from an LDAP directory.

From Message Attribute:

Select this option if the user ID is stored in a message attribute. A user's credentials are stored in the `authentication.subject.id` message attribute after authenticating to the Enterprise Gateway and so this is the most likely attribute to enter in this field. Typically this will contain the Distinguished Name (DName) or username of the authenticated user. The name extracted from the selected message attribute will be used to query the directory server.

From LDAP Search:

In cases where you have not already obtained the user's identity and the `authentication.subject.id` attribute has not been pre-populated by a prior authentication filter, you must configure the Enterprise Gateway to retrieve the user's identity from an LDAP search. Click the **Configure Directory Search** button to configure the search criteria to use to retrieve the user's unique DName from the LDAP repository.

Retrieve Attributes

This section instructs the Enterprise Gateway to search the LDAP tree according to certain conditions in order to locate a specific user profile. Once the appropriate profile has been retrieved, the Enterprise Gateway will extract the specified user attributes from it.

Base Criteria:

The value entered specifies where the Enterprise Gateway should begin searching the LDAP directory. You can enter a property representing the value of a message attribute in this field. The two most likely message attributes to use are the authenticated client's ID and DistinguishedName. The corresponding property values are supplied by default:

- `${authentication.subject.id}`
- `${authentication.subject.dname}`

Search Filter:

This is the name given by the particular LDAP directory to the *User* class. This will depend on the type of LDAP directory that is configured. You can also use properties to represent the value of a message attribute. For example, the

`user.role` attribute can be used to store the user class. The syntax for using the property representing this attribute is as follows:

- `(objectclass=${user.role})`

Search Scope:

If the Enterprise Gateway retrieves a user profile node from the LDAP tree, the option selected here dictates the level that the Enterprise Gateway will search the node to. The options available are:

- Object level
- One level
- Sub-tree

Select the **Unique Result** checkbox to force the Enterprise Gateway to retrieve a unique user profile from the LDAP directory. This is useful in cases where the LDAP search has returned several profiles.

The **Attribute Name** table lists the attributes that the Enterprise Gateway will retrieve from the user profile. If no attributes are explicitly listed here, the Enterprise Gateway will extract all user attributes. In both cases, the retrieved attributes will be set to the `attribute.lookup.list` message attribute.

Click the **Add** button to add the name of an attribute to extract from the returned user profile. Simply enter the name of the attribute to extract from the profile in the **Attribute Name** field of the **Attribute Lookup** dialog.

Important Note:

It is important to note the following:

- If the search returns results for more than one user and the **Unique Result** option is enabled, an error will be generated. If this option is not enabled, all attributes will be merged.
- If an attribute is configured that does not exist in the repository, no error will be generated.
- If no attributes are configured, all attributes present for the user will be retrieved.

Retrieve Attribute from HTTP Header

Overview

The **Retrieve from HTTP Header** attribute retrieval filter can be used to retrieve the value of an HTTP header and set it to a message attribute. For example, it is possible to retrieve an X.509 certificate from a "USER_CERT" HTTP header and set it to the `authentication.cert` message attribute. This certificate can then be used by the filter's successors. The following HTTP request shows an example of such a header:

```
POST /services/getEmployee HTTP/1.1
Host: localhost:8095
Content-Length: 21
SOAPAction: HelloService
USER_CERT: MIIEDCCA0 ...9aKD1fEQgJ
```

It is also possible to retrieve a value from a named query string parameter and set this to the specified message attribute. The following example shows a request URL that contains a query string:

```
http://hostname.com/services/getEmployee?first=john&last=smith
```

In the above example the query string is "first=john&last=smith". As is clear from the example, query strings consist of attribute name-value pairs. Each name-value pair is separated by the '&' character.

Configuration

The following fields are available on the **Retrieve from HTTP Header** filter configuration screen:

Name:

Enter a name for this filter here.

HTTP Header Name:

Enter the name of the HTTP header that will contain the value that we want to set to the message attribute.

Base64 Decode:

Check this box if the extracted value should be base64 decoded before it is set to the message attribute.

Use Query String Parameters:

Check this box if the Enterprise Gateway should attempt to extract the **HTTP Header Name** from the query string parameters instead of from the HTTP headers.

Attribute ID:

Finally, select the attribute that will be used to store the value extracted from the request.

Retrieve Attribute from SAML Attribute Assertion

Overview

A SAML (Security Assertion Markup Language) attribute assertion contains information about a user in the form of a series of attributes. The **Retrieve from SAML Attribute Assertion** can retrieve these attributes and store them in the `attribute.lookup.list` message attribute.

The following SAML attribute assertion contains 3 attributes, "role", "email", and "dept". The **Retrieve from SAML Attribute Assertion** will store all 3 attributes and their values in the `attribute.lookup.list` message attribute.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance">
  <soap:Header>
    <wsse:Security>
      <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
        ID="Id-0000010a3c4ff12c-00000000000000002"
        IssueInstant="2006-03-27T15:26:12Z" Version="2.0">
        <saml:Issuer Format="urn:oasis ... WindowsDomainQualifiedNames">
          TestCA
        </saml:Issuer>
        <saml:Subject>
          <saml:NameIdentifier Format="urn:oasis ... WindowsDomainQualifiedNames">
            TestUser
          </saml:NameIdentifier>
        </saml:Subject>
        <saml:Conditions NotBefore="2005-03-27T15:20:40Z"
          NotOnOrAfter="2028-03-27T17:20:40Z"/>
        <saml:AttributeStatement>
          <saml:Attribute Name="role" NameFormat="http://www.oracle.com">
            <saml:AttributeValue>admin</saml:AttributeValue>
          </saml:Attribute>
          <saml:Attribute Name="email" NameFormat="http://www.oracle.com">
            <saml:AttributeValue>joe@oracle.com</saml:AttributeValue>
          </saml:Attribute>
          <saml:Attribute Name="dept" NameFormat="">
            <saml:AttributeValue>engineering</saml:AttributeValue>
          </saml:Attribute>
        </saml:AttributeStatement>
      </saml:Assertion>
    </wsse:Security>
  </soap:Header>

  <soap:Body>
    <product>
      <name>Enterprise Gateway</name>
      <company>Oracle</company>
      <description>Web Services Security</description>
    </product>
  </soap:Body>
</soap:Envelope>
```

Details

The following fields are available on the **Details** configuration tab:

Name:

Enter a name for this filter here.

SOAP Actor/Role:

If you expect the SAML assertion to be embedded within a WS-Security block, you can identify this block by specifying the SOAP Actor or Role of the WS-Security header that contains the assertion.

XPath Expression:

Alternatively, if the assertion is not contained within a WS-Security block, you can enter an XPath expression to locate the attribute assertion. XPath expressions can be added by selecting the **Add** button. Expressions can be edited and deleted by selecting an XPath expression and clicking the **Add** and **Delete** buttons respectively.

SAML Namespace:

Select the SAML namespace that must be used on the SAML assertion in order for this filter to succeed. If you do not wish to check the namespace, select the "Do not check version" option from the dropdown.

SAML Version:

Enter the SAML Version that the assertion must adhere to by entering the major version in the 1st field, followed by the minor version in the 2nd field. For example, for SAML version 2.0, enter "2" in the 1st field and "0" in the 2nd field.

Drift Time:

When the Enterprise Gateway receives a SAML attribute assertion, it first checks to make sure that it has not expired. The lifetime of the assertion is specified using the "NotBefore" and "NotOnOrAfter" attributes of the <Conditions> element in the assertion itself. The Enterprise Gateway makes sure that the time at which it validates the assertion is between the "NotBefore" and "NotOnOrAfter" times.

The **Drift Time** is used to account for differences in the clock time of the machine that generated the assertion and the machine hosting the Enterprise Gateway. The time specified here will be subtracted from the time at which the Enterprise Gateway attempts to validate the assertion.

Trusted Issuers

You can use the table on this tab to select the issuers that you consider trusted. In other words, this filter will only accept assertions that have been issued by the SAML Authorities selected here.

Click the **Add** button to display the **Trusted Issuers** screen. Select the Distinguished Name of a SAML Authority whose certificate has been added to the Certificate Store and click the **OK** button. Repeat this step to add more SAML Authorities to the list of trusted issuers.

Subject Configuration

The Enterprise Gateway can perform some very basic authentication checks on the subject or sender of the assertion using the options available on the **Subject** tab. The Enterprise Gateway can compare the subject of the assertion (i.e. the <NameIdentifier>) to one of the following values:

- **Subject of the Authentication Filter:**
Select this option if the user specified in the <NameIdentifier> element must match the user that authenticated to the Enterprise Gateway. The subject of the authentication event is stored in the `authentication.subject.id` message attribute.
- **A User-Specified Value:**
This option can be used if the <NameIdentifier> must match a user-specified value. Select this radio button and enter the value in the field provided.
- **No Authentication:**
If the **Neither of the above** radio button is selected, the Enterprise Gateway will not attempt to match the <NameIdentifier> to any value.

Lookup Attributes

The **Lookup Attributes** tab is used to determine what attributes the Enterprise Gateway should extract from the SAML attribute assertion. Extracted attributes and their values will be set to the `attribute.lookup.list` message attribute.

The table lists the attributes that the Enterprise Gateway will extract from the assertion and set to the `attribute.lookup.list`.

Alternatively, check the **Extract all of the attributes from the SAML assertion** checkbox to configure the Enterprise Gateway to extract all attributes from the assertion. All attributes will be set to the `attribute.lookup.list` message attribute.

To configure a specific attribute to lookup in the message, click the **Add** button to display the **Attribute Lookup** dialog. Enter the value of the "Name" attribute of the `<Attribute>` element in the **Name** field. Enter the value of the "Name-Format" attribute of the `<Attribute>` element in the **Namespace** field.

SAML PDP Attributes

Overview

The Enterprise Gateway can request information about an authenticated end-user in the form of user *attributes* from a SAML PDP (Policy Decision Point) using the SAML Protocol (SAML). In such cases, the Enterprise Gateway presents evidence to the PDP in the form of some user credentials, such as the Distinguished Name of a client's X.509 certificate.

The PDP looks up its configured user store and retrieves attributes associated with that user. The attributes are inserted into a SAML attribute assertion and returned to the Enterprise Gateway in a SAML response. The assertion and/or SAML response is usually signed by the PDP.

When the Enterprise Gateway receives the SAML response, it performs a number of checks on the response, such as validating the PDP signature and certificate, and examining the assertion. It can also insert the SAML attribute assertion into the original message for consumption by a downstream Web Service.

Request Configuration

This section describes how the Enterprise Gateway should package the SAML request before sending it to the SAML PDP. You can configure a group of SAML PDPs to which the Enterprise Gateway connects in a round-robin fashion if one or more of the PDPs are unavailable. This is known as a SAML PDP URL Set. You can configure a SAML PDP URL Set on the **External Connections** tab in the Policy Studio. Expand the **URL Connection Sets** node, right-click **SAML PDP URL Set**, and select **Add a URL Set**. For more details, see the [Configuring URL Groups](#) topic.

When you have configured a group of SAML PDPs to connect to, you can configure the following general fields:

- **SAML PDP URL Set:**
Select a previously configured SAML PDP URL Set from the drop-down list. You can configure a SAML PDP URL Set on the **External Connections** tab.
- **SOAPAction:**
Enter the SOAP Action required to send SAML Protocol requests to the PDP. Click the **Use Default** button to use the following default SOAP Action as specified by the SAML Protocol:
`http://www.oasis-open.org/committees/security`
- **SAML Version:**
Select the SAML version to use in the SAML request.
- **Signing Key:**
If the SAML request is to be signed, click the **Signing Key** button, and select the appropriate signing key from the Certificate Store.

SAML Subject:

These details describe the *subject* of the SAML assertion. Complete the following fields:

- **Subject Attribute:**
Select the message attribute that contains the name of an authenticated username. By default, the `authentication.subject.id` message attribute is selected, which contains the username of the authenticated user.
- **Subject Format:**
Select the format of the message attribute selected in the **Subject Attribute** field above. Note that there is no need to select a format here if the **Subject Attribute** field is set to `authentication.subject.id`

Subject Confirmation:

The settings on the **Confirmation Method** tab determine how the `<SubjectConfirmation>` block of the SAML assertion is generated. When the assertion is consumed by a downstream Web Service, the information contained in the

<SubjectConfirmation> block can be used to authenticate either the end-user that authenticated to the Enterprise Gateway, or the issuer of the assertion, depending on what is configured.

The following is a typical <SubjectConfirmation> block:

```
<saml:SubjectConfirmation>
  <saml:ConfirmationMethod>
    urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
  </saml:ConfirmationMethod>
  <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
    <dsig:X509Data>
      <dsig:X509SubjectName>CN=oracle</dsig:X509SubjectName>
      <dsig:X509Certificate>
        MIICmzCCAY . . . . . mB9CJEw4Q=
      </dsig:X509Certificate>
    </dsig:X509Data>
  </dsig:KeyInfo>
</saml:SubjectConfirmation>
</saml:SubjectConfirmation>
```

You must configure the following fields on the **Subject Confirmation** tab:

Method:

The selected value determines the value of the <ConfirmationMethod> element. The following table shows the available methods, their meanings, and their respective values in the <ConfirmationMethod> element:

Method	Meaning	Value
Holder Of Key	A <SubjectConfirmation> is inserted into the SAML request. The <SubjectConfirmation> contains a <dsig:KeyInfo> section with the certificate of the user selected to sign the SAML request. The user selected to sign the SAML request must be the authenticated subject (authentication.subject.id). Select the Certificate is included if the signer's certificate is to be included in the SubjectConfirmation block. Alternatively, select the Only key name is included radio button if only the key name is to be included. Select the user whose private key is used to sign part of the message in the User Name drop-down list on the Sign Request tab.	urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
Bearer	A <SubjectConfirmation> is inserted into the SAML request.	urn:oasis:names:tc:SAML:1.0:cm:bearer
SAML Artifact	A <SubjectConfirmation> is inserted into the SAML request.	urn:oasis:names:tc:SAML:1.0:cm:artifact
Sender Vouches	A <SubjectConfirmation> is inserted into the SAML request.	urn:oasis:names:tc:SAML:1.0:cm:sender-vouches

Method	Meaning	Value
	ted into the SAML request. The SAML request must be signed by a user.	m:bearer

If the **Method** field is left blank, no `<ConfirmationMethod>` block is inserted into the assertion.

Include Certificate:

Select this option if you wish to include the SAML subject's certificate in the `<KeyInfo>` section of the `<SubjectConfirmation>` block.

Include Key Name:

Alternatively, if you do not want to include the certificate, you can select this option to only include the key name in the `<KeyInfo>` section.

Attributes:

You can list a number of user attributes to include in the SAML attribute assertion that is generated by the Enterprise Gateway. If no attributes are explicitly listed in this section, the Enterprise Gateway inserts all attributes associated with the user (all user attributes in the `attribute.lookup.list` message attribute) in the assertion.

To add a specific attribute to the SAML attribute assertion, click the **Add** button. A user attribute can be configured using the **Attribute Lookup** dialog.

Enter the name of the attribute that is added to the assertion in the **Attribute Name** field. Enter the namespace that is associated with this attribute in the **Namespace** field.

You can edit and remove previously configured attributes using the **Edit** and **Remove** buttons.

Response Configuration

The fields on this tab relate to the SAML Response returned from the SAML PDP. The following fields are available:

SOAP Actor/Role:

If the SAML response from the PDP contains a SAML attribute assertion, the Enterprise Gateway can extract it from the response and insert it into the downstream message. The SAML assertion is inserted into the WS-Security block identified by the specified SOAP actor/role.

Drift Time:

The SAML request to the PDP is timestamped by the Enterprise Gateway. To account for differences in the times on the machines running the Enterprise Gateway and the SAML PDP the specified time is subtracted from the time at which the Enterprise Gateway generates the SAML request.

Retrieve Attribute from Message

Overview

The **Retrieve from Message** filter uses XPath expressions to extract the value of an XML element or attribute from the message and set it to an internal message attribute. It is also possible for the XPath expression to return a NodeList, and for the NodeList to be set to the specified message attribute.

Configuration

The following fields are available on the **Retrieve from Message** filter configuration screen:

Name:

Enter a name for this filter here.

Attribute Location:

Configure an XPath expression to retrieve the desired content.

Click the **Add** button to add an XPath expression. Existing expressions can be added and removed by clicking the **Edit** and **Remove** buttons respectively.

Extract the Content of the Node:

If this option is selected the content of the XML element or attribute in the message will be extracted and set to the selected message attribute.

Serialize All Nodes in the NodeList:

Use this option to select a NodeList to set to the message attribute. This option is useful for extracting <Signature>, <Security>, and <UsernameToken> blocks, as well as proprietary blocks of XML from messages.

Attribute ID:

The Enterprise Gateway will set the value of the message attribute selected here to the value extracted from the message. It is possible to enter a user-defined message attribute here as well.

Retrieve Attribute from User Store

Overview

The **User Store** stores a user's profile, including attributes relating to that user. After a user has successfully authenticated to the Enterprise Gateway, the **Retrieve From User Store** filter can retrieve attributes belonging to that user from the **User Store**. All attributes that are retrieved are set to the `attribute.lookup.list`.

Configuration

The following fields are available on the **Retrieve From User Store** filter configuration screen:

Name:

Enter an appropriate name for this filter.

User ID:

Select or enter the name of the message attribute that contains the name of the user to look up in the **User Store**. For example, if the user name is stored as `admin`, you must select the message attribute containing the value `admin`. The Enterprise Gateway then looks up the user the **User Store** using this name.

Attributes:

Enter the list of attributes that the Enterprise Gateway should retrieve if it successfully looks up the user identified by the message attribute specified in the **User ID** field. All attribute values are stored in the `attribute.lookup.list` message attribute.

If no user attributes are specified, the Enterprise Gateway retrieves all the user's registered attributes and sets them to the `attribute.lookup.list` attribute.

You can add attributes by selecting the **Add** button. Similarly, you can edit and remove existing attributes by selecting the **Edit** and **Remove** buttons.

Insert SAML Attribute Assertion

Overview

A Security Assertion Markup Language (SAML) attribute assertion contains information about a user in the form of a series of attributes. Having collated a certain amount of information about a user, the Enterprise Gateway can generate a SAML attribute assertion, and insert it into the downstream message.

A *SAML Attribute* (see example below) is generated for each entry in the `attribute.lookup.list` attribute. Other filters from the **Attributes** filter group can be used to insert user attributes into the `attribute.lookup.list` attribute.

It may be useful to refer to the following example of a SAML attribute assertion when configuring this filter.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance">
  <soap:Header>
    <wsse:Security>
      <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
        ID="Id-0000010a3c4ff12c-0000000000000002"
        IssueInstant="2006-03-27T15:26:12Z" Version="2.0">
        <saml:Issuer Format="urn:oasis ... WindowsDomainQualifiedName">
          TestCA
        </saml:Issuer>
        <saml:Subject>
          <saml:NameIdentifier Format="urn:oasis ... WindowsDomainQualifiedName">
            TestUser
          </saml:NameIdentifier>
        </saml:Subject>
        <saml:Conditions NotBefore="2005-03-27T15:20:40Z"
          NotOnOrAfter="2028-03-27T17:20:40Z"/>
        <saml:AttributeStatement>
          <saml:Attribute Name="role" NameFormat="http://www.oracle.com">
            <saml:AttributeValue>admin</saml:AttributeValue>
          </saml:Attribute>
          <saml:Attribute Name="email" NameFormat="http://www.oracle.com">
            <saml:AttributeValue>joe@oracle.com</saml:AttributeValue>
          </saml:Attribute>
          <saml:Attribute Name="dept" NameFormat="">
            <saml:AttributeValue>engineering</saml:AttributeValue>
          </saml:Attribute>
        </saml:AttributeStatement>
      </saml:Assertion>
    </wsse:Security>
  </soap:Header>

  <soap:Body>
    <product>
      <name>Enterprise Gateway</name>
      <company>Oracle</company>
      <description>Web Services Security</description>
    </product>
  </soap:Body>
</soap:Envelope>
```

General Configuration

Configure the following field:

Name:

Enter an appropriate name for the filter.

Assertion Details

Configure the following fields on the **Assertion Details** tab:

Issuer Name:

Select the certificate containing the Distinguished Name (DName) that you want to use as the Issuer of the SAML assertion. This DName is included in the SAML assertion as the value of the `Issuer` attribute of the `<saml:Assertion>` element. For an example, see the sample SAML assertion above.

Expire In:

Specify the lifetime of the assertion in this field. The lifetime of the assertion lasts from the time of insertion until the specified amount of time has elapsed.

Drift Time:

The **Drift Time** is used to account for differences in the clock times of the machine hosting the Enterprise Gateway (that generate the assertion) and the machines that consume the assertion. The specified time is subtracted from the time at which the Enterprise Gateway generates the assertion.

SAML Version:

You can create SAML 1.0, 1.1, and 2.0 attribute assertions. Select the appropriate version from the drop-down list.

Important Note:

SAML 1.0 recommends the use of the <http://www.w3.org/TR/2001/REC-xml-c14n-20010315> XML Signature Canonicalization algorithm. When inserting signed SAML 1.0 assertions into XML documents, it is quite likely that subsequent signature verification of these assertions will fail. This is due to the side effect of the algorithm including inherited namespaces into canonical XML calculations of the inserted SAML assertion that were not present when the assertion was generated.

For this reason, Oracle recommend that SAML 1.1 or 2.0 is used when signing assertions as they both uses the exclusive canonical algorithm <http://www.w3.org/2001/10/xml-exc-c14n#>, which safeguards inserted assertions from such changes of context in the XML document. Please see section 5.4.2 of the [oasis-sstc-saml-core-1.0.pdf](#) and section 5.4.2 of [sstc-saml-core-1.1.pdf](#) documents, both of which are available at <http://www.oasis-open.org>.

Assertion Location

The options on the **Assertion Location** tab specify where the SAML assertion is inserted in the message. By default, the SAML assertion is added to the WS-Security block with the current SOAP actor/role. The following options are available:

Append to Root or SOAP Header:

Appends the SAML assertion to the message root for a non-SOAP XML message, or to the SOAP Header for a SOAP message. For example, this option may be suitable for cases where this filter may process SOAP XML messages or non-SOAP XML messages.

Add to WS-Security Block with SOAP Actor/Role:

Adds the SAML assertion to the WS-Security block with the specified SOAP actor (SOAP 1.0) or role (SOAP 1.1). By default, the assertion is added with the current SOAP actor/role only, which means the WS-Security block with no actor. You can select a specific SOAP actor/role when available from the drop-down list.

XPath Location:

If you wish to insert the SAML assertion at an arbitrary location in the message, you can use an XPath expression to

specify the exact location in the message. You can select XPath expressions from the drop-down list. The default is the `First WSSE Security Element`, which has an XPath expression of `//wsse:Security`. You can add, edit, or remove expressions by clicking the relevant button. For more details, see the [Configuring XPath Expressions](#) topic.

You can specify exactly how the SAML assertion is inserted using the following options:

- **Append to node returned by XPath expression** (the default)
- **Insert before node returned by XPath expression**
- **Replace node returned by XPath expression**

Insert into Message Attribute:

Specify a message attribute to store the SAML assertion from the drop-down list (for example, `saml.assertion`). Alternatively, you can also enter a custom message attribute in this field (for example, `my.test.assertion`). The SAML assertion can then be accessed downstream in the policy circuit.

Subject Confirmation Method

The settings on the **Subject Confirmation Method** tab determine how the `<SubjectConfirmation>` block of the SAML assertion is generated. When the assertion is consumed by a downstream Web Service, the information contained in the `<SubjectConfirmation>` block can be used to authenticate either the end-user that authenticated to the Enterprise Gateway, or the issuer of the assertion, depending on what is configured.

The following is a typical `<SubjectConfirmation>` block:

```
<saml:SubjectConfirmation>
  <saml:ConfirmationMethod>
    urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
  </saml:ConfirmationMethod>
  <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
    <dsig:X509Data>
      <dsig:X509SubjectName>CN=oracle</dsig:X509SubjectName>
      <dsig:X509Certificate>
        MIICmzCCAY . . . . . mB9CJEw4Q=
      </dsig:X509Certificate>
    </dsig:X509Data>
  </dsig:KeyInfo>
</saml:SubjectConfirmation>
```

The following configuration fields are available on the **Subject Confirmation Method** tab:

Method:

The value selected here determines the value of the `<ConfirmationMethod>` element. The following table shows the available methods, their meanings, and their respective values in the `<ConfirmationMethod>` element:

Method	Meaning	Value
Holder Of Key	The Enterprise Gateway includes the key used to prove that the Enterprise Gateway is the holder of the key, or includes a reference to the key.	urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
Bearer	The subject of the assertion is the	urn:oasis:names:tc:SAML:1.0:cm

Method	Meaning	Value
	bearer of the assertion.	m:bearer
SAML Artifact	The subject of the assertion is the user that presented a SAML Artifact to the Enterprise Gateway.	urn:oasis:names:tc:SAML:1.0:cm:artifact
Sender Vouches	Use this confirmation method to assert that the Enterprise Gateway is acting on behalf of the authenticated end-user. No other information relating to the context of the assertion is sent. It is recommended that both the assertion and the SOAP Body must be signed if this option is selected. These message parts can be signed by using the Sign Message filter.	urn:oasis:names:tc:SAML:1.0:cm:bearer

Note that you can also leave the **Method** field blank, in which case no <ConfirmationMethod> block is inserted into the assertion.

Holder-of-Key Configuration:

When you select **Holder-of-Key** as the SAML subject confirmation in the **Method** field, you must configure how information about the key is to be included in the message. There are a number of configuration options available depending on whether the key is a symmetric or asymmetric key.

Asymmetric Key:

If you want to use an asymmetric key as proof that the Enterprise Gateway is the holder-of-key entity, you must select the **Asymmetric Key** radio button, and then configure the following fields on the **Asymmetric** tab:

- **Private Key from Certificate Store:**
If you want to select a key that is stored in the Certificate Store, select this option and click the **Signing Key** button. On the **Select Certificate** screen, select the box next to the certificate that is associated with the key that you want to use.
- **Private Key from Message Attribute:**
Alternatively, the key may have already been used by a previous filter in the circuit, for example to sign a part of the message. In this case, the key is stored in a message attribute. You can specify this message attribute in this field.

Symmetric Key:

If you want to use a symmetric key as proof that the Enterprise Gateway is the holder of key, select the **Symmetric Key** radio button, and configure the fields on the **Symmetric** tab:

- **Generate Symmetric Key:**
If you select this option, the Enterprise Gateway generates a symmetric key, which is included in the message before it is sent to the client.
- **Symmetric Key in Message Attribute:**
If a previous filter (for example, a **Sign Message** filter) has already used a symmetric key, you can to reuse this key as proof that the Enterprise Gateway is the holder-of-key entity. You must enter the name of the message attribute in the field provided.
- **Encrypt using Certificate from Certificate Store:**
When a symmetric key is used, you must assume that the recipient has no prior knowledge of this key. It must,

therefore, be included in the message so that the recipient can validate the key. To avoid meet-in-the-middle style attacks, where a hacker could eavesdrop on the communication channel between the Enterprise Gateway and the recipient and gain access to the symmetric key, the key must be encrypted so that only the recipient can decrypt the key. One way of doing this is to select the recipient's certificate from the Certificate Store. By encrypting the symmetric key with the public in the recipient's certificate, the key can only be decrypted by the recipient's private key, to which only the recipient has access. Select the **Signing Key** button and then select the recipient's certificate on the **Select Certificate** dialog.

- **Encrypt using Certificate from Message Attribute:**
Alternatively, if the recipient's certificate has already been used (perhaps to encrypt part of the message) this certificate is stored in a message attribute. You can enter the message attribute in this field.
- **Symmetric Key Length:**
Enter the length (in bits) of the symmetric key to use.
- **Key Wrap Algorithm:**
Select the algorithm to use to encrypt (*wrap*) the symmetric key.

Key Info:

The **Key Info** tab must be configured regardless of whether you have elected to use symmetric or asymmetric keys. It determines how the key is included in the message. The following options are available:

- **Do Not Include Key Info:**
Select this option if you do not wish to include a `<KeyInfo>` section in the SAML assertion.
- **Embed Public Key Information:**
If this option is selected, details about the key are included in a `<KeyInfo>` block in the message. You can include the full certificate, expand the public key, include the distinguished name, and include a key name in the `<KeyInfo>` block by selecting the appropriate boxes. When selecting the **Include Key Name** field, you must enter a name in the **Value** field, and select the **Text Value** or **Distinguished Name Attribute** radio button, depending on the source of the key name.
- **Put Certificate in Attachment:**
Select this option to add the certificate as an attachment to the message. The certificate is then referenced from the `<KeyInfo>` block.
- **Security Token Reference:**
The Security Token Reference (STR) provides a way to refer to a key contained in a SOAP message from another part of the message. It is often used in cases where different security blocks in a message use the same key material, and it is considered an overhead to include the key more than once in the message.
When this option is selected, a `<wsse:SecurityTokenReference>` element is inserted into the `<KeyInfo>` block. It references the key material using a URI to point to the key material and a `ValueType` attribute to indicate the type of reference used. For example, if the STR refers to an encrypted key, you should select `EncryptedKey` from the drop-down list, whereas if it refers to a `BinarySecurityToken`, select `X509v3` from the dropdown. Other options are available to enable more specific security requirements.

Advanced

The settings on the **Advanced** tab include the following fields.

Select Required Layout Type:

WS-Policy and SOAP Message Security define a set of rules that determine the layout of security elements that appear in the WS-Security header in a SOAP message. The SAML assertion is inserted into the WS-Security header according to the layout option selected here. The available options correspond to the WS-Policy Layout assertions of `Strict`, `Lax`, `LaxTimestampFirst`, and `LaxTimestampLast`.

Indent:

Select this method to ensure that the generated signature is properly indented.

Security Token Reference:

The generated SAML attribute assertion can be encapsulated in a <SecurityTokenReference> block. The following example demonstrates this:

```
<soap:Header>
  <wsse:Security
    xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"
    soap:actor="oracle">
    <wsse:SecurityTokenReference>
      <wsse:Embedded>
        <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
          ID="Id-0000010a3c4ff12c-0000000000000002"
          IssueInstant="2006-03-27T15:26:12Z" Version="2.0">
          <saml:Issuer Format="urn:oasis ... WindowsDomainQualifiedName">
            TestCA
          </saml:Issuer>
          <saml:Subject>
            <saml:NameID Format="urn:oasis ... WindowsDomainQualifiedName">
              TestUser
            </saml:NameID>
          </saml:Subject>
          <saml:Conditions NotBefore="2005-03-27T15:20:40Z"
            NotOnOrAfter="2028-03-27T17:20:40Z"/>
          <saml:AttributeStatement>
            <saml:Attribute Name="role" NameFormat="http://www.oracle.com">
              <saml:AttributeValue>admin</saml:AttributeValue>
            </saml:Attribute>
            <saml:Attribute Name="email" NameFormat="http://www.oracle.com">
              <saml:AttributeValue>joe@oracle.com</saml:AttributeValue>
            </saml:Attribute>
            <saml:Attribute Name="attrib1" NameFormat="">
              <saml:AttributeValue xsi:nil="true"/>
              <saml:AttributeValue>value1</saml:AttributeValue>
            </saml:Attribute>
          </saml:AttributeStatement>
        </saml:Assertion>
      </wsse:Embedded>
    </wsse:SecurityTokenReference>
  </wsse:Security>
</soap:Header>
```

To add the SAML assertion to a <SecurityTokenReference> block like in this example, select the **Embed SAML assertion within Security Token Reference** option. Otherwise, select **No Security Token Reference**.

Extract REST Request Attributes

Overview

This filter extracts the values of query string parameters and/or HTTP headers from a REST request and stores them in separate message attributes. The REST request can be an HTTP GET or POST request. This filter is found in the **Attributes** category in the Policy Studio. For details on how to create a REST request, see the [Create REST Request](#) filter.

Example REST Request

The following example shows an incoming REST request with query string and HTTP headers:

```
POST /services?name=Niall&location=Dublin&location=Pembroke%20St HTTP/1.1
Host: mail.google.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-GB; rv:1.9.2.15)
Gecko/20110303 Firefox/3.6.15
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
```

Using this example, the **Extract REST Request Attributes** filter generates and populates the following message attributes:

```
http.header.Host = mail.google.com
http.header.User-Agent = Mozilla/5.0 (Windows; U; Windows NT 6.1; en-GB; rv:1.9.2.15)
Gecko/20110303 Firefox/3.6.15
http.header.Accept = text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
http.header.Accept-Language = en-gb,en;q=0.5
http.header.Accept-Encoding = gzip,deflate
http.header.Accept-Charset = ISO-8859-1,utf-8;q=0.7,*;q=0.7

http.querystring.name = Niall
http.querystring.location.1 = Dublin
http.querystring.location.2 = Pembroke St
```

Note:

For multi-valued query string parameters (for example, `location`), each value is given an incremental index. For example, the multi-valued `location` parameter results in the creation of the `http.querystring.location.1` and `http.querystring.location.2` message attributes.

The purpose of this filter is to extract all parameters from an incoming REST request and store them in separate message attributes so that they can be validated easily, without needing to iterate through the set of `http.headers`.

Configuration

Configure the following fields on the **Extract REST Request Attributes** screen:

Name:

Enter an appropriate name for this filter.

Request Querystring:

Select whether to extract the values of query string parameters from an HTTP POST or GET request. These are simple name-value pairs (for example, `Name=Joe Bloggs`). This setting is selected by default.

HTTP Headers:

Select whether to extract the values of HTTP headers from an HTTP POST or GET request. This setting is selected by default.

Extract WSS Timestamp

Overview

You can use the **Extract WSS Timestamp** filter to extract a WSS Header Timestamp from a message. The timestamp is stored in a specified message attribute so that it can be processed later in a circuit. This filter requires the WSS Header block to have been extracted previously. For more details, see the [Extract WSS Header](#) filter.

Typically, the **Validate Timestamp** filter is used to retrieve the timestamp from the specified message attribute and validate it. The **Validate Timestamp** filter is available from the **Content Filtering** filter category. For more details, see the [Validate Timestamp](#) filter.

Configuration

Configure the following fields on the **Extract WSS Timestamp** filter configuration screen:

Name:

Enter an appropriate name for this filter.

Message Attribute to Contain the Timestamp:

When the Enterprise Gateway extracts the WSS Header Timestamp from the message at runtime, it stores the timestamp in the specified message attribute. If you wish to validate the timestamp later in the circuit, you *must* specify this message attribute in the configuration screen for the **Validate Timestamp** filter.

Extract WSS UsernameToken

Overview

You can use the **Extract WSS Username Token** filter to extract a WS-Security Username Token from a message if it exists. The extracted Username Token is stored in the `wss.usernameToken` message attribute.

If you want to process the Username Token later in the circuit, you can specify this message attribute in the configuration screen for the processing filter. For example, if you want to sign the Username Token, you can simply specify the `wss.usernameToken` message attribute in the **What to Sign** section of the **Sign Message** filter. Open the **Message Attribute** tab on the **What to Sign** screen, and specify this attribute to sign the Username Token.

Configuration

Configure the following field on the **Extract WSS Username Token** filter configuration screen:

Name:

Enter an appropriate name for the filter. Remember that the WS-Security Username Token is stored in the `wss.usernameToken` message attribute.

Extract WSS Header

Overview

The **Extract WSS Header** filter extracts a WS-Security <Header> block from a message. The extracted security header is stored in the `authentication.ws.wsblockinfo` message attribute.

If you want to process this security header later in the circuit, you can specify this message attribute in the configuration screen for the specific processing filter. For example, if you want to sign the security header, you can specify the `authentication.ws.wsblockinfo` message attribute in the **What to Sign** section of the **Sign Message** filter. Open the **Message Attribute** tab on the **What to Sign** screen, and specify this attribute to sign the security header.

Configuration

Configure the following fields on the **Extract WSS Header** filter configuration screen:

Name:

Enter an intuitive name for this filter (for example, **Extract Current Actor WSS Header**).

Actor or Role:

Specify the name of the SOAP Actor or Role of the WS-Security header that you want to extract. Remember, the WS-Security header is stored in the `authentication.ws.wsblockinfo` message attribute.

Remove enclosing WS-Security element:

This option removes the enclosing `<wsse:Security>` element from the message.

Authentication Repository

Overview

The Enterprise Gateway supports a wide range of common authentication schemes, including SSL, XML Signatures, WS-Security Username tokens, and HTTP Authentication. With SSL, the client authenticates to the Enterprise Gateway using a client certificate. With XML Signatures, the client is authenticated by validating the signature contained within the XML message. However, when the Enterprise Gateway attempts to authenticate a client using a username and password (for example, WS-Security Username tokens and HTTP Authentication), it must compare the username and password presented by the client to those stored in the Oracle **Authentication Repository**.

The **Authentication Repository** acts as a repository for **Users**. **Users** serve many roles in the Enterprise Gateway. For example, clients whose username and password combinations are stored in the **Authentication Repository** can authenticate to the Enterprise Gateway using that username and password combination. For more information on **Users**, see the [Enterprise Gateway Users](#) tutorial.

The **Authentication Repository** can be maintained in the Enterprise Gateway's local configuration store, in an LDAP directory, or in a range of third-party Identity Management products and services. When a user has been successfully authenticated against one of these repositories, the Enterprise Gateway can use any one of that user's stored attributes (for example, DName, email address, username) to authorize that same user in a subsequent Authorization Filter.

For example, this *credential mapping* is useful in cases where your client-base uses username-password combinations for authentication (*authentication attributes*), yet their access rights must be looked up in an authorization server using the client's DName (*authorization attribute*). In this way, the client possesses a single *virtual identity* within the Enterprise Gateway. The client can use one identity for authentication, and another for authorization, yet the Enterprise Gateway sees both identities as representing the same client.

You can add a new repository on the **External Connections** tab by right-clicking the appropriate node (for example, **Database Repositories**), and selecting **Add a new Repository**. Similarly, you can edit an existing repository by right-clicking the repository node (for example, the default **Local User Store**), and selecting **Edit Repository**. Repositories added on the **External Connections** tab are available for reuse by multiple filters.

Local Repositories

The **Authentication Repository** can be maintained in the same database that the Enterprise Gateway uses to store all its configuration information. To edit the default user store, select **Local Repositories -> Local User Store -> Edit Repository**. Alternatively, to create a new user store, select **Local Repositories -> Add a new Repository**.

You can enter an appropriate name for the repository in the **Repository Name** field. The **Authorization Attribute Format** field enables administrators to specify whether to use the client's **X.509 Distinguished Name** or **User Name** in subsequent Authorization Filters. If **User Name** is selected, the user name used by the client to authenticate to the Enterprise Gateway is used in any configured Authorization filters. If **X.509 Distinguished Name** is selected, the X.509 DName stored by the Enterprise Gateway for that user is used for subsequent authorization.

For example, if the administrator selected **User Name** from the **Authorization Attribute Format** drop-down list, `admin` (the **User Name** field) is used for authorization. Alternatively, if **X.509 Distinguished Name** is selected, the X.509 DName is used for authorization (for example, `O=Company, OU=comp, EMAIL=emp@company.com, CN=emp`).

For more information on adding and configuring users to the **Authentication Repository**, see the [Users](#) tutorial.

LDAP Repositories

In cases where an organization stores user profiles in an LDAP directory, it does not make sense to re-enter those profiles into the default Enterprise Gateway store. Rather, the Enterprise Gateway can leverage an existing LDAP directory by querying it for user profile data. If a user's profile can be retrieved and you can bind to the LDAP directory as that user, the user is authenticated.

When a filter is configured to authenticate a user against an LDAP repository using a username and password combination, the following steps occur:

1. A pooled LDAP connection with details corresponding to the repository selected in the **LDAP Directory** field is retrieved.
2. A search filter (for example, `(&(objectClass={User}) (sAMAccountName={c05vc}))`) is run using the retrieved connection. Attributes configured in the **Login Authentication Attribute** and **Authorization Attribute** fields are also retrieved in the same search operation.
For example, if **Distinguished Name** is selected from the drop-down list, the user's DName is retrieved from the LDAP directory. The user's DName uniquely identifies the user in the LDAP directory, and is used to bind to the directory so that the user's password can be verified. The attribute specified in the **Login Authentication Attribute** field is used for the bind operation when you bind as any user. The value of the LDAP attribute specified in the **Authorization Attribute** field is stored in the `authentication.subject.id` and may be used by subsequent filters in the circuit (for example, by authorization filters to authorize the authenticated user).
3. If no results are returned from the search, the user has not been found in the directory. It is important that the administrator user configured on the **Configure LDAP Server** screen has the ability to see the user you are attempting to authenticate.
4. If multiple results (users) are returned from the search, an attempt is made to bind to the directory using each **Login Authentication Attribute** value retrieved from the search, together with the password from the message.
5. If more than one user is authenticated correctly, an error is returned because you only want to authenticate a single user.
6. If no user is authenticated, an error is returned.
7. If a single user's **Login Authentication Attribute** value and password binds successfully to the directory, authentication has succeeded.
8. Any successful bind is immediately closed.

To create a new LDAP repository, right click **LDAP Repositories** and select **Add a new Repository**. The details entered on the **Authentication Repository** screen depend on the type of LDAP directory that you are using. The Policy Studio has default entries for some of the more common LDAP directories, which are available from the drop-down lists on the screen. However, you can also connect to alternative LDAP directories.

The following subsections demonstrate how to configure this screen for typical user searches on three common LDAP servers:

- Oracle Directory Server
- Microsoft Active Directory Server
- IBM Directory Server

Oracle Directory Server

To configure the **Authentication Repository** dialog for Oracle Directory Server, perform the following steps:

1. Enter a suitable name for this user store in the **Repository Name** field.
2. Click **Add/Edit** to add details of your Oracle Directory Server.

The **User Search Conditions** section instructs the Enterprise Gateway to search the LDAP tree according to certain conditions:

- **Base Criteria:**
This specifies where the Enterprise Gateway should begin searching the LDAP directory.
- **User Class:**

This is the name given by the particular LDAP directory to the *User* class. For Oracle Directory Server, select 'inet-orgperson' LDAP Class from the drop-down list.

- **User Search Attribute:**
The value entered depends on the type of LDAP directory to which you are connecting. When a user is stored in an LDAP directory, a number of user *attributes* are stored along with that user. One of these attributes corresponds to the user name presented by the client for authentication. However, different LDAP directories use different names for that user attribute. For Oracle Directory Server, select `cn` from the drop-down list.
- **Allow Blank Passwords:**
Select this box to allow the use of blank passwords.

In the next section, you must specify the following:

- **Login Authentication Attribute:**
In an LDAP directory tree, there must be one user attribute that uniquely distinguishes any one user from all the others. This is usually the Distinguished Name of the user, however this is called different things in different LDAP directories. In Oracle Directory Server, the Distinguished name is referred to as the *entrydn*, so select **Entry Domain Name** in this drop-down list to uniquely identify the client authenticating to the Enterprise Gateway.
- **Authorization Attribute:**
When the client has been successfully authenticated, you can use any one of that user's stored attributes in a subsequent Authorization Filter. In this case, you want to use the user's Distinguished Name for an Authorization filter, so enter `entrydn` in the text box. However, you can enter any user attribute as long as the subsequent Authorization filter supports it. The value of the LDAP attribute specified is stored in the `authentication.subject.id` Oracle message attribute.
- **Authorization Attribute Format:**
Because any user attribute can be specified in the **Authorization Attribute** above, it is necessary to inform the Enterprise Gateway of the type of this attribute. This information is used internally by the Enterprise Gateway in subsequent Authorization filters. Select **X.509 Distinguished Name** from the drop-down list.

Microsoft Active Directory Server

This subsection describes how to configure the **Authentication Repository** dialog for Microsoft Active Directory Server. It is important to note how the values entered here differ from those entered when interfacing to the Oracle Directory Server:

- **Repository Name:**
Enter a suitable name for this search.
- **LDAP Directory:**
Click **Add/Edit** to add details of your Active Directory Server.

The **User Search Conditions** instruct the Enterprise Gateway to search the LDAP tree according to certain criteria. The values entered/selected are different from those selected for Oracle Directory Server, because MS Active Directory Server uses different attributes and classes to Oracle Directory Server:

- **Base Criteria:**
The base criteria specify the base object under which to search for the user's profile.
- **User Class:**
In Active Directory Server, the user class is simply called *User*, so select 'User' LDAP Class from this drop-down list.
- **User Search Attribute:**
This field specifies the name of the user attribute whose value corresponds to the user name entered by the client during a successful authentication process. With Active Directory Server, this attribute is called *givenName* and it represents the name of the user. Enter `givenName` in this drop-down list.
- **Login Authentication Attribute:**

Enter the name of the user attribute that uniquely identifies the user in the LDAP directory. This attribute is the Distinguished Name and is called *distinguishedName* in Active Directory Server. Select *Distinguished Name* from the drop-down list to uniquely identify the user. The Enterprise Gateway authenticates the username and password presented by the client against the values stored for the user identified in this field.

- **Authorization Attribute:**
When the client has been successfully authenticated, the Enterprise Gateway can use any of that user's stored attributes in subsequent Authorization filters. Because most Authorization filters require a Distinguished Name, enter *Distinguished Name* in the text box. However, any user attribute could be entered here, as long as the subsequent Authorization filter supports it.
- **Authorization Attribute Format:**
The Enterprise Gateway needs to know the format of the **Authorization Attribute**. Select *X.509 Distinguished Name* from the drop-down list.

IBM Directory Server

The configuration details for IBM Directory Server deserve a special mention as an example of a directory server that does not return a full Distinguished Name (DName) as the result of a standard LDAP user search. Instead, it returns a *contextualized* DName, which is relative to the specified **Base Criteria**. In such cases, the Enterprise Gateway can build up the full DName by combining the **Base Criteria** and the returned name. The following example shows how this works in practice.

If *C=IE* is specified as the **Base Criteria**, the IBM Directory Server returns *CN=niatl, OU=Dev*, instead of the full DName, which is *C=IE, CN=niatl, OU=Dev*. To enable the Enterprise Gateway to do this, leave the **Login Authentication Attribute** field blank. The Enterprise Gateway then automatically concatenates the specified **Base Criteria** (*C=IE*) with the contextualized DName returned from the directory server (*CN=niatl, OU=Dev*) to obtain the fully qualified DName (*C=IE, CN=niatl, OU=Dev*).

You can also leave the **Authorization Attribute** field blank, which enables the Enterprise Gateway to automatically use the fully qualified DName for subsequent Authorization Filters. You should select *X.509 Distinguished Name* from the **Authorization Attribute Format** drop-down list.

CA SiteMinder Repositories

In cases where user profiles have been stored in an existing SiteMinder server, the Enterprise Gateway can query SiteMinder to authenticate users.

To authenticate users against a CA SiteMinder repository, right-click **CA SiteMinder Repositories**, and select **Add a new Repository**. Complete the following fields on the **Authentication Repository** dialog:

Repository Name:

Enter a suitable name for this repository.

Agent Name:

Select a previously configured SiteMinder Agent name from the drop-down list. Click **Add** to register a new agent. Complete the following fields in the **SiteMinder Connection Details** dialog:

- **Agent Name:**
Enter the name of the agent to connect to SiteMinder in the **Agent Name** field. This name **must** correspond with the name of an agent that was previously configured in the **Policy Server**.
- **Agent Configuration Object:**
The name entered must match the name of the Agent Configuration Object (ACO) configured in the Policy Server. The Enterprise Gateway currently does not support any of the features represented by the ACO parameters except for the *PersistentIPCheck* setting. For example, the Enterprise Gateway disregards the *DefaultAgent* parameter and uses the agent value it collects separately during agent registration.
When the *PersistentIPCheck* ACO parameter is set to *yes*, it instructs the Enterprise Gateway to compare the IP address from the last request (stored in a persistent cookie) with the IP address in the current request to see if they match. If the IP addresses do not match, the Enterprise Gateway rejects the request. If this parameter is set to

no, this check is disabled.

- **Connection Details:**

For more information on configuring this section, please refer to the instructions in the **SiteMinder Connection Details** section of the [SiteMinder Certificate Authentication](#) help page.

Resource:

Enter the name of the protected resource for which the user must be authenticated.

Alternatively, you can enter a property representing a message attribute, which is looked up and expanded to a value at runtime. Properties have the following format:

```
${message.attribute}
```

For example, to specify the original path on which the request was received by the Enterprise Gateway as the resource, enter the following property:

```
${http.request.uri}
```

Action:

The user must be authenticated for a specific action on the protected resource. By default, this action is taken from the HTTP verb used in the incoming request. You can use the following property to get the HTTP verb:

```
${http.request.verb}<br/>
```

Alternatively, you can enter any user-specified value.

Create Single Sign-On Token:

When this option is selected, SiteMinder generates a single sign-on token as part of the authentication event and returns it to the Enterprise Gateway. This is then inserted into the downstream message for re-use later, either by another instance of the Enterprise Gateway running the **SiteMinder Session Validation** filter, or by another SiteMinder-aware agent.

Put Token in Message Attribute:

Enter the name of the message attribute where you wish to store the single sign-on token. By default, the token is stored in the `siteminder.session` attribute. Please refer to the [Message Attribute Reference](#) for a complete list of available message attributes.

Database Repositories

The Enterprise Gateway can store its **Authentication Repository** in an external database. This option makes sense when an organization already has a silo of user profiles stored in the database and does not want to duplicate this store within the Enterprise Gateway's local configuration storage.

To authenticate users against a database repository, right-click **Database Repositories**, and select **Add a new Repository**. Complete the following fields on the **Authentication Repository** dialog:

Repository Name:

Enter an appropriate name for the database in the **Repository Name** field.

Database:

There are two basic configuration items required to retrieve a user's profile from the database:

- **Database Location:**
You can configure connection details for the database by clicking **Add**, and completing the **Database Connection** dialog. For details on configuring the fields on this dialog, see the [Database Connection](#) topic. You can edit or remove previously configured database connections by selecting them in the drop-down list and clicking **Edit** or **Delete**.
- **Database Query:**
The **Database Query** retrieves a specific user's profile from the database to enable the Enterprise Gateway to authenticate them. Having successfully authenticated the user, you can select an attribute of this user to use for the authorization filter later in the policy. The **Database Query** can take the form of an SQL statement, stored procedure, or function call. For details on how to configure the **Database Query**, see the [Database Query](#) topic.

Format Password Received From Client:

If the user sends up a clear-text password to the Enterprise Gateway, but that user's password is stored in a hashed format in the database, it is the Enterprise Gateway must hash the password before performing the authentication step.

- **Hash Client Password:**
Depending on whether you wish to hash the user's submitted password, select the appropriate radio button.
- **Hash Format:**
If you have selected to hash the client's password, the Enterprise Gateway needs to know the format of the hashed password. The most typical formats are available from the drop-down list, however, you can also enter another format. Formats should be entered in terms of message attributes. The following formats are available from the **Hash Format** drop-down list. The first option combines the username, authentication realm, and password respectively. This combination is then hashed. The second option simply creates a hash of the user's password.

```
${authentication.subject.id}:${authentication.subject.realm}:${authentication.subject.password}
${authentication.subject.password}
```

- **Hash Algorithm:**
Select either **MD5** or **SHA1** to use as the digest algorithm to use when creating the hash.

Query Result Processing:

This section enables you to provide the Enterprise Gateway with some meta information about the result returned by the **Database Query** configured earlier on this screen. It enables allows you to identify the name of the database table column or row that contains the user's password, and also the name of the column or row that contains the attribute that is to be used for the authorization filter.

- **Password Column:**
Specify the name of the database table column that contains the user's password. The contents of this column are compared to the password submitted by the user.
- **Password Type:**
Depending on how the user's password has been stored in the database, select either **Clear Password** or **Digest Password** from the drop-down list.
- **Authorization Attribute Column:**
By running the **Database Query**, all of the user's attributes are returned. Only the user's username and password are used for the authentication event. You can also use one of the other user's attributes for authorization at a later stage in the policy. The additional authorization attribute should be either a username or an X.509 distinguished name (DName). You should enter the name of the column containing the username or the DName here, but only if this value is required for authorization purposes.
- **Authorization Attribute Format:**
The Enterprise Gateway's authorization filters all operate on the basis of a username or DName. They all evaluate whether a user identified by a username or DName is allowed to access a specific resource. Select the appropriate format from the drop-down list depending on what type of user credential is stored in the database table column entered above.

Entrust GetAccess Repositories

Entrust GetAccess provides Identity Management and access control services for Web resources. It centrally manages access to Web applications, enabling users to benefit from a single sign-on capability when accessing the applications that they are authorized to use.

You can configure the Enterprise Gateway to connect to a group of GetAccess servers in a round-robin fashion. This provides the necessary failover capability when one or more GetAccess servers are not available. When the Enterprise Gateway successfully authenticates to a GetAccess server, it obtains authorization information about the end-user from the GetAccess SAML PDP. The authorization details are returned in a SAML authorization assertion, which is then validated by the Enterprise Gateway to determine whether the request should be denied.

To authenticate users against an Entrust GetAccess repository, right-click **Entrust GetAccess Repositories**, and select **Add a new Repository**. Configure the following fields on the **Authentication Repository** dialog:

Repository Name:

Enter an appropriate name for this repository

Request:

Configure the following request settings:

- **URL Group:**
Select a URL group from the drop-down list. This group consists of a number of GetAccess Servers to which the Enterprise Gateway round-robins connection attempts. You can add URL groups on the **External Connections** tab in Policy Studio. Expand the **URL Connection Sets** node, right-click **Entrust GetAccess URL Sets**, and select **Add a URL Set**. For more details on adding and editing URL groups, see the [Configuring URL Groups](#) topic.
- **WS-Trust Attribute Field Name:**
Specify the field name for the `Id` field in the WS-Trust request. The default is `Id`.

Response:

Configure the following response settings:

- **SOAP Actor/Role:**
To add the SAML authorization assertion to the response message, select a SOAP actor/role to indicate the WS-Security block where the assertion is added. By leaving this field blank, the assertion is not added to the message.

Drift Time:

The specified time is used to allow for the possible difference between the time on the GetAccess SAML PDP and

the time on the machine hosting the Enterprise Gateway. This comes into effect when validating the SAML authorization assertion.

Further Information

For details on using a filter to integrate the Enterprise Gateway with Entrust GetAccess, see the [Entrust GetAccess Authorization](#) topic.

Oracle Access Manager Repositories

You can authorize an authenticated user for a particular resource against an Oracle Access Manager (OAM) repository. After successful authentication, OAM issues a Single Sign On (SSO) token, which can then be used instead of the user name and password.

To authenticate users against an Oracle Access Manager repository, right-click **Oracle Access Manager Repositories**, and select **Add a new Repository**. Configure the following fields on the **Authentication Repository** dialog:

Repository Name:

Enter an appropriate name for this repository.

Resource Request:

Configure the following settings for the resource request:

- **Resource Type:**
Enter the type of the resource for which you are requesting access. For example, for access to a Web-based URL, enter `http`.
- **Resource Name:**
Enter the name of the resource for which the user is requesting access. By default, this field is set to `/hostname${http.request.uri}`, which contains the original path requested by the client.
- **Operation:**
In most access management products, users are authorized for a limited set of actions on the requested resource. For example, users with management roles may be permitted to write (`HTTP POST`) to a certain Web Service, but users with junior roles might only have read access (`HTTP GET`) to the same service. Use this field to specify the operation to which you want to grant the user access on the specified resource. By default, this is set to the `http.request.verb` message attribute, which contains the HTTP verb used by the client to send the message to the Enterprise Gateway (for example, `HTTP POST`).

Single Sign On:

Configure the following settings for single sign on:

- **Create SSO Token:**
Select whether to create an SSO token. This is selected by default.
- **Store SSO Token in User Attribute:**
Enter the name of the message attribute that contains the user's SSO token. This attribute is populated when authenticating to Oracle Access Manager using the [HTTP Basic](#) or [HTTP Digest](#) filter. By default, the SSO token is stored in the `oracle.sso.token` message attribute.
- **Add SSO Token to User Attributes:**
Select whether to add the SSO Token to user message attributes. This is selected by default.

Oblix Installation Directory:

Enter the path to your Oblix installation directory. For more details on Oblix, see your Oracle Access Manager documentation.

Further Information

For details on using filters to integrate the Enterprise Gateway with Oracle Access Manager, see the [Oracle Access Manager Index](#) topic.

Oracle Entitlements Server Repositories

You can authenticate and authorize a user for a particular resource against an Oracle Entitlements Server (OES) repository.

For example, the Enterprise Gateway can extract credentials from the message sent by the client, and delegate authentication to the Oracle Entitlements Server. When the client has been authenticated, the Enterprise Gateway queries Oracle Entitlements Server to see if the client is permitted to access the Web Service resource. When authentication and authorization have passed, the message is trusted and forwarded to the target Web Service.

To authenticate and authorize users against an Oracle Entitlements Server repository, right-click **Oracle Entitlements Server Repositories**, and select **Add a new Repository**. Configure the following fields on the **Authentication Repository** dialog:

Repository Name:

Enter an appropriate name for this repository.

Oracle SSM Settings:

Click **Configure** to launch the **Oracle Security Service Module Settings** dialog. For details on configuring these settings, see the [Oracle Security Service Module Settings](#) topic.

RADIUS Repositories

You can configure the Enterprise Gateway to authenticate users in a Remote Authentication Dial In User Service (RADIUS) repository. RADIUS is a client-server network protocol that provides centralized authentication and authorization for clients connecting to remote services.

To authenticate users against a RADIUS repository, perform the following steps:

1. Right-click **RADIUS Repositories**, and select **Add a new Repository**.
2. In the **Authentication Repository** dialog, enter the **RADIUS Repository Name**.
3. On the **Client** tab, select the RADIUS clients that you wish to authenticate to the repository. For details on how to add clients to this list, see the [RADIUS Clients](#) topic.
4. On the **Attributes** tab, click **Add** to add a RADIUS attribute. This is a name-value pair used to determine how access is granted. Examples include `User-Name`, `User-Password`, `NAS-IP-Address`, or `NAS-Port`.
5. In the **RADIUS Attributes** dialog, specify a **Name** (for example, `User-Name`). You can select standard RADIUS attributes from the drop-down list, or enter a custom attribute.
6. Enter a **Value**, and click **OK**.
7. Click **OK**.

Repeat steps 4-6 to add multiple attributes. You can edit or delete attributes using the buttons provided.

RSA Access Manager Repositories

RSA Access Manager (formerly known as RSA ClearTrust) provides Identity Management and access control services for Web applications. It centrally manages access to Web applications, ensuring that only authorized users are allowed access to resources. Integration with RSA Access Manager requires RSA ClearTrust SDK version 6.0.

To authenticate users against an RSA Access Manager repository, right-click **RSA Access Manager Repositories**, and select **Add a new Repository**. Configure the following fields on the **Authentication Repository** dialog:

Repository Name:

Enter an appropriate name for this repository.

Connection Details:

The Enterprise Gateway can connect to a group of Access Manager *Authorization Servers* or *Dispatcher Servers*. When

multiple Access Manager Authorization Servers are deployed for load-balancing purposes, the Enterprise Gateway first connects to a Dispatcher Server, which returns a list of active Authorization Servers. An attempt is made to connect to one of these Authorization Servers using round-robin DNS. If the first Dispatcher Server in the Connection Group is not available, the Enterprise Gateway attempts to connect to the Dispatcher Server with the next highest priority in the group, and so on.

If a Dispatcher Server has not been deployed, the Enterprise Gateway can connect directly to an Authorization Server. If the Authorization Server with the highest priority in the Connection Group is not available, the Enterprise Gateway attempts to connect to the Authorization Server with the next highest priority, and so on. You can select the type of the Connection Group using the **Authorization Server** or **Dispatcher Server** radio button. All servers in the group must be of the same type.

Connection Group:

Select the **Connection Group** to use for authenticating clients. You can add Connection Groups on the **External Connections** tab in Policy Studio. Expand the **Connection Sets** node, right-click **RSA ClearTrust Connection Sets**, and select **Add a Connection Set**. For more details on adding and editing Connection Groups, see the [Configuring Connection Groups](#) topic

Authentication Type:

Select one of the following authentication types for the connection:

- HTTP Basic
- Windows NT
- RSA SecureID
- LDAP
- Certificate Distinguished Name

Further Information

For more details on prerequisites and on using a filter to integrate the Enterprise Gateway with RSA Access Manager, see the [RSA Access Manager Authorization](#) topic.

Tivoli Repositories

The Enterprise Gateway can integrate with Tivoli Access Manager to authenticate users. To authenticate users against a Tivoli repository, right-click **Tivoli Repositories**, and select **Add a new Repository**.

For more information on how to configure Enterprise Gateway to communicate with a Tivoli server, see the [Tivoli Integration](#) topic.

HTML Form-based Authentication

Overview

HTML Form-based Authentication enables users to supply their user name and password details in an HTML form, and submit them to login to a system. Using HTML form-based authentication, normal HTTP authentication features such as HTTP Basic or HTTP Digest are not used. Instead, the user name and password are typically sent as HTML `<FORM>` data in an HTTP `POST` over SSL.

When the **HTML Form-based Authentication** filter is configured, the Enterprise Gateway can authenticate the user details specified in the HTML form against a user profile stored in the Enterprise Gateway local repository, a database, or an LDAP directory.

Configuration

To configure the **HTML Form-based Authentication** filter, complete the following fields:

Name

Enter an appropriate name for the filter.

Username

Enter the name of the HTML form field in which the user enters their username. Defaults to `username`.

Password

Enter the name of the HTML form field in which the user enters their password. Defaults to `password`.

Credential Format

You must specify the format of the user credentials presented by the client because the Enterprise Gateway has no way of telling one credential format from another. Select from `User Name` or `Distinguished Name` in the drop-down list. The selected format is then used internally by the Enterprise Gateway when performing authorization lookups against third-party Identity Management servers.

Repository Name

This specifies the name of the Authentication Repository where all user profiles are stored. This can be in the Enterprise Gateway's local repository, in a database, or in an LDAP directory. Select a pre-configured **Repository Name** from the drop-down list.

You can add a new repository on the **External Connections** tab by right-clicking the appropriate node under **Authentication Repository Profiles** (for example, **Database Repositories**), and selecting **Add a new Repository**. For more details, see the [Authentication Repository](#) tutorial.

HTTP Basic Authentication

Overview

A client can authenticate to the Enterprise Gateway with a username and password combination using *HTTP Basic Authentication*. Whenever an **HTTP Basic Authentication** filter is configured, the Enterprise Gateway requests the client to present a username and password combination as part of the *HTTP Basic challenge-response* mechanism.

With HTTP Basic Authentication, the client's username and password are concatenated, base64-encoded, and passed in the `Authorization` HTTP header as follows:

```
Authorization: Basic dm9yZGVsOnZvcmlbA==
```

The Enterprise Gateway can then authenticate this user against a user profile stored in the Enterprise Gateway's local repository, a database, or an LDAP directory. The realm presented in the challenge for HTTP Basic Authentication is the realm currently specified in the system settings. See the [Default Settings](#) topic for more information.

Configuration

The information specified on this screen informs the Enterprise Gateway where it can find user profiles for authentication purposes. The Enterprise Gateway can look up user profiles in the Enterprise Gateway's local repository, in a database, or in an LDAP directory. You can add Users to the local repository using the **Users** interface. See the [Users](#) tutorial for more information.

To configure the **HTTP Basic Authentication** filter, complete the following fields:

Name

Enter a name for the filter here.

Credential Format

The username presented to the Enterprise Gateway during the HTTP Basic handshake can be of many formats, usually either username or Distinguished Name (DN). Because the Enterprise Gateway has no way of inherently telling one format from the other (for example, the client's username could be a DN), you must specify the format of the credential presented by the client. This format is then used internally by the Enterprise Gateway when performing authorization lookups against third-party Identity Management servers.

Allow Client Challenge

HTTP Basic Authentication can be configured with the following options:

- **Direct Authentication**
The client sends up the `Authorization` HTTP Basic Authentication header in its first request to the server.
- **Challenge-Response Handshake**
The client does *not* send the `Authorization` header when sending its request to the server (it does not know that the server requires HTTP Basic Authentication). The server responds with an *HTTP 401 response code*, instructing the client to authenticate to the server by sending the `Authorization` header. The client then sends a second request, this time including the `Authorization` header and the relevant username and password.

The first case is used mainly for machine-to-machine transactions in which there is no human intervention. The second case is typical of situations where a browser is talking to a Web server. When the browser receives the HTTP 401 re-

sponse to its initial request, it displays a dialog to enable the user to enter the username and password combination.

If you wish to force clients to always send the HTTP Basic `Authorization` header to the Enterprise Gateway, unselect the **Allow client challenge** checkbox. If you wish to allow clients to engage in the HTTP Basic Authentication challenge-response handshake with the Enterprise Gateway, ensure this feature is enabled by selecting this option.

Remove HTTP Authentication Header

Select this checkbox to remove the HTTP `Authorization` header from the downstream message. If this option is not selected, the incoming `Authorization` header is forwarded on to the destination Web Service.

Repository Name

This specifies the name of the Authentication Repository where all user profiles are stored. This can be in the Enterprise Gateway's local repository, in a database, or in an LDAP directory. Select a pre-configured **Repository Name** from the drop-down list.

You can add a new repository on the **External Connections** tab by right-clicking the appropriate node under **Authentication Repository Profiles** (for example, **Database Repositories**), and selecting **Add a new Repository**. For more details, see the [Authentication Repository](#) tutorial.

HTTP Digest Authentication

Overview

A client can authenticate to the Enterprise Gateway with a username and password digest using *HTTP Digest Authentication*. Whenever an **HTTP Digest Authentication** filter is configured, the Enterprise Gateway requests the client to present a username and password digest as part of the *HTTP Digest challenge-response* mechanism. The Enterprise Gateway can then authenticate this user against a user profile stored in the Enterprise Gateway database.

The realm presented in the challenge for HTTP Digest Authentication is the realm currently specified in the system settings. See the [Default Settings](#) topic for more information.

Configuration

The information specified on this screen informs the Enterprise Gateway where it can find user profiles for authentication purposes. The Enterprise Gateway can lookup user profiles in the Enterprise Gateway's local repository, in a database, or in an LDAP directory. Users can be added to the local repository using the **Users** interface. See the [Users](#) tutorial for more information on how to do this.

To configure the **HTTP Digest Authentication** filter, complete the following fields.

Name

Enter a name for the filter here.

Credential Format

The username presented to the Enterprise Gateway during the HTTP Digest handshake can be of many formats, usually either username or Distinguished Name (DName). Because the Enterprise Gateway has no way of inherently telling one format from the other (for example, the client's username could be a DName), it is necessary to specify the format of the credential presented by the client. This format is then used internally by the Enterprise Gateway when performing authorization lookups against third party Identity Management servers.

Session Timeout

As part of the *HTTP Digest Authentication* protocol, the Enterprise Gateway must generate a *nonce* (number used once) value, and send it to the client. The client uses this nonce to create the digest of the username and password. However, it should only be allowed a certain amount of time to do so. The **Session Timeout** field specifies the length of time (in milliseconds) for which the nonce is valid.

Remove HTTP Authentication Header

Select this checkbox to remove the HTTP *Authorization* header from the downstream message. If this option is not selected, the incoming *Authorization* header is forwarded on to the destination Web Service.

Repository Name

This specifies the name of the Authentication Repository where all user profiles are stored. This can be in the Enterprise Gateway's local repository, in a database, or in an LDAP directory. Select a pre-configured **Repository Name** from the drop-down list.

You can add a new repository on the **External Connections** tab by right-clicking the appropriate node under **Authentication Repository Profiles** (for example, **Database Repositories**), and selecting **Add a new Repository**. For more details, see the [Authentication Repository](#) tutorial.

HTTP Header Validation

Overview

You can use the **HTTP Header** filter in cases where the Enterprise Gateway receives end-user authentication credentials in an HTTP header. A typical scenario would see the end-user (or message originator) authenticating to an intermediary. The intermediary authenticates the end-user, and to propagate the end-user credentials to the destination Web Service, the intermediary inserts the credentials into an HTTP header and forwards them onwards.

When the Enterprise Gateway receives the message, it performs the following tasks:

- Authenticate the sender of the message (the intermediary)
- Extract the *end-user* identity from the token in the HTTP header for use in subsequent Authorization filters

Important Note:

In the case outlined above, the Enterprise Gateway does *not* attempt to re-authenticate the end-user. It trusts that the intermediary has already authenticated the end-user, and so the Enterprise Gateway does not authenticate the user again. However, it is good practice to authenticate the message sender (the intermediary). Any subsequent Authorization filters use the end-user credentials that were passed in the HTTP header.

Configuration

The following configuration fields are available on this screen:

Name:

Enter an appropriate name for this filter in the **Name** field.

HTTP Header Name:

Enter the name of the HTTP Header that contains the end-user credentials.

HTTP Header Type:

Select the type of credentials that are passed in the named HTTP Header. The following types are supported:

1. X.509 Distinguished Name
2. Certificate
3. Username

IP Address

Overview

You can configure the Enterprise Gateway to allow or deny machines, or groups of machines, access to resources based on IP address. The main table on the screen shows the IP addresses from which the Enterprise Gateway accepts or denies messages depending on what is configured.

The **IP Address** Authentication filter uses the value stored in the `http.request.clientaddr` message attribute to determine whether or not to allow or deny access. This message attribute contains the remote host address from the TCP socket used in the connection between the client and the Enterprise Gateway.

Configuration

The following fields must be configured:

Name:

Enter a name for the filter.

IP Addresses:

You can add IP addresses by clicking the **Add** button, which displays the **Add IP Filter** dialog. Enter an **IP Address** and **Subnet Mask** to indicate a network to filter.

Messages sent from hosts belonging to this network will be accepted or rejected based on what is configured in the section below. A **Subnet Mask** of 255.255.255.255 can be used to filter specific IP addresses. For more details, see [Configuring Subnet Masks](#).

Important Note:

If requests are made across a proxy, portal, or other such intermediary, the Enterprise Gateway filters on the IP address of the intermediary. Therefore, you should enter the IP address of the intermediary on this screen, and not that of the user/client machine.

You can edit and remove existing IP addresses by selecting the **Edit** and **Remove** buttons.

Access:

Depending on whether the **Allow Access** or **Deny Access** radio button is checked, the IP addresses listed in the table are allowed or denied access to the Web Service.

Configuring Subnet Masks

An IP address is normally represented by a string of 4 numbers separated by periods (for example, 192.168.0.20). Each number is normally represented as the decimal equivalent of an eight-bit binary number, which means that each number may take any value between 0 (all eight bits cleared) and 255 (all eight bits set).

A *subnet mask* (or netmask) is also a set of four number blocks separated by periods, each of which has a value in the range 0-255. Every IP address consists of two parts: the network address and the host number. The netmask is used to determine the size of these two parts. The positions of the bits set in the netmask represent the space reserved for the network address, while the bits that are cleared represent the space reserved for the host number. The netmask determines the range of IP addresses.

The following examples illustrate how netmasks work in practice:

Example 1: Specifying a Range of IP Addresses:

You only want to allow requests from the following IP addresses:

192.168.0.16, 192.168.0.17, 192.168.0.18, and 192.168.0.19.

Use the following address/netmask combination to cover the 4 IP addresses listed above:

192.168.0.16/255.255.255.252

In more detail, the binary representation of the netmask is as follows:

```
11111111.11111111.11111111.11111100
```

The top 30 bits of the netmask indicate the network and the last 2 bits refer to the host on the network. These last 2 bits allow 4 different addresses as shown in the worked example below.

When the Enterprise Gateway receives a request from a certain IP address, the Enterprise Gateway performs a logical AND on the client IP address and the configured netmask. It also does a logical AND with the IP address entered in the IP Address filter and the configured subnet mask. If the AND-ed binary values are the same, the request from the IP address can be considered in the same network range as that configured in the filter.

The following worked example illustrates the mechanics of the IP address filtering. It assumes that you have entered the following in the IP Address and Netmask fields in the IP Address filter:

Field	Value
IP Address	192.168.0.16
Net Mask	255.255.255.252

Step 1: AND the IP address and Netmask configured in the IP Address Filter:

```
11000000.10100000.00000000.00010000 (192.168.0.16)
AND
11111111.11111111.11111111.11111100 (255.255.255.252)
=====
11000000.10100000.00000000.00010000
```

Step 2: Request is received from 192.168.0.18:

```
11000000.10100000.00000000.00010010 (192.168.0.18)
AND
11111111.11111111.11111111.11111100 (255.255.255.252)
=====
11000000.10100000.00000000.00010000
===> AND-ed value is equal to the result for 192.168.0.16.
===> Therefore the client IP address is inside the configured range.
```

Step 3: Request is received from 192.168.0.20:

```
11000000.10100000.00000000.00010100 (192.168.0.20)
AND
11111111.11111111.11111111.11111100 (255.255.255.252)
=====
11000000.10100000.00000000.00010100
===> AND-ed value is NOT equal to the result for 192.168.0.16.
===> Therefore the client IP address is NOT inside the configured range.
```

Example 2: Specifying an Exact IP Address:

You can also specify an exact IP address by using a netmask of 255.255.255.255. When this netmask is used, only requests from this client IP address is allowed or blocked, depending on what is configured in the filter. This example assumes that the following details have been configured in the IP Address filter:

<i>Field</i>	<i>Value</i>
IP Address	192.168.0.36
Net Mask	255.255.255.255

Step 1: AND the IP address and Netmask configured in the IP Address Filter:

```
11000000.10100000.00000000.00100100 (192.168.0.36)
AND
11111111.11111111.11111111.11111111 (255.255.255.255)
=====
11000000.10100000.00000000.00100100
```

Step 2: Request is received from client with IP address of 192.168.0.37:

```
11000000.10100000.00000000.00100101 (192.168.0.37)
AND
11111111.11111111.11111111.11111111 (255.255.255.255)
=====
11000000.10100000.00000000.00100101
===> AND-ed value is NOT equal to the result for 192.168.0.36
===> Therefore the client IP address is NOT inside the configured range.
```


SSL Authentication

Overview

A client can mutually authenticate to the Enterprise Gateway through the exchange of X.509 certificates. An X.509 certificate contains identity information about its owner and is digitally signed by the Certificate Authority that issued it.

A client will present such a certificate to the Enterprise Gateway while the initial SSL/TLS session is being negotiated, in other words, during the *SSL handshake*. The **SSL Authentication** filter extracts this information from the client certificate and sets it as message attributes. These attributes can then be used by subsequent filters in the policy.

The **SSL Authentication** filter can be used as a decision-making node on the policy. For example, it can be used to determine a path through a policy based on how users authenticate to the Enterprise Gateway.

Configuration

Name:

Enter a name for the filter on the **SSL Authentication** configuration screen.

Attribute Authentication

Overview

In cases where user credentials are passed to the Enterprise Gateway in a non-standard way, these credentials can be copied into Enterprise Gateway message attributes, and then authenticated against a specified authentication repository, such as the Enterprise Gateway User Store, an LDAP directory, or a database.

For example, assume that username and password credentials are passed to the Enterprise Gateway in the following XML message:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <ns:User xmlns:ns="http://www.user.com">
      <ns:Username>1</ns:Username>
      <ns:Password>2</ns:Password>
    </ns:User>
  </s:Body>
</s:Envelope>
```

In this example, the standard methods of passing credentials, such as HTTP Basic/Digest authentication, SAML assertions, WS-Security Username tokens, are bypassed, and the client sends the username and password as parameters in a simple SOAP message.

When the Enterprise Gateway receives this message, it can extract the value of the `<Username>` and `<Password>` elements using an XPath expression configured in the **Retrieve Attributes from Message** filter. This filter uses an XPath expression to retrieve the value of an element or attribute, and can then store this value in the specified message attribute.

In this example, you can configure an instance of this filter to retrieve the value of the `<Username>` attribute, and store it in the `authentication.subject.id` message attribute. Similarly, you can configure another filter to retrieve the value of the `<Password>`, and store it in the `authentication.subject.password` message attribute.

The **Attribute Authentication** filter can then use the username and password values stored in these message attributes to authenticate the user against the specified authentication repository.

Configuration

Complete the following fields to configure this filter:

Name:

Enter an appropriate name for this filter.

Username:

Specify the Enterprise Gateway message attribute that contains the username of the user to be authenticated. The default attribute is the `authentication.subject.id` attribute, which is typically used to store a username.

Password:

Enter the Enterprise Gateway message attribute that contains the password of the user to authenticate. The default message attribute is the `authentication.subject.password` attribute, which is typically used to store a password.

Credential Format:

Select the format of the credential stored in the Enterprise Gateway message attribute specified in the **Username** field above. By default, `User Name` is selected.

Repository Name:

Select an existing repository to authenticate the user against from the drop-down list. Alternatively, you can configure a new authentication repository by clicking the **Add** button. For more details on configuring the various types of repository supported by the Enterprise Gateway, see the [Authentication Repository](#) tutorial.

SAML Authentication

Overview

A Security Assertion Markup Language (SAML) *authentication assertion* is issued as proof of an authentication event. Typically, an end-user authenticates to an intermediary, who generates a SAML authentication assertion to prove that it has authenticated the user. The intermediary inserts the assertion into the message for consumption by a downstream Web Service.

When the Enterprise Gateway receives a message containing a SAML authentication assertion, it does not attempt to authenticate the end-user again. Instead, it authenticates the sender of the assertion (the intermediary) to ensure that only the intermediary could have issued the assertion, and then validates the authentication details contained in the assertion.

Therefore, the Enterprise Gateway performs the following tasks in this scenario:

- Authenticates the sender of the message (the intermediary)
- Extracts the end-user's identity from the authentication assertion and validates the authentication details

The **SAML Authentication** filter performs the second task. A separate authentication filter must be placed before this filter in the policy to authenticate the sender of the assertion. The end-user's identity is used in any subsequent authorization filters.

The following sample SOAP message contains a SAML authentication assertion:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
  <soap-env:Header xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext">
    <wsse:Security>
      <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
        AssertionID="oracle-1056477425082"
        Id="oracle-1056477425082"
        IssueInstant="2003-06-24T17:57:05Z"
        Issuer="CN=Sample User, . . . , C=IE"
        MajorVersion="1"
        MinorVersion="0">
        <saml:Conditions
          NotBefore="2003-06-20T16:20:10Z"
          NotOnOrAfter="2003-06-20T18:20:10Z"/>
        <saml:AuthenticationStatement
          AuthenticationInstant="2003-06-24T17:57:05Z"
          AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
          <saml:SubjectLocality IPAddress="192.168.0.32"/>
          <saml:Subject>
            <saml:NameIdentifier
              Format="urn:oasis:names:tc:SAML:1.0:assertion#X509SubjectName">
              sample
            </saml:NameIdentifier>
          </saml:Subject>
        </saml:AuthenticationStatement>
      <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
        id="Sample User">
        <dsig:SignedInfo>
          . . . . .
        </dsig:SignedInfo>
      </dsig:Signature>
    </wsse:Security>
  </soap-env:Header>
  <soap-env:Body>
    . . . . .
  </soap-env:Body>
</soap-env:Envelope>
```

```

    <dsig:SignatureValue>
      rpa/.....0g==
    </dsig:SignatureValue>
    <dsig:KeyInfo>
      .....
    </dsig:KeyInfo>
  </dsig:Signature>
</saml:Assertion>
</wsse:Security>
</soap-env:Header>
<soap-env:Body>
  <ns1:getTime xmlns:ns1="urn:timeservice">
    </ns1:getTime>
  </soap-env:Body>
</soap-env:Envelope>

```

General Settings

Configure the following field:

Name:

Enter an appropriate name for the filter.

Details

Configure the following fields on the **Details** tab:

SOAP Actor/Role:

If you expect the SAML assertion to be embedded in a WS-Security block, you can identify this block by specifying the SOAP Actor or Role of the WS-Security header that contains the assertion.

XPath Expression:

Alternatively, if the assertion is not contained in a WS-Security block, you can enter an XPath expression to locate the authentication assertion. You can configure XPath expressions using the **Add Edit** and **Delete** buttons.

SAML Namespace:

Select the SAML namespace that must be used on the SAML assertion for this filter to succeed. If you do not wish to check the namespace, select the `Do not check version` option from the drop-down list.

SAML Version:

Enter the SAML Version that the assertion must adhere to by entering the major version in the first field, followed by the minor version in the second field. For example, for SAML version 2.0, enter 2 in the first field and 0 in the second field.

Drift Time:

The *drift time*, specified in seconds, is used when checking the validity dates on the authentication assertion. The drift time allows for differences between the clock times of the machine on which the assertion was generated and the machine hosting the Enterprise Gateway.

Remove Enclosing WS-Security Element on Successful Validation:

Select this checkbox if you wish to remove the WS-Security block that contains the SAML assertion after the assertion has been successfully validated.

Trusted Issuers

You can use the table on this tab to select the issuers that you consider trusted. In other words, this filter only accepts

assertions that have been issued by the SAML Authorities selected here.

Click the **Add** button to display the **Trusted Issuers** screen. Select the Distinguished Name of a SAML Authority whose certificate has been added to the Certificate Store, and click **OK**. Repeat this step to add more SAML Authorities to the list of trusted issuers.

SAML PDP Authentication

Overview

The Enterprise Gateway can request an authentication decision from a Security Assertion Markup Language (SAML) Policy Decision Point (PDP) for an authenticated client using the SAML Protocol (SAML). In such cases, the Enterprise Gateway presents evidence to the PDP in the form of some user credentials, such as the Distinguished Name of a client's X.509 certificate.

The PDP decides whether to authenticate the end-user. It then creates an authentication assertion, signs it, and returns it to the Enterprise Gateway in a SAML Protocol response. The Enterprise Gateway can then perform a number of checks on the response, such as validating the PDP signature and certificate, and examining the assertion. It can also insert the SAML authentication assertion into the message for consumption by a downstream Web Service.

Request Configuration

You can configure a group of SAML PDPs to which the Enterprise Gateway connects in a round-robin fashion if one or more of the PDPs are unavailable. This is known as a SAML PDP URL Set. You can configure a SAML PDP URL Set on the **External Connections** tab in the Policy Studio. Expand the **URL Connection Sets** node, right-click **SAML PDP URL Set**, and select **Add a URL Set**. For more details, see the [Configuring URL Groups](#) topic.

When you have configured a group of SAML PDPs to connect to, you can configure the following general fields:

- **SAML PDP URL Set:**
Select a previously configured SAML PDP URL Set from the drop-down list. You can configure a SAML PDP URL Set on the **External Connections** tab.
- **SOAPAction:**
Enter the SOAP Action required to send SAML Protocol requests to the PDP. Click the **Use Default** button to use the following default SOAP Action as specified by the SAML Protocol:
<http://www.oasis-open.org/committees/security>
- **SAML Version:**
Select the SAML version to use in the SAML request.
- **Signing Key:**
If the SAML request is to be signed, click the **Signing Key** button, and select the appropriate signing key from the Certificate Store.

SAML Subject:

The specified details describe the *subject* of the SAML assertion. Complete the following fields:

- **Subject Attribute:**
Select the message attribute that contains the name of an authenticated user name. By default, the `authentication.subject.id` message attribute is selected, which contains the user name of the authenticated user.
- **Subject Format:**
Select the format of the message attribute selected in the **Subject Attribute** field above. You do not need to select a format if the **Subject Attribute** field is set to `authentication.subject.id`

Subject Confirmation:

The settings on the **Confirmation Method** tab determine how the `<SubjectConfirmation>` block of the SAML assertion is generated. When the assertion is consumed by a downstream Web Service, the information contained in the `<SubjectConfirmation>` block can be used to authenticate the end-user that authenticated to the Enterprise Gateway, or the issuer of the assertion, depending on what is configured.

The following is a typical <SubjectConfirmation> block:

```
<saml:SubjectConfirmation>
  <saml:ConfirmationMethod>
    urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
  </saml:ConfirmationMethod>
  <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
    <dsig:X509Data>
      <dsig:X509SubjectName>CN=oracle</dsig:X509SubjectName>
      <dsig:X509Certificate>
        MIICmzCCAY . . . . . mB9CJEw4Q=
      </dsig:X509Certificate>
    </dsig:X509Data>
  </dsig:KeyInfo>
</saml:SubjectConfirmation>
</saml:SubjectConfirmation>
```

You must configure the following fields on the **Subject Confirmation** tab:

Method:

The selected value determines the value of the <ConfirmationMethod> element. The following table shows the available methods, their meanings, and their respective values in the <ConfirmationMethod> element:

Method	Meaning	Value
Holder Of Key	Inserts a <SubjectConfirmation> into the SAML request. The <SubjectConfirmation> contains a <dsig:KeyInfo> section with the certificate of the user selected to sign the SAML request. The user selected to sign the SAML request must be the authenticated subject (authentication.subject.id). Select the Include Certificate option if the signer's certificate is to be included in the SubjectConfirmation block. Alternatively, select the Include Key Name option if only the key name is to be included.	urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
Bearer	Inserts a <SubjectConfirmation> into the SAML request.	urn:oasis:names:tc:SAML:1.0:cm:bearer
SAML Artifact	Inserts a <SubjectConfirmation> into the SAML request.	urn:oasis:names:tc:SAML:1.0:cm:artifact
Sender Vouches	Inserts a <SubjectConfirmation> into the SAML request. A user must sign the SAML request.	urn:oasis:names:tc:SAML:1.0:cm:bearer

If the **Method** field is left blank, no `<ConfirmationMethod>` block is inserted into the assertion.

Include Certificate:

Select this option if you wish to include the SAML subject's certificate in the `<KeyInfo>` section of the `<SubjectConfirmation>` block.

Include Key Name:

Alternatively, if you do not want to include the certificate, select this option to only include the key name in the `<KeyInfo>` section.

Response Configuration

This tab configures the SAML Response returned from the SAML PDP. The following fields are available:

SOAP Actor/Role:

If the SAML response from the PDP contains a SAML authentication assertion, the Enterprise Gateway can extract it from the response and insert it into the downstream message. The SAML assertion is inserted into the WS-Security block identified by the specified SOAP actor/role.

Drift Time:

The SAML request to the PDP is timestamped by the Enterprise Gateway. To account for differences in the times on the machines running the Enterprise Gateway and the SAML PDP the specified time is subtracted from the time at which the Enterprise Gateway generates the SAML request.

Insert SAML Authentication Assertion

Overview

After successfully authenticating a client, the Enterprise Gateway can insert a SAML (Security Assertion Markup Language) authentication assertion into the SOAP message. Assuming all other security filters in the policy are successful, the assertion will eventually be consumed by a downstream Web Service.

It may be useful to refer to the following example of a signed SAML authentication assertion when configuring this filter.

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
<soap-env:Header xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext">
<wsse:Security>
  <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
    AssertionID="oracle-1056477425082"
    Id="oracle-1056477425082"
    IssueInstant="2003-06-24T17:57:05Z"
    Issuer="CN=Sample User, . . . , C=IE"
    MajorVersion="1"
    MinorVersion="0">
    <saml:Conditions
      NotBefore="2003-06-20T16:20:10Z"
      NotOnOrAfter="2003-06-20T18:20:10Z"/>
    <saml:AuthenticationStatement
      AuthenticationInstant="2003-06-24T17:57:05Z"
      AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
      <saml:SubjectLocality IPAddress="192.168.0.32"/>
      <saml:Subject>
        <saml:NameIdentifier
          Format="urn:oasis:names:tc:SAML:1.0:assertion#X509SubjectName">
          sample
        </saml:NameIdentifier>
      </saml:Subject>
    </saml:AuthenticationStatement>
  <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
    id="Sample User">
    <dsig:SignedInfo>
      . . . . .
    </dsig:SignedInfo>
    <dsig:SignatureValue>
      rpa/.....0g==
    </dsig:SignatureValue>
    <dsig:KeyInfo>
      . . . . .
    </dsig:KeyInfo>
  </dsig:Signature>
</saml:Assertion>
</wsse:Security>
</soap-env:Header>
<soap-env:Body>
  <ns1:getTime xmlns:ns1="urn:timeservice">
</ns1:getTime>
</soap-env:Body>
</soap-env:Envelope>
```

General Configuration

Configure the following field:

Name:

Enter an appropriate name for the filter.

Assertion Details

Configure the following fields on the **Assertion Details** tab:

Issuer Name:

Select the certificate containing the Distinguished Name (DName) that you want to use as the Issuer of the SAML assertion. This DName is included in the SAML assertion as the value of the `Issuer` attribute of the `<saml:Assertion>` element. For an example, see the sample SAML assertion above.

Expire In:

Specify the lifetime of the assertion in this field. The lifetime of the assertion lasts from the time of insertion until the specified amount of time has elapsed.

Drift Time:

The **Drift Time** is used to account for differences in the clock times of the machine hosting the Enterprise Gateway (that generate the assertion) and the machines that consume the assertion. The specified time is subtracted from the time at which the Enterprise Gateway generates the assertion.

SAML Version:

You can create SAML 1.0, 1.1, and 2.0 attribute assertions. Select the appropriate version from the drop-down list.

Important Note:

SAML 1.0 recommends the use of the <http://www.w3.org/TR/2001/REC-xml-c14n-20010315> XML Signature Canonicalization algorithm. When inserting signed SAML 1.0 assertions into XML documents, it is quite likely that subsequent signature verification of these assertions will fail. This is due to the side effect of the algorithm including inherited namespaces into canonical XML calculations of the inserted SAML assertion that were not present when the assertion was generated.

For this reason, Oracle recommend that SAML 1.1 or 2.0 is used when signing assertions as they both uses the exclusive canonical algorithm <http://www.w3.org/2001/10/xml-exc-c14n#>, which safeguards inserted assertions from such changes of context in the XML document. Please see section 5.4.2 of the [oasis-sstc-saml-core-1.0.pdf](#) and section 5.4.2 of [sstc-saml-core-1.1.pdf](#) documents, both of which are available at <http://www.oasis-open.org>.

Assertion Location

The options on the **Assertion Location** tab specify where the SAML assertion is inserted in the message. By default, the SAML assertion is added to the WS-Security block with the current SOAP actor/role. The following options are available:

Append to Root or SOAP Header:

Appends the SAML assertion to the message root for a non-SOAP XML message, or to the SOAP Header for a SOAP message. For example, this option may be suitable for cases where this filter may process SOAP XML messages or non-SOAP XML messages.

Add to WS-Security Block with SOAP Actor/Role:

Adds the SAML assertion to the WS-Security block with the specified SOAP actor (SOAP 1.0) or role (SOAP 1.1). By default, the assertion is added with the current SOAP actor/role only, which means the WS-Security block with no actor. You can select a specific SOAP actor/role when available from the drop-down list.

XPath Location:

If you wish to insert the SAML assertion at an arbitrary location in the message, you can use an XPath expression to

specify the exact location in the message. You can select XPath expressions from the drop-down list. The default is the `First WSSE Security Element`, which has an XPath expression of `//wsse:Security`. You can add, edit, or remove expressions by clicking the relevant button. For more details, see the [Configuring XPath Expressions](#) topic.

You can also specify how exactly the SAML assertion is inserted using the following options:

- **Append to node returned by XPath expression** (the default)
- **Insert before node returned by XPath expression**
- **Replace node returned by XPath expression**

Insert into Message Attribute:

Specify a message attribute to store the SAML assertion from the drop-down list (for example, `saml.assertion`). Alternatively, you can also enter a custom message attribute in this field (for example, `my.test.assertion`). The SAML assertion can then be accessed downstream in the policy circuit.

Subject Confirmation Method

The settings on the **Subject Confirmation Method** tab determine how the `<SubjectConfirmation>` block of the SAML assertion is generated. When the assertion is consumed by a downstream Web Service, the information contained in the `<SubjectConfirmation>` block can be used to authenticate the end-user that authenticated to the Enterprise Gateway, or the issuer of the assertion, depending on what is configured.

The following is a typical `<SubjectConfirmation>` block:

```
<saml:SubjectConfirmation>
  <saml:ConfirmationMethod>
    urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
  </saml:ConfirmationMethod>
  <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
    <dsig:X509Data>
      <dsig:X509SubjectName>CN=oracle</dsig:X509SubjectName>
      <dsig:X509Certificate>
        MIICmzCCAY . . . . . mB9CJEw4Q=
      </dsig:X509Certificate>
    </dsig:X509Data>
  </dsig:KeyInfo>
</saml:SubjectConfirmation>
```

The following configuration fields are available on the **Subject Confirmation Method** tab:

Method:

The selected value determines the value of the `<ConfirmationMethod>` element. The following table shows the available methods, their meanings, and their respective values in the `<ConfirmationMethod>` element:

Method	Meaning	Value
Holder Of Key	The Enterprise Gateway includes the key used to prove that the Enterprise Gateway is the holder of the key, or it includes a reference to the key.	urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
Bearer	The subject of the assertion is the	urn:oasis:names:tc:SAML:1.0:cm

Method	Meaning	Value
	bearer of the assertion.	m:bearer
SAML Artifact	The subject of the assertion is the user that presented a SAML Artifact to the Enterprise Gateway.	urn:oasis:names:tc:SAML:1.0:cm:artifact
Sender Vouches	Use this confirmation method to assert that the Enterprise Gateway is acting on behalf of the authenticated end-user. No other information relating to the context of the assertion is sent. It is recommended that both the assertion and the SOAP Body must be signed if this option is selected. These message parts can be signed by using the Sign Message filter.	urn:oasis:names:tc:SAML:1.0:cm:bearer

Note that you can also leave the **Method** field blank, in which case no <ConfirmationMethod> block is inserted into the assertion.

Holder-of-Key Configuration:

When you select **Holder-of-Key** as the SAML subject confirmation in the **Method** field, you must configure how information about the key is included in the message. There are a number of configuration options available depending on whether the key is a symmetric or asymmetric key.

Asymmetric Key:

If you want to use an asymmetric key as proof that the Enterprise Gateway is the holder-of-key entity, you must select the **Asymmetric Key** radio button and then configure the following fields on the **Asymmetric** tab:

- **Private Key from Certificate Store:**
If you want to select a key that is stored in the Certificate Store, select this option, and click the **Signing Key** button. On the **Select Certificate** screen, select the checkbox next to the certificate that is associated with the key that you want to use.
- **Private Key from Message Attribute:**
Alternatively, the key may have already been used by a previous filter in the circuit, for example to sign a part of the message. In this case, the key is stored in a message attribute. You can specify this message attribute in this field.

Symmetric Key:

If you want to use a symmetric key as proof that the Enterprise Gateway is the holder of key, select the **Symmetric Key** radio button, and configure the fields on the **Symmetric** tab:

- **Generate Symmetric Key:**
If you select this option, the Enterprise Gateway generates a symmetric key, which is included in the message before it is sent to the client.
- **Symmetric Key in Message Attribute:**
If a previous filter (for example, a **Sign Message** filter) has already used a symmetric key, you can reuse this key as proof that the Enterprise Gateway is the holder-of-key entity. You must enter the name of the message attribute in this field.
- **Encrypt using Certificate from Certificate Store:**
When a symmetric key is used, you must assume that the recipient has no prior knowledge of this key. It must,

therefore, be included in the message so that the recipient can validate the key. To avoid meet-in-the-middle style attacks, where a hacker could eavesdrop on the communication channel between the Enterprise Gateway and the recipient and gain access to the symmetric key, the key must be encrypted so that only the recipient can decrypt the key. One way of doing this is to select the recipient's certificate from the Certificate Store. By encrypting the symmetric key with the public in the recipient's certificate, the key can only be decrypted by the recipient's private key, to which only the recipient has access. Select the **Signing Key** button, and select the recipient's certificate on the **Select Certificate** dialog.

- **Encrypt using Certificate from Message Attribute:**
Alternatively, if the recipient's certificate has already been used (perhaps to encrypt part of the message) this certificate is stored in a message attribute. You can enter this message attribute in this field.
- **Symmetric Key Length:**
Enter the length (in bits) of the symmetric key to use.
- **Key Wrap Algorithm:**
Select the algorithm to use to encrypt (*wrap*) the symmetric key.

Key Info:

The **Key Info** tab must be configured regardless of whether you have elected to use symmetric or asymmetric keys. It determines how the key is included in the message. The following options are available:

- **Do Not Include Key Info:**
Select this option if you do not wish to include a <KeyInfo> section in the SAML assertion.
- **Embed Public Key Information:**
If this option is selected, details about the key are included in a <KeyInfo> block in the message. You can include the full certificate, expand the public key, include the distinguished name, and include a key name in the <KeyInfo> block by selecting the appropriate boxes. When selecting the **Include Key Name** field, you must enter a name in the **Value** field, and then select the **Text Value** or **Distinguished Name Attribute** radio button, depending on the source of the key name.
- **Put Certificate in Attachment:**
Select this option to add the certificate as an attachment to the message. The certificate is then referenced from the <KeyInfo> block.
- **Security Token Reference:**
The Security Token Reference (STR) provides a way to refer to a key contained within a SOAP message from another part of the message. It is often used in cases where different security blocks in a message use the same key material and it is considered an overhead to include the key more than once in the message. When this option is selected, a <wsse:SecurityTokenReference> element is inserted into the <KeyInfo> block. It references the key material using a URI to point to the key material and a *ValueType* attribute to indicate the type of reference used. For example, if the STR refers to an encrypted key, you should select *EncryptedKey* from the dropdown, whereas if it refers to a *BinarySecurityToken*, you should select *X509v3* from the dropdown. Other options are available to enable more specific security requirements.

Advanced

Select Required Layout Type:

WS-Policy and SOAP Message Security define a set of rules that determine the layout of security elements that appear in the WS-Security header within a SOAP message. The SAML assertion will be inserted into the WS-Security header according to the layout option selected here. The available options correspond to the WS-Policy Layout assertions of *Strict*, *Lax*, *LaxTimestampFirst*, and *LaxTimestampLast*.

Insert SAML Attribute Statement:

You can insert a SAML attribute statement into the generated SAML authentication assertion. If you select this option, a SAML attribute assertion is generated using attributes stored in the `attribute.lookup.list` message attribute and subsequently inserted into the assertion. The `attribute.lookup.list` attribute must have been populated previously by an attribute lookup filter for the attribute statement to be generated successfully.

Indent:

Select this method to ensure that the generated signature is properly indented.

Security Token Reference:

The generated SAML authentication assertion can be encapsulated within a <SecurityTokenReference> block. The following example demonstrates this:

```
<soap:Header>
  <wsse:Security
    xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"
    soap:actor="oracle">
    <wsse:SecurityTokenReference>
      <wsse:Embedded>
        <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
          AssertionID="Id-00000109fee52b06-0000000000000012"
          IssueInstant="2006-03-15T17:12:45Z"
          Issuer="oracle" MajorVersion="1" MinorVersion="0">
          <saml:Conditions NotBefore="2006-03-15T17:12:39Z"
            NotOnOrAfter="2006-03-25T17:12:39Z"/>
          <saml:AuthenticationStatement
            AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
            AuthenticationInstant="2006-03-15T17:12:45Z">
            <saml:Subject>
              <saml:NameIdentifier Format="Oracle-Username-Password">
                admin
              </saml:NameIdentifier>
              <saml:SubjectConfirmation>
                <saml:ConfirmationMethod>
                  urn:oasis:names:tc:SAML:1.0:cm:artifact
                </saml:ConfirmationMethod>
                </saml:SubjectConfirmation>
              </saml:Subject>
            </saml:AuthenticationStatement>
          </saml:Assertion>
        </wsse:Embedded>
      </wsse:SecurityTokenReference>
    </wsse:Security>
  </soap:Header>
```

To add the SAML assertion to a <SecurityTokenReference> block as in the example above, select the **Embed SAML assertion within Security Token Reference** option. Otherwise, select **No Security Token Reference**.

WS-Security Username Authentication

Overview

A WS-Security *Username Token* enables an end-user identity to be passed over multiple hops before reaching the destination Web Service. The user identity is inserted into the message and is available for processing at each hop on its path.

The client user name and password are encapsulated in a WS-Security `<wsse:UsernameToken>`. When the Enterprise Gateway receives this token, it can perform one of the following tasks, depending on the requirements:

- Ensure that the timestamp on the token is still valid
- Authenticate the user name against a repository
- Authenticate the user name and password against a repository

The following sample SOAP message contains two `<wsse:UsernameToken>` blocks:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
      <wsse:UsernameToken wsu:Id="sample"
        xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
        <wsse:Username>sample</wsse:Username>
        <wsse:Password Type="wsse:PasswordText">oracle</wsse:Password>
        <wsu:Created>2004-05-19T08:44:51Z</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
    <wsse:Security soap:actor="oracle"
      xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
      <wsse:UsernameToken wsu:Id="oracle"
        xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
        <wsse:Username>oracle</wsse:Username>
        <wsse:Password Type="wsse:PasswordText">oracle</wsse:Password>
        <wsu:Created>2004-05-19T08:46:04Z</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <getHello xmlns="http://www.oracle.com"/>
  </soap:Body>
</soap:Envelope>
```

This topic explains how to configure the Enterprise Gateway to authenticate users using a WS-Security `<wsse:UsernameToken>`.

General Configuration

To configure general settings, complete the following fields:

Name:

Enter an appropriate name for this filter.

Actor:

The example SOAP message at the top of this page contains two `<wsse:UsernameToken>` blocks. You must specify which block contains the `<wsse:UsernameToken>` used to authenticate the end-user. Specify the SOAP Actor/Role of the WS-Security block that contains the token.

Credential Format:

The Enterprise Gateway can authenticate users against a user profile repository based on User Names, X.509 Distinguished Names, or email addresses. Unfortunately, the WS-Security specification does not provide a means of specifying the type of `<wsse:UsernameToken>`, and so it is necessary for the administrator to do so using the **Credential Format** field. The type specified here is used internally by the Enterprise Gateway in subsequent authorization filters.

Token Validation

Each `wsse:UsernameToken` contains a timestamp inserted into the `<wsu:Created>` element. Using this timestamp together with the details entered in this section, the Enterprise Gateway can determine whether the WS-Security `UsernameToken` has expired. The `<wsu:Created>` element is as follows:

```
<wsse:UsernameToken wsu:Id="oracle"
  xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
  <wsu:Created>2006.01.13T-10:42:43Z</wsu:Created>
  ...
</wsse:UsernameToken>
```

To configure token validation settings, complete the following fields:

Drift Time:

Specified in seconds to account for differences in the clock times between the machine on which the token was generated and the machine running the Enterprise Gateway. Using the *start time*, *end time*, and *drift time*, the token is considered valid if the current time falls between the following times:

```
[start - drift] and [start + drift + end]
```

Validity Period:

Specifies the lifetime of the token, where the value of the `<wsu:Created>` element represents the *start time* of the assertion, and the time period entered represents the *end time*.

Timestamp Required:

Select this option if you want to ensure that the Username Token contains a timestamp. If no timestamp is found in the Username Token, a SOAP Fault is returned.

Nonce Required:

Select this option to ensure that the Username Token contains a `<wsse:Nonce>` element. You can use the combination of a timestamp and a nonce to help prevent replay attacks.

Token Verification via Repository

Having validated the timestamp on the token, the Enterprise Gateway can then optionally authenticate the user name

and password contained in the token. The following options are available:

- **No Verification**
No verification of the user name and password is performed. Only the timestamp on the token is validated. This is the default behavior.
- **Verify Username Only**
Only the user name is looked up in the selected repository. If the user name is found in this repository, the user is authenticated. Select the **No password allowed** checkbox to block messages that contain a Username Token with a `<wsse:Password>` element.
- **Verify Username and Password**
The user name is looked up in the selected repository and is only authenticated if the corresponding password matches the one configured in the repository. If you select this option, you must select the type of the password. Both cleartext and digest formats are supported. Select the appropriate option.

Repository Name:

The Enterprise Gateway attempts to authenticate users against the selected **Authentication Repository**. User profiles can be stored in the local store, a database, or an LDAP directory. For details on adding a new repository, and editing or deleting a repository, see the [Authentication Repository](#) tutorial.

Remove enclosing WS-Security element on successful validation:

Select this option if you wish to remove the WS-Security block that contains the Username Token after the token has been successfully authenticated. For example, in the above sample SOAP message that contains two `<wsse:UsernameToken>` elements in two different WS-Security blocks, you could configure the Enterprise Gateway to remove one of these on successful authentication.

Insert WS-Security Username Token

Overview

When a client has been successfully authenticated, the Enterprise Gateway can insert a *WS-Security Username Token* into the downstream message as proof of the authentication event. The `<wsse:UsernameToken>` token enables a user's identity to be inserted into the XML message so that it can be propagated over a chain of Web Services.

A typical example would see a user authenticating to the Enterprise Gateway using HTTP Digest Authentication. After successfully authenticating the user, the Enterprise Gateway inserts a WS-Security Username Token into the message and digitally signs it to prevent anyone from tampering with the token.

The following example shows the format of the `<wsse:UsernameToken>` token:

```
<wsse:UsernameToken wsu:Id="oracle"
  xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
  <wsu:Created>2006.01.13T-10:42:43Z</wsu:Created>
  <wsse:Username>oracle</wsse:Username>
  <wsse:Nonce EncodingType="UTF-8">
    KFIy9LgzhmDPNiQ/B9ZiWKXfEVNvFyn6KWYP+1zVt8=
  </wsse:Nonce>
  <wsse:Password Type="wsse:PasswordDigest">
    CxWj1OMnYj7dddMnU/DrOhY3j4=
  </wsse:Password>
</wsse:UsernameToken>
```

This topic explains how to configure the Enterprise Gateway to insert a WS-Security Username Token after successfully authenticating a user.

General Configuration

To configure general settings, complete the following fields:

Name:

Enter an appropriate name for the filter.

Actor:

The Username Token is inserted into the WS-Security block identified by the specified SOAP *Actor*.

Credential Details

To configure the credential details, complete the following fields:

Username:

Enter the name of the user included in the Username Token. By default, the `authentication.subject.id` message attribute is stored, which contains the name of an authenticated user.

Include Nonce:

Select this option if you wish to include a nonce in the Username Token. A nonce is a random number that is typically used to help prevent replay attacks.

Include Password:

Select this option if you wish to include a password in the Username Token.

Password:

If the **Include Password** checkbox is selected, the Enterprise Gateway inserts the user's password into the generated WS-Security Username Token. It can insert **Clear** or **SHA1 Digest** version of the password, depending on which radio button you select. Oracle recommends the digest form of the password to avoid potential eavesdropping.

You can either explicitly enter the password for this user in the **Password** field, or use a message attribute by selecting the **Wildcard** option, and entering the message attribute in the field provided. By default, the `authentication.subject.password` attribute is used, which contains the password used by the user to authenticate to the Enterprise Gateway.

Advanced

To configure advanced settings, complete the following field:

Indent:

Select this option to add indentation to the generated `UsernameToken` and `Signature` blocks. This makes the security tokens more human-readable.

Insert Timestamp

Overview

In any secure communications protocol, it is crucial that secured messages do not have an indefinite life span. In secure Web Services transactions, a WS-Utility (WSU) Timestamp can be inserted into a WS-Security Header to define the lifetime of the message in which it is placed. A message containing an expired timestamp should be rejected immediately by any Web Service that consumes the message.

Typically, the timestamp contains `Created` and `Expires` times, which combine to define the lifetime of the timestamp. The following shows an example `Timestamp`:

```
<wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">  
  <wsu:Created>2009-03-16T16:32:22Z</wsu:Created>  
  <wsu:Expires>2009-03-16T16:42:22Z</wsu:Expires>  
</wsu:Timestamp>
```

Because the WS-Utility Timestamp is inserted into the WS-Security header block, it is also referred to as a WSS Timestamp. For example, see the [Extract WSS Timestamp](#) filter.

Configuration

Complete the following fields to configure the Enterprise Gateway to insert a timestamp into the message:

Name:

Enter an intuitive name for the filter.

Actor:

The timestamp is inserted into the WS-Security header identified by the SOAP Actor selected here.

Expires In:

Configure the lifetime of the timestamp (and hence the message into which the timestamp is inserted) by specifying the expiration time of the assertion. The expiration time is expressed in days, hours, minutes, and/or seconds.

Layout Type:

In cases where the timestamp must adhere to a particular layout as mandated by the WS-Policy `<Layout>` assertion, you must select the appropriate layout type. A Web Service that enforces a WS-Policy may reject the message if the layout of security elements in the SOAP header is incorrect. Therefore, you must ensure that you select the correct layout type.

Kerberos Client Authentication

Overview

The Enterprise Gateway can be configured to act as a Kerberos Client by obtaining a service ticket for a specific Kerberos Service. The service ticket makes up part of the Kerberos client-side token that is injected into a SOAP message and then sent to the Service. If the Service can validate the token the client will be authenticated successfully.

It is also possible to configure a **Connection** filter (from the Routing category of filters) to authenticate to a Kerberos Service by inserting a client-side Kerberos token into the `Authorization` HTTP header.

Therefore, the **Connection** filter should be used if you want to send the client-side Kerberos token in an HTTP header to the Kerberos Service, while the **Kerberos Client Authentication** filter should be used if you want to send the client-side Kerberos token in a `BinarySecurityToken` block within the SOAP message. Please refer to the [Connection](#) filter help page for more information on authenticating to a Kerberos Service using a client-side Kerberos token.

The **Kerberos Client Authentication** filter is available from the Authentication category of filters. Drag and drop this filter on to the Policy Editor to configure this filter. The sections below describe how to configure the fields on this filter screen.

Kerberos Client

The fields configured on this tab determine how the Kerberos Client obtains a service ticket for a specific Kerberos Service. The following fields must be configured:

Kerberos Client:

The role of the Kerberos Client selected here is twofold: first, it must obtain a Kerberos TGT (Ticket Granting Ticket) and second, it uses this TGT to obtain a service ticket for the **Kerberos Service Principal** selected below. The TGT is acquired at server startup, refresh (for example, when a configuration update is deployed), and when the TGT expires. For more details on configuring Kerberos Clients, see [Kerberos Client](#).

Kerberos Clients are configured globally under the "External Connections" node in the tree view of the Policy Studio. They can then be selected in the **Kerberos Client** dropdown on this tab. It is also possible to add a global Kerberos Client directly from this tab by clicking the **Add** button. Similarly, existing Kerberos Clients can be edited and deleted by selecting the client in the dropdown and clicking the **Edit** and **Delete** buttons, respectively.

Kerberos Service Principal:

The Kerberos Client selected from the dropdown above must obtain a service ticket from the Kerberos TGS (Ticket Granting Server) for the Kerberos Service Principal selected here. The Service Principal can be used to uniquely identify the Service within the Kerberos realm. The TGS will grant a ticket for the selected Principal, which the client can then send to the Kerberos Service. The Principal in the ticket must match the Kerberos Service's Principal in order for the client to be successfully authenticated.

Kerberos Principals are also configured globally under the "External Connections" node in the tree view of the Policy Studio. Alternatively, it is possible to add a global Kerberos Principal directly by clicking the **Add** button. Existing Kerberos Principals can be edited and deleted by clicking the **Edit** and **Delete** buttons, respectively. For more information on configuring Kerberos Principals please refer to the [Kerberos Principals](#) help page.

Kerberos Standard:

When using the **Kerberos Client Authentication** filter to insert Kerberos tokens into SOAP messages in order to authenticate to Kerberos Services, it can do so according to 2 different standards:

- Web Services Security Kerberos Token Profile 1.1
- WS-Trust for Simple and Protected Negotiation Protocol (SPNEGO)

When using the Kerberos Token Profile, the client-side Kerberos token is inserted into a `BinarySecurityToken` block

within the SOAP message. The Kerberos session key may be used to sign and encrypt the SOAP message using the signing and encrypting filters. When this option is selected, the fields on the **Kerberos Token Profile** tab must be configured.

When the WS-Trust for SPNEGO standard is used, a series of requests and responses occur between the Kerberos Client and the Kerberos Service in order to establish a secure context. Once the secure context has been established (using WS-Trust and WS-SecureConversation), a further series of requests and responses are used to produce a shared secret key that can be used to sign and encrypt "real" requests to the Kerberos Service.

If the **WS-Trust for SPNEGO** option is selected it will not be necessary to configure the fields on the **Kerberos Token Profile** tab. However, the **Kerberos Client Authentication** filter must be configured as part of a complicated policy that is set up to handle the multiple request and response messages that are involved in setting up the secure context between the Kerberos Client and Service.

Kerberos Token Profile

The fields on this tab need only be configured if the **Kerberos Token Profile** option has been selected on the **Kerberos Client** tab. This tab allows you to configure where to insert the `BinarySecurityToken` within the SOAP message.

Where to Place BinarySecurityToken:

It is possible to insert the `BinarySecurityToken` inside a named WS-Security Actor/Role within the SOAP message or else an XPath expression can be specified to indicate where the token should be inserted.

Select the **WS-Security Element** radio button if you wish to insert the token into a WS-Security element within the SOAP Header element. You can either select the default "Current actor/role only" option from the dropdown or enter a named actor/role in the field provided. The `BinarySecurityToken` will be inserted into a WS-Security block for the actor/role specified here.

Alternatively, you should select the **XPath Location** option if you want to use an XPath expression to specify where the `BinarySecurityToken` is to be inserted. Click the **Add** button to add a new XPath expression or select an XPath and click the **Edit** or **Delete** buttons to edit or delete an existing XPath expression. Take a look at the [Configuring XPath Expressions](#) help page for more information on how to configure XPath expressions.

Note that it is possible to insert the `BinarySecurityToken` *before* or *after* the node pointed to by the XPath expression. Select either the **Append** or **Before** radio buttons depending on where you want to insert the token relative to the node pointed to by the XPath expression.

BinarySecurityToken Value Type:

Currently, the only supported `BinarySecurityToken` type is the "GSS_Kerberosv5_AP_REQ" type. The selected type will be specified in the generated `BinarySecurityToken`.

Kerberos Service Authentication

Overview

The Enterprise Gateway can act as a Kerberos Service to consume Kerberos tokens sent from a client in either the HTTP header or in the message itself. The client must have obtained a ticket from the Ticket Granting Server (TGS) for this Service.

The **Kerberos Service Authentication** filter is available from the Authentication category of filters. Drag and drop this filter on to the Policy Editor to configure this filter. The sections below describe how to configure the fields on this filter screen.

Kerberos Service

The **Kerberos Service** selected here is responsible for consuming the client's Kerberos token. The client must have obtained a ticket for the Service's Principal name to be able to use the Service.

Kerberos Services are configured globally under the "External Connections" node in the tree view of the Policy Studio. For convenience, it is also possible to add a Kerberos Service by clicking the **Add** button. Existing Services can be edited or deleted by clicking the **Edit** or **Delete** buttons.

Kerberos Standard

Complete the following fields on this tab.

Kerberos Standard:

You must first select one of the following Kerberos standards:

- Kerberos Token Profile
- WS-Trust for SPNEGO
- SPNEGO over HTTP

Note that the Kerberos Service Authentication filter will be used to consume the Kerberos client-side token regardless of whether the token is sent at the message layer (i.e. in the SOAP message) or at the transport layer (in an HTTP header).

Client Token Location for Message-Level Standards:

The Kerberos Service ticket can either be sent in the `Authorization` HTTP header or inside the message itself, for example, inside a `<BinarySecurityToken>` element. Alternatively, it may be contained within a message attribute. Select one of the following options:

- **Message Body:**
Select this option if you expect the Kerberos Service ticket to be contained within the XML message. You must enter an XPath expression to point to the expected location of the Kerberos token.

Some default expressions that point to common locations are available for selection from the dropdown. Otherwise you can add a new XPath expression by clicking the **Add** button. Similarly, existing XPath expressions can be configured by clicking the **Edit** and **Delete** buttons, respectively. Please refer to the [Configuring XPath Expressions](#) for more information on configuring XPath expressions.
- **Message Attribute:**
When using the **WS-Trust for SPNEGO** standard above, the **Consume WS-Trust** filter will place the client-side Kerberos token inside the `ws.trust.spnego.token` message attribute.

Message Level

This section allows you to configure settings that adhere to the message-level standards, i.e. Kerberos Token Profile and WS-Trust for SPNEGO.

Extract Session Keys:

You must check this checkbox if you want to use the Kerberos/SPNEGO session keys to perform a signing or encryption/decryption operation in a subsequent filter. This option is only available when the token is extracted from the message body.

WS-Trust Settings: Key Length:

When using **WS-Trust for SPNEGO**, the **Kerberos Service Authentication** filter will generate a new symmetric key and wrap it using the Kerberos session key. This setting determines the length of the new symmetric key.

WS-Trust Settings: Cache Security Context Session Key:

The service-side may need to cache the session key in order to process (i.e. decrypt and verify) multiple requests from the client.

Transport Level

The options available in this section are specific to Kerberos tokens received over HTTP and are only relevant when the **SPNEGO Over HTTP** option is selected above.

Cookie Name:

The initial handshake between a Kerberos Client and Service can sometimes involve the exchange of a series of request and responses until the secure context has been established. In such cases, an HTTP cookie can be used to keep track of the context across multiple request and response messages. Enter the name of this cookie in the field provided.

Allow Client Challenge:

In some cases, the client may not authenticate (i.e. send the `Authorization` HTTP header) to the Kerberos Service on its first request. The Kerberos Service should then respond with an HTTP 401 response code, instructing the client to authenticate to the server by sending up the `Authorization` header. The client then sends up a second request, this time with the `Authorization` header, which contains the relevant Kerberos token. Check this option if you want to allow this type of negotiation between the client and service.

Client Sends Body Only After Context is Established:

The Kerberos client may wait to mutually authenticate the Kerberos service before sending the body of the message. If this setting is enabled, the Kerberos service will accept the body after the context has been established if the client provides the known cookie. The cookies are cached in the configured cache.

Advanced SPNEGO

Complete the following fields on this tab:

Cache Partially Established Contexts:

In theory, the Kerberos client and service may need to send and receive a number of tokens between each other in order to authenticate to each other. In this case, the **Kerberos Service Authentication** filter will need to cache the partially established context for each client. The contexts will only be cached during the establishment of the context.

In practice however, a single client-side Kerberos token is normally enough to establish a context on the service-side, in which case this setting is not required. This setting applies to the **WS-Trust for SPNEGO** and **SPNEGO over HTTP** standards only.

Kerberos Configuration

Overview

The **Kerberos Configuration** dialog allows you to configure Process-wide Kerberos settings. The most important setting allows you to upload a Kerberos configuration file to the Enterprise Gateway, which contains information about the location of the Kerberos KDC (Key Distribution Center), encryption algorithms and keys, and domain realms to use.

It is also possible to configure trace options for the various APIs used by the Kerberos system, for example, the GSS (Generic Security Services) and SPNEGO (Simple and Protected GSSAPI Negotiation) APIs.

Linux and Solaris platforms ship with a native implementation of the GSS library, which can be leveraged by the Enterprise Gateway. The location of the GSS library can be specified using settings on this dialog.

Kerberos Configuration File - krb5.conf

The Kerberos configuration file (i.e. `krb5.conf`) is required by the Kerberos system to configure the location of the Kerberos KDC, supported encryption algorithms, and default realms.

The file is required by both Kerberos Clients and Services that are configured for the Enterprise Gateway. Kerberos Clients need to know the location of the KDC so that they can obtain a Ticket Granting Ticket (TGT). They also need to know what encryption algorithms to use and to what realm they belong.

A Kerberos Client or Service will know what realm they belong to because either the realm is appended to the principal name after the "@" symbol or, on the other hand, if the realm is not specified in the principal name they are assumed to be in the "default_realm" as specified in the `krb5.conf` file.

Kerberos Services do not need to talk to the KDC to request a TGT. However, they still require the information about supported encryption algorithms and default realms contained within the `krb5.conf` file. There is only one "default_realm" specified in this file, but it is possible to specify a number of additional named realms. The "default_realm" setting can be found in the `[libdefaults]` section of the `krb5.conf` file. It will point to a realm in the `[realms]` section. This setting is not required.

A default `krb5.conf` is displayed in the text area, which can be modified where appropriate and then uploaded to the Enterprise Gateway's configuration by clicking the **OK** button. Alternatively, if you already have a `krb5.conf` file that you want to use, browse to this file using the **Load File** button. The contents of the file will be displayed in the text area and can subsequently be uploaded by clicking the **OK** button.

Note that it is also possible to type directly into the text area to modify the `krb5.conf` contents. Please refer to your Kerberos documentation for more information on the settings that can be configured within the `krb5.conf` file.

Advanced Settings

The checkboxes on this screen allow you to configure various tracing options for the underlying Kerberos API. Trace output is always written to the `/trace` directory of your Enterprise Gateway installation.

Kerberos Debug Trace:

Enables extra tracing from the Kerberos API layer.

SPNEGO Debug Trace:

Turns on extra tracing from the SPNEGO API layer.

Extra Debug at Login:

Provides extra tracing information during login to the Kerberos KDC.

Native GSS Library

The Generic Security Services API (GSS-API) is an API for accessing security services, including Kerberos. Implementations of the GSS-API ship with the Linux and Solaris platforms and can be leveraged by the Enterprise Gateway when it is installed on these platforms. The fields on this tab allow you to configure various aspects of the GSS-API implementation for your target platform.

Note that these are process-wide settings. If use of the native GSS API is selected, it will be used for all Kerberos operations. All Kerberos Clients and Services must therefore be configured to load their credentials natively.

If the native API is used the following will not be supported:

- The SPNEGO mechanism.
- The WS-Trust for SPNEGO standard as it requires the SPNEGO mechanism.
- The SPNEGO over HTTP standard as it requires the SPNEGO mechanism. Note that it is possible to use the KERBEROS mechanism with this protocol, but this would be non-standard.
- Signing and encrypting using the Kerberos session keys.

Use Native GSS Library:

Check this checkbox to use the operating system's native GSS implementation. This option only applies to Enterprise Gateway installations on the Linux and Solaris platforms.

Native GSS Library Location:

If you have opted to use the native GSS library, enter the location of the GSS library in the field provided, for example, `/usr/lib/libgssapi.so`. On Linux platforms, the library is called `libgssapi.so`, while on Solaris this library is called `libgss.so`. It is important to note that this setting is only required when this library is in a non-default location.

Native GSS Trace:

Use this option to enable debug tracing for the native GSS library.

Kerberos Clients

Overview

The Enterprise Gateway can act as a Kerberos client. In doing so, it must authenticate to the Kerberos KDC (Key Distribution Center) as a specific Principal and use the TGT (Ticket Granting Ticket) granted to it to obtain tickets from the TGS (Ticket Granting Service) so that it can authenticate to Kerberos services.

Kerberos Clients can be configured globally under the "External Connections" node in the tree view of the Policy Studio. To configure a Kerberos Client, right-click on the "Kerberos Clients" node in the tree and select the **Add a Kerberos Client** option from the context menu. Enter a name for the Kerberos Client in the **Name** field of the **Kerberos Client** dialog and then complete the following sections where necessary.

Having configured the Kerberos Client, it will be available for selection when configuring other Kerberos-related filters. Make sure to select the **Enabled** checkbox at the bottom of the screen, which is checked by default.

Ticket Granting Ticket Source

Use this section to configure where to obtain the Kerberos client credentials required in order to request service tickets, i.e. Ticket Granting Tickets (TGT) and the session key used in communications with the TGS. The TGT can be retrieved from a cache created as part of a JAAS (Java Authentication and Authorization Service) login, from delegated credentials, or from the native GSS implementation on Linux and Solaris platforms.

It is important to note that depending on what option is selected here, the **Kerberos Principal**, **Password**, and **Keytab File** fields below may or may not be disabled since some of the TGT source options do not require these fields to be configured.

Load via JAAS Login:

By default, the Enterprise Gateway will perform a JAAS login to the Kerberos KDC, after which the credentials will be cached by the Enterprise Gateway and used to acquire service tickets as they are needed. The JAAS login acquires the credentials in one of the following ways:

- *Request from KDC:*
Request a new TGT from the Key Distribution Center. This is performed at server startup, refresh (for example, when a configuration update is deployed), and also when the TGT expires.
- *Extract from Default System Ticket Cache:*
If a TGT has already been obtained out of bounds of the Enterprise Gateway and has been stored in the default system ticket cache, this option can be used to retrieve the TGT from this cache. On a Windows 2000 machine, the TGT will be extracted from the cache using the Local Security Authority (LSA) API. On a Linux/Solaris box, it is assumed that the ticket cache resides in `/tmp/krb5cc_uid`, where the uid is a numeric user identifier. If the ticket cache can not be found in these locations (or if we are running on a different Windows platform), the Enterprise Gateway will look for the cache in `{user.home}{file.separator}krb5cc_{user.name}`.
Note that a system ticket cache may only hold the credentials of a single Kerberos client. If you wish to load the credentials of more than one client from system ticket caches, they will have to be explicitly named using the **Extract from System Cache** option below. Ticket caches can be populated with client credentials using an external utility such as `kinit`.
- *Extract from System Ticket Cache:*
Get the TGT from an explicitly named system ticket cache instead of from the default ticket cache. Browse to the location of the alternative cache using the **Browse** button.

Load from Delegated Credentials:

The Kerberos Client can use a TGT that has been delegated for use by the server and has been already retrieved from a Kerberos Service Authentication filter. In this case, the TGT is extracted from message attributes (i.e. `authentication.delegated.credentials` and `authentication.delegated.credentials.client.name`) that have

been set by a previous Kerberos Service Authentication filter. It is not necessary to configure the **Kerberos Principal** or **Secret Key** fields if this option is selected.

Load via Native GSS Library:

Select this option to have the Native GSS API acquire the client's credentials. The Native GSS API will expect the credentials to be already in a system ticket cache that it can access.

If **Load via Kinit** is not selected, the client credentials must exist in the default system ticket cache. In this case only one Kerberos client can be used within the Enterprise Gateway, as the Enterprise Gateway cannot support accessing credentials natively from the default system ticket cache and other system ticket caches. See above for more details on the location of the default system ticket cache.

If **Load via Kinit** is selected the Enterprise Gateway can support multiple Kerberos clients natively. In this case, the Enterprise Gateway will run `kinit` and create a ticket cache for each client in the `/conf/plugins/kerberos/cache` directory. The Native GSS API will know to acquire the client credentials from these caches.

Important Note:

Note that in order to use the GSS library and optionally the `kinit` tool in this manner you must select to use the native GSS library on the Process-level **Kerberos Configuration** settings. To configure these settings, right-click on the Process in the tree view of the Policy Studio and select the **Kerberos -> Add** option from the context menu. Open the **Native GSS Library** tab of the **Kerberos Configuration** dialog and check the **Use Native GSS Library** checkbox.

Kerberos Principal

A Kerberos Principal is used to assign a unique identity to the Enterprise Gateway for use in the Kerberos environment. Select a previously configured Principal from the dropdown. You can configure Kerberos Principals globally on the **External Connections** tab in the Policy Studio. For more information, see the [Kerberos Principals](#) topic.

Note that the semantics of this field are slightly different depending on what you selected as the TGT source above. If you have opted to retrieve the TGT from the KDC, then you are effectively asking the KDC to issue a TGT for the Principal selected here.

Alternatively, if you have opted to retrieve the TGT from a system ticket cache, then the Principal selected here will be used to lookup the cache in order to retrieve the TGT for this Principal. Similarly, if you want to use the `kinit` utility, the Principal name selected here will be passed as an argument to `kinit`.

Finally, if you wish to retrieve a TGT from delegated credentials, it is not necessary to specify any Principal.

Secret Key

The secret key is used by the principal to talk to the KDC's Authentication Service in order to acquire a TGT. The secret key can either be generated from a password or it can be taken from the principal's keytab file. Once again, the options available here will depend on what has been selected as the source of the TGT.

Password:

A password can only be entered if you have chosen to request the TGT from the KDC. The password will be used when generating the secret key. A secret key is not required at all if the TGT has been already retrieved either from a system ticket cache or from delegated credentials.

It is important to note that the password entered here is stored by default in clear-text form in the Enterprise Gateway's underlying configuration data. If necessary, this can be encrypted using a Passphrase. For more information on encrypting all sensitive Enterprise Gateway configuration data, such as passwords, see [Setting the Encryption Passphrase](#).

Keytab:

When the **Request from KDC** option is selected above, the secret key for the principal can also be extracted from a *Keytab* file, which maps Principal names to encryption keys. Similarly, the `kinit` tool requires a *Keytab* file.

It is possible to load the Principal-to-key mappings into the table by selecting the **Load Keytab** button and then browsing to the location of an existing Keytab file. A new Keytab Entry can be added by clicking the **Add Principal** button. Take a

look at the [Kerberos Keytab Entry](#) help page for more information on configuring the **Keytab Entry** dialog.

A Keytab Entry can be deleted by selecting the entry in the table and clicking on the **Delete Entry** button. It is also possible to export the entire contents of the Keytab table by clicking on the **Export Keytab** button.

It is important to note that the contents of the Keytab table (whether derived from a Keytab file or manually entered using the **Keytab Entry** dialog) are stored in the clear in the Enterprise Gateway's underlying configuration data. The Keytab contents can be stored encrypted, if required, by setting a passphrase. For more details, see [Setting the Encryption Passphrase](#).

When the server starts up it writes the stored Keytab contents out to the `/conf/plugin/kerberos/keytabs/` folder of your Enterprise Gateway installation. Oracle recommends that you configure directory- or file-based access control for this directory and its contents.

Advanced Tab

The following fields can be configured on this tab:

Mechanism:

Select the mechanism used to establish a context between the Enterprise Gateway and the Kerberos service. The Kerberos service must use the same mechanism.

Mutual Authentication:

Request that mutual authentication be carried out during context setup, i.e. the service authenticates back to the client. For the SPNEGO mechanism this must be turned on.

Integrity:

Enables data integrity for GSS operations.

Confidentiality:

Enables data confidentiality for GSS operations.

Credential Delegation:

Request that the initiator's credentials be delegated to the acceptor during context setup. When this option is checked, the acceptor can then assume the initiator's identity and authenticate to other Kerberos services on behalf of the initiator.

Anonymity:

Request that the client's identity is not disclosed to the service.

Replay Detection:

Enables replay detection for the per-message security services after context establishment.

Sequence Checking:

Turns on sequence checking for the per-message security services after context establishment.

Synchronize to Avoid Replays Errors at Service:

In cases where the Kerberos Client is running "under stress" and is attempting to send many requests to a Kerberos Service within a very short (millisecond) timeframe, it is possible that sequential Kerberos *Authenticator* tokens generated by the client will contain identical values for the *ctime* (i.e. the current time on the client's host) and *cusec* (i.e. the microsecond portion of the client's timestamp) fields.

Since Kerberos Service implementations often compare the *ctime* and *cusec* values on successive Authenticator tokens to determine replay attacks, it is possible that the Service will reject Authenticator requests in which the *ctime* and *cusec* fields have the same value.

To avoid situations where the Client may generate successive Authenticator requests (for a particular Service) in which the *ctime* and *cusec* fields are identical, you can select this option to synchronize the creation of the Authenticator requests. The Authenticator request generation will be synchronized using the **Pause Time** field below.

Pause Time:

Specify the time interval (in milliseconds) to wait before generating client-side Authenticator tokens when synchronizing to avoid over-zealous replay detection at the Kerberos Service. This field is only enabled if the **Synchronize to Avoid Replays Errors at Service** checkbox is checked above.

It is important to note here that the default value of 15 milliseconds matches the clock resolution time of operating systems such as Windows. Consult your operating system documentation for more information on the clock resolution for your target system.

Kerberos Services

Overview

The Enterprise Gateway can act as a Kerberos Service. In this case, Kerberos clients must obtain a Kerberos service ticket in order to authenticate to the Kerberos Service exposed by the Enterprise Gateway. Clients must present this ticket to the Enterprise Gateway in order for their requests to be processed, i.e. to be successfully authenticated. It is the Kerberos Service that is responsible for consuming these tickets.

Kerberos Services can be configured globally under the "External Connections" node in the tree view of the Policy Studio. To configure a Kerberos Service, right-click on the "Kerberos Services" node in the tree and select the **Add a Kerberos Service** option from the context menu. The following sections describe how to configure the various fields on the **Kerberos Service** dialog.

Globally configured Kerberos Services are selected by name as part of the Kerberos Service Authentication filter, which is responsible for validating the tickets consumed by the Kerberos Service. Make sure to enter a descriptive name for the service in the **Name** field of the **Kerberos Service** dialog. Please refer to the [Kerberos Service Authentication](#) help page for more information on configuring this filter.

Having configured the Kerberos Service, it will be available for selection when configuring other Kerberos-related filters. Make sure to select the **Enabled** checkbox at the bottom of the screen, which is checked by default.

Kerberos Endpoint Tab

Complete the following fields on this tab:

Kerberos Principal:

Select the name of the principal that will be associated with the Enterprise Gateway. Clients wishing to authenticate to the Enterprise Gateway **must** present a service ticket containing a matching principal name to the Enterprise Gateway.

Kerberos Principals are configured globally under the "External Connections" node in the tree view of the Policy Studio. Right-click on the "Kerberos Principals" node and select the **Add a Kerberos Principal** option from the context menu.

Alternatively, you can select the **Add** button beneath the **Kerberos Principal** dropdown to add a new principal. For more information on configuring a principal, please refer to the [Kerberos Principals](#) help page.

Secret Key:

Use this section to specify the location of the Kerberos Service's secret key, which will be used to decrypt service tickets received from Kerberos clients.

Password:

The Kerberos Service's secret key is originally created for a specific Principal on the KDC. A password is required to generate this key, which can be entered directly into the **Password** field here.

Keytab:

Usually, however, a *Keytab* file is generated, which contains a mapping between a Principal name and that Principal's secret key. The Keytab file can then be loaded into the Enterprise Gateway configuration using the fields provided on this section.

It is possible to load the Principal-to-key mappings into the table by selecting the **Load Keytab** button and then browsing to the location of an existing Keytab file. A new Keytab Entry can be added by clicking the **Add Principal** button. Take a look at the [Kerberos Keytab Entry](#) help page for more information on configuring the **Keytab Entry** dialog.

A Keytab Entry can be deleted by selecting the entry in the table and clicking on the **Delete Entry** button. It is also possible to export the entire contents of the Keytab table by clicking on the **Export Keytab** button.

It is important to note that the contents of the Keytab table (whether derived from a Keytab file or manually entered using

the **Keytab Entry** dialog) are stored in the clear in the Enterprise Gateway's underlying configuration. The Keytab contents can be stored encrypted, if required, by setting a passphrase for the Enterprise Gateway configuration data. For more information on how to do this please refer to the [Setting the Encryption Passphrase](#) help page.

When the server starts up it writes the stored Keytab contents out to the `/conf/plugin/kerberos/keytabs/` folder of your Enterprise Gateway installation. Oracle recommends that you configure directory- or file-based access control for this directory and its contents.

Load via Native GSS Library:

If you have configured the Enterprise Gateway to **Use Native GSS Library** on the Process-level **Kerberos Configuration** settings, you must choose to load the Kerberos Service's secret key from the location preferred by the GSS library. The native GSS library will expect the Kerberos service's secret key to be in the system's default Keytab file. The location of this Keytab file is specified in the "default_keytab_name" setting in the `krb5.conf` file that the native GSS library reads via the `KRB5_CONFIG` environment variable. Note that this Keytab may contain keys for multiple Kerberos services.

Advanced Tab

Configure the following fields on this tab:

Mechanism:

Select the mechanism used to establish a context between this Kerberos service and the Kerberos client. The Kerberos client must use the same mechanism selected here.

Extract Delegated Credentials:

A Kerberos client can set an attribute on the context with the Kerberos service to indicate that they wish to allow the service to act on behalf of the client in subsequent communications. So, for example, this allows the Kerberos service (i.e. the Enterprise Gateway) to assume the identity of the client when communicating with a back-end Kerberos service. In this way the client's credentials are propagated to the back-end service as opposed to the Enterprise Gateway's credentials. This is called *credential delegation*.

In cases where a Kerberos client wishes to delegate its credentials to a Kerberos service you can configure the service to extract the delegated credentials from the context it establishes with the client. Check the **Extract Delegated Credentials** checkbox to extract the client's delegated credentials and store them in the `gss.delegated.credentials` and `gss.delegated.credentials.client.name` message attributes.

The extracted delegated credentials can be forwarded on to the back-end Kerberos service (on behalf of the user) using the Kerberos settings on the **Kerberos Client Authentication** filter or the **Connection** filter. When configuring the **Kerberos Client** used on the **Kerberos Authentication** tab of the **Connection** filter, make sure to select the option to retrieve the TGT (Ticket Granting Ticket) from the extracted delegated credentials (i.e. check the **Extract from delegated credentials** checkbox on the **Kerberos Endpoint** tab).

Take a look at the following help pages for more information on configuring these options:

- [Connection Filter](#)
- [Kerberos Client](#)

Kerberos Principals

Overview

A Kerberos Principal represents a unique identity in a Kerberos system to which Kerberos can assign tickets to access Kerberos-aware services. Principal names are made up of several components separated by the "/" separator. You can also specify a realm as the last component of the name by using the "@" character. If no realm is given, the Principal is assumed to belong to the default realm, as configured in the `krb5.conf` file.

Typically a Principal name comprises 3 parts: the *primary*, the *instance*, and the *realm*. The format of a typical Kerberos v5 Principal name is:

```
primary/instance@realm
```

- **Primary:**
If the Principal represents a user in the system, the primary is the username of the user. Alternatively, for a host, the primary is specified as the string, "host".
- **Instance:**
The instance can be used to further qualify the primary, for example, user/admin@foo.abc.com.
- **Realm:**
This is your Kerberos realm, which is usually a domain name in upper case letters. For example, the machine foo.abc.com is in the ABC.COM Kerberos realm.

Configuration

You can configure Kerberos Principals globally on the **External Connections** tab in the Policy Studio. To configure a Kerberos Principal, right-click the **Kerberos Principals** node, and select the **Add a Kerberos Principal** option from the context menu. Complete the following fields on the **Kerberos Principal** dialog:

Name:

Enter a friendly name for the Kerberos Principal. This name will be available for selection from dropdowns in other Kerberos-related configuration screens in the Policy Studio.

Principal Name:

Enter the name of the Kerberos Principal in this field. The Principal name consists of a number of components separated using the "/" separator. The realm should be specified here if the Principal belongs to either a non-default realm or if a default realm is not specified.

Principal Type:

Select the type of Principal specified in the field above. The following table lists the available Principal Types. It is important to note that the Principal Name Types and their corresponding OIDs are defined in the GSS (General Security Services) API.

Principal Name Type	Explanation
NT_USER_NAME	The Principal name identifies a named user on the local system
KERBEROS_V5_PRINCIPAL_NAME	The Principal name represents a Kerberos version 5 Principal.
NT_EXPORT_NAME	The Principal name represents an exported canonical byte representation of the name, which can be used when searching for the Principal in an ACL (Access Control List), for example.

<i>Principal Name Type</i>	<i>Explanation</i>
NT_HOSTBASED_SERVICE	The Principal name identifies a service associated with a specific host.

It is possible to add new Principal Types by clicking on the **Add** button. The name entered in the **Name** field on the **Kerberos Principal Name OID** must correspond to one of the constant fields defined in the `org.ietf.jgss.GSSName` Java class. Please refer to the Javadocs for the [GSSName](http://java.sun.com/javase/6/docs/api/index.html) [http://java.sun.com/javase/6/docs/api/index.html] class for other allowable name types. Similarly, the corresponding OID for this name type must be entered in the **OID** field of the dialog. Please consult the GSSName Javadoc [here](http://java.sun.com/javase/6/docs/api/index.html) [http://java.sun.com/javase/6/docs/api/index.html] for more information.

Important Note:

It is important to note that OIDs and Principal Type Names should only be changed to reflect changes in the underlying GSS API. Because of this, you should only choose to **Edit** existing **Principal Types** under strict supervision from the Oracle support team.

Kerberos Keytab

Overview

The *Kerberos Keytab* file contains mappings between Kerberos Principal names and DES-encrypted keys that are derived from the password used to login to the Kerberos KDC (Key Distribution Center). The purpose of the Keytab file is to allow the user to access distinct Kerberos Services without being prompted for a password at each Service. Furthermore, it allows scripts and daemons to login to Kerberos Services without the need to store clear-text passwords or for human intervention.

It is important to note that anyone with read access to the Keytab file will have full control of all keys contained within the file. For that reason it is imperative that the Keytab file is protected using very strict file-based access control.

The **Keytab Entry** dialog, which is available from the **Secret Key** section on both the Kerberos Client and Kerberos Service screens after clicking the **Add Principal** button, is essentially a graphical interface to entries in a Kerberos Keytab file.

This dialog allows you to generate keytab entries. Entries may be removed from the Keytab file by clicking the **Delete Entry** button on the Kerberos Client and Kerberos Service screens. You can configure Kerberos Clients and Kerberos Services on the **External Connections** tab in the Policy Studio.

Each key entry in the file is identified by a Kerberos Principal and an encryption type. For this reason, the Keytab file may hold multiple keys for the same principal where each key has a different encryption type. It may also contain keys for several different Principals.

In cases where the Keytab file contains encryption keys for different Principals, at run-time the Kerberos Client or Service will only consider keys mapped to the Principal name selected in the **Kerberos Principal** dropdown on their respective screens.

If the Keytab file contains several keys for the Principal, the Kerberos Client or Service will use the key with the strongest encryption type as agreed during the negotiation of previous messages with the Kerberos Key Distribution Center (KDC).

Configuration

Configure the following fields on the **Keytab Entry** dialog:

Kerberos Principal:

Select an existing Kerberos Principal from the dropdown or add a new one by clicking on the **Add** buttons. You can configure Kerberos Principals globally on the **External Connections** tab in the Policy Studio. For more information on configuring Kerberos Principals, see the [Kerberos Principals](#) topic.

Password:

The password entered here is used to seed the encryption algorithm(s) selected below.

Encryption Types:

The encryption types selected here determine the algorithms used to generate the encryption keys that will be stored in the Keytab file. In cases where the Keytab file contains multiple keys for the Principal, the encryption type is used to select an appropriate encryption key.

To ensure maximum interoperability between Kerberos Clients/Services configured within the Enterprise Gateway and different types of KDC, all encryption types are selected by default. With this configuration, the generated Keytab file will contain a separate encryption key for each encryption type listed here where each key is mapped to the Principal name selected above.

It is important to ensure that the required encryption types exist in the Keytab as defined by settings in the `krb5.conf`. In order for a Kerberos Client to request a Ticket Granting Ticket, it must have at least one key that matches one of the encryption types listed in the "default_tkt_enctypes" setting in the `krb5.conf` file. A Kerberos Service will require a key

of a certain encryption type to be able to decrypt the service ticket presented by a client.

Note that, by default, for Windows 2003 Active Directory, the service ticket is encrypted using the `rc4-hmac` encryption type. However, if the service user has the "Use DES encryption types for this account" option enabled, the `des-cbc-md5` encryption type is used.

CA SOA Security Manager Authentication

Overview

CA SOA Security Manager can authenticate end-users and authorize them to access protected Web resources. When the Enterprise Gateway receives a message containing user credentials, it can forward the message to CA SOA Security Manager where the passed credentials is extracted from the message to authenticate the end-user. When the message has been passed to CA SOA Security Manager, it can authenticate the user by the following methods:

- **XML Document Credential Collector:**
Gathers credentials from the message and maps them to fields within a user directory.
- **XML Digital Signature:**
Validates the X.509 certificate contained within an XML-Signature on the message.
- **WS-Security:**
Extracts user credentials from WS-Security tokens contained within the message.
- **SAML Session Ticket:**
Consumes a SAML session ticket from an HTTP header, SOAP envelope, or session cookie to authenticate the end-user.

By delegating the authentication decision to CA SOA Security Manager, the Enterprise Gateway acts as a Policy Enforcement Point (PEP). It *enforces* the decisions made by the CA SOA Security Manager, which acts a Policy Decision Point (PDP).

Please refer to the Authentication Methods section of the CA SOA Security Manager Policy Configuration Guide for more information on these authentication methods.

Enter a name for the filter in the **Name:** field before configuring the Agent and Message Details sections described below.

Prerequisites

CA SOA Security Manager integration requires CA TransactionMinder SDK version 6.0 or later.

Enterprise Gateway

When adding third-party binaries to the Enterprise Gateway, you must perform the following steps:

1. Add the binary files as follows:
 - Add `.jar` files to the `InstallDir/ext/lib` directory.
 - Add `.dll` files to the `InstallDir\win32\lib` directory.
 - Add `.so` files to the `InstallDir/platform/lib` directory.
2. Restart the Enterprise Gateway.

Policy Studio

When adding third-party binaries to the Policy Studio, you must perform the following steps:

1. Add `.jar` files to the `InstallDir/plugins/thirdparty.runtime.dependencies_6.0.3` directory.
2. Restart the Policy Studio.

Agent Configuration

Name:

Enter a name for this authentication filter in the field provided.

Agent Name:

In order to act as a PEP for the CA SOA Security Manager, the Enterprise Gateway must have been set up as a SOA *Agent* with the Policy Server. Please refer to the **CA SOA Security Manager Agent Configuration Guide** for more information on how to do this.

Select a previously configured agent to connect to SOA Security Manager in the **Agent Name** field. This name *must* correspond with the name of an agent previously configured in the SOA Security Manager **Policy Server**. At runtime, the Enterprise Gateway connects as this agent to a running instance of SOA Security Manager.

For details on how to configure SOA Security Manager connections, see the [CA SOA Security Manager Connection Details](#).

Message Details Configuration

While authenticating the user against CA SOA Security Manager, the user can also be authorized for a specified action on a particular resource. Configure the following fields in the **Message Details** section:

Resource:

Enter the name of the resource for which you want to ensure that the user has access to. By default, the `http.request.uri` message attribute is used, which contains the relative path on which the request was received by the Enterprise Gateway.

Action:

Specify the action that the user is attempting to perform on the specified resource. The Enterprise Gateway will check the user's entitlements in CA SOA Security Manager to ensure that the user is allowed to perform this action on the resource entered above. By default the `http.request.verb` message attribute is used, which stores the HTTP verb used by the client when sending up the message.

Protocol:

Select the protocol used by the client to access the requested resource. Users may have different access rights depending on their roles within the organization. For example, managers may be allowed to FTP to a given resource, but junior employees may only be allowed to GET a resource using HTTP.

This field is pre-populated with the `http.request.protocol` message attribute, which contains the protocol used by the client to send the message to the Enterprise Gateway.

Headers:

In order to carry out further authorization checks on the message, it is possible to forward the HTTP headers associated with the client message to the CA SOA Security Manager. By default, the `http.headers` message attribute is used to ensure that the original client headers are sent to the CA SOA Security Manager.

XmlToolkit.properties File

The `XmlToolkit.properties` file contains default properties used by the SOA agent, such as the URL of the CA SOA Manager, an identifier for the SOA agent, and an indication to the SOA Manager if it should perform fine-grained resource identification or not. The `XmlToolkit.properties` file can be found in the `/lib/modules/soasm` directory of your Enterprise Gateway installation.

```
#Wed Jul 18 15:02:16 BST 2007
WSDMResourceIdentification=yes
WS_UT_CREATION_EXPIRATION_MINUTES=60
```

The following properties are available:

- **WSDMResourceIdentification:**
This value cannot be configured from the Policy Studio GUI and so can only be set directly in the properties file. If this property is set to "no" (or if the properties file cannot be found) only a "coarse-grained" resource identification will be performed on the requested URL. If this value is set to "yes", a "fine-grained" resource identification including the requested URL, Web Service name, and SOAP operation, i.e. [url]/[web service name]/[soap operation].
- **WS_UT_CREATION_EXPIRATION_MINUTES:**
Specifies the WS-Username Token age limit restriction in minutes. This setting helps prevent against replay attacks. The default token age limit is 60 minutes. See the section below for more information on modifying this setting.

Configuring the Username and Password Digest Token Age Restriction:

By default, the WS-Security authentication scheme imposes a 60 minute restriction on the age of Username and Password Digest Tokens to protect against replay attacks.

You can configure a different value for the token age restriction for the Enterprise Gateway by setting the `WS_UT_CREATION_EXPIRATION_MINUTES` parameter in the `XmlToolkit.properties` file for that Enterprise Gateway. To configure the Enterprise Gateway to use a non-default age restriction for Username and Password Token authentication, complete the following steps:

1. Navigate to the `INSTALL_DIR/system/lib/modules/soasm` directory, where `INSTALL_DIR` points to the root of your Enterprise Gateway installation.
2. Open the `XmlToolkit.properties` file in a text editor.
3. Add the following line, where `token_age_limit` specifies the token age limit in minutes:
`WS_UT_CREATION_EXPIRATION_MINUTES=token_age_limit`
4. Save and close the `XmlToolkit.properties` file.
5. Restart the Enterprise Gateway.

Important Points:

- The properties file is written to the `/lib/modules/soasm` directory when a SOA Security Manager Authentication or Authorization filter is loaded at startup, or on server refresh (for example, when a configuration update is deployed), but only if the file does not already exist in this location.
- If the properties file already exists in the `/lib/modules/soasm` directory, the **WSDMResourceIdentification** property is *not* overwritten. In other words, the user is allowed to manually set this property independently of the Policy Studio.
- If the **WSDMResourceIdentification** property does not exist, it is given a default value of "yes" and written to the file.

RADIUS Clients

Overview

The Enterprise Gateway provides support for integration with remote systems over the Remote Authentication Dial In User Service (RADIUS) protocol. RADIUS is a client-server network protocol that provides centralized authentication and authorization for clients connecting to remote services. For more details, see the [RADIUS specification](http://tools.ietf.org/html/rfc2865) [http://tools.ietf.org/html/rfc2865].

To configure a client connection to a remote server over the RADIUS protocol, click the **External Connections** tab in the Policy Studio, and select **RADIUS Clients** -> **Add a RADIUS Client**. This topic explains how to configure the settings the **RADIUS Client** dialog.

For details on how to configure a RADIUS Authentication Repository, see the [Authentication Repository](#) topic.

Configuration

Configure the following fields in the **RADIUS Client** dialog:

- **Name:**
Enter an appropriate name for the RADIUS client to display in the Policy Studio.
- **Host name:**
Enter the host name used by the RADIUS client.
- **Client port:**
Enter the port number used by the RADIUS client.
- **RADIUS servers:**
This field displays a list of configured RADIUS servers. To add a server to the list, click **Add**, and complete the following fields:

Name	Name of the RADIUS server.
Port	Port number used by the RADIUS server.
Secret	Shared secret used to access the RADIUS server.
Response timeout (sec)	Response timeout in seconds before the connection to the server is closed.
Retransmit count	Number of times retransmission is attempted before the connection to the server fails.

Attributes

Overview

The purpose of the filters in the **Attributes** filter group is to extract user attributes from various sources. It is possible to retrieve attributes from the message, an LDAP directory, a database, the **User Store**, HTTP headers, and finally, from a SAML attribute assertion.

Having retrieved a set of user attributes, the Enterprise Gateway then stores them in the `attribute.lookup.list` message attribute, which is essentially a map of name-value pairs. It is the role of the **Attributes** authorization filter to check the value of these attributes in order to authorize the user.

Configuration

The following fields are available on the **Attributes** configuration screen:

Name:

Enter a name for this filter here.

Attributes:

The **Attributes** table lists the checks that the Enterprise Gateway will perform on user attributes stored in the `attribute.lookup.list` message attribute. The following points describe how the Enterprise Gateway carries out the checks listed in the table.

- The entries in the table are OR-ed together so that if any one of them succeeds, the filter will return a "pass" result.
- The attribute checks listed in the table will be run in series until one of them passes.
- It is possible to add a number of attribute-value pairs to a single attribute check by separating them with commas, e.g. "company=oracle, department=engineering, role=engineer".
- If multiple attribute-value pairs are present in a given attribute check, these pairs are AND-ed together so that the overall attribute check will only "pass" if all the attribute-value pairs "pass". So, for example, if the attribute check comprises, "department=engineering, role=engineer", this check will only "pass" if both attributes are found with the correct values in the `attribute.lookup.list` message attribute.

To add an attribute check to the **Attributes** table, click the **Add** button. Attributes can then be entered in the **Add Attributes** dialog.

For attribute checks involving attributes extracted from a SAML attribute assertion, it is necessary to specify the namespace of the attribute as it was given in the assertion. So, for example, the Enterprise Gateway can extract the "role" attribute from the following SAML <Attribute Statement> and store it in the `attribute.lookup.list` map:

```
<saml:AttributeStatement>
  <saml:Attribute Name="role" NameFormat="http://www.company.com">
    <saml:AttributeValue>admin</saml:AttributeValue>
  </saml:Attribute>
  <saml:Attribute Name="email" NameFormat="http://www.company.com">
    <saml:AttributeValue>joe@company.com</saml:AttributeValue>
  </saml:Attribute>
  <saml:Attribute Name="dept" NameFormat="">
    <saml:AttributeValue>engineering</saml:AttributeValue>
  </saml:Attribute>
</saml:AttributeStatement>
```

The "NameFormat" attribute of the <Attribute> gives the namespace of the attribute name. This namespace must be entered (together with a corresponding prefix) in the **Add Attributes** dialog.

For example, to extract the "role" attribute from the SAML attribute statement above, you should enter "pre:role=admin" in the **Attribute Requirement** field. Then you must also map the "pre" prefix to the "http://www.company.com" namespace, as specified by the "NameFormat" attribute in the attribute statement.

Certificate Attributes

Overview

The Enterprise Gateway can authorize access to a Web Service based on the X.509 attributes of an authenticated client's certificate. For example, a simple **Certificate Attributes** filter might only authorize clients whose certificates have a Distinguished Name (DName) containing the following attribute: *O=oracle*. In other words, only "oracle" users are authorized to access the Web Service.

An X.509 certificate consists of a number of fields. The `Subject` field is the one of most relevance to this tutorial. It gives the DName of the client to which the certificate belongs. A DName is a unique name given to an X.500 directory object. It consists of a number of attribute-value pairs called Relative Distinguished Names (RDNs). Some of the most common RDNs and their explanations are as follows:

- CN: CommonName
- OU: OrganizationalUnit
- O: Organization
- L: Locality
- S: StateOrProvinceName
- C: CountryName

For example, the following is the DName of the *sample.p12* client certificate supplied with the Enterprise Gateway:

```
CN=Sample Cert, OU=R&D, O=Company Ltd., L=Dublin 4, S=Dublin, C=IE
```

Using the **Certificate Attributes** filter, it is possible to authorize clients based on, for example, the "CN", "OU", or "C" in the DName.

Configuration

The **X.509 Attributes** table lists a number of attribute checks that will be run against the client certificate. Each entry tests a number of certificate attributes in such a way that the check will only pass if all of the configured attribute values match those in the client certificate. So, in effect, the attributes listed within a single attribute check are AND-ed together.

For example, imagine the following is configured as an entry in the **X.509 Attributes** table:

```
OU=Eng, O=Company Ltd
```

If the Enterprise Gateway receives a certificate with the following DName, this attribute check will pass because *all* the

configured attributes match those in the certificate DName:

```
CN=User1, OU=Eng, O=Company Ltd, L=D4, S=Dublin, C=IE
CN=User2, OU=Eng, O=Company Ltd, L=D2, S=Dublin, C=IE
```

However, if the Enterprise Gateway receives a certificate with the following DName, the attribute check will fail because the attributes in the DName do not match *all* the configured ones (i.e. the "OU" attribute has the wrong value):

```
CN=User1, OU=qa, O=Company Ltd, L=D4, S=Dublin, C=IE
```

The **X.509 Attributes** table can contain several attribute check entries. In such cases, the attribute checks (i.e. the entries in the table) are OR-ed together, so that if any of the checks succeed, the overall **Certificate Attributes** filter succeeds.

So to summarize:

- Attribute values within an attribute check will only succeed if *all* the configured attribute values match those in the DName of the client certificate.
- The filter will succeed if *any* of the attribute checks listed in the **X.509 Attributes** table succeed.

To configure a **Certificate Filter** complete the following fields:

Name:

Enter a name for the filter here.

X.509 Attributes:

To add a new X.509 attribute check, click the **Add button** button. In the **Add X.509 Attributes** dialog, enter a comma-separated list of name-value pairs representing the X.509 attributes and their values, for example, "OU=dev,O=Company".

The new attribute check will appear in the **X.509 Attributes** table. Existing entries can be edited and deleted by clicking the **Edit** and **Remove** buttons respectively.

Check Group Membership

Overview

The **Check Group Membership** filter checks whether the specified Enterprise Gateway User is a member of the specified Enterprise Gateway Group. The User and the Group are both stored in the Enterprise Gateway User Store. For more details, see [Enterprise Gateway Users](#).

Configuration

Configure the following required fields:

Name:

Enter an appropriate name for this filter.

User:

Enter the User name. You can specify this as a property that expands to the value of the specified message attribute at runtime. Defaults to `${authentication.subject.id}`.

Group:

Enter the Group name (for example, `Administrators`). You can specify this as a property that expands to the value of the specified message attribute at runtime.

Possible Paths

The possible paths through this filter are as follows:

<i>Outcome</i>	<i>Description</i>
True	The specified user is a member of the specified group.
False	The specified user is not a member of the specified group.
CircuitAbort	An exception has occurred while executing the filter.

RSA Access Manager Authorization

Overview

RSA Access Manager (formerly known as RSA ClearTrust) provides Identity Management and access control services for Web applications. It centrally manages access to Web applications, ensuring that only authorized users are allowed access to resources.

The Enterprise Gateway's **Access Manager** filter enables integration with RSA Access Manager. This filter can query Access Manager for authorization information for a particular user on a given resource. In other words, the Enterprise Gateway asks Access Manager to make the authorization decision. If the user has been given authorization rights to the Web Service, the request is allowed through to the service. Otherwise, the request is rejected.

Prerequisites

RSA Access Manager integration requires RSA ClearTrust SDK version 6.0.

Enterprise Gateway

When adding third-party binaries to the Enterprise Gateway, you must perform the following steps:

1. Add the binary files as follows:
 - Add .jar files to the *InstallDir/ext/lib* directory.
 - Add .dll files to the *InstallDir\win32\lib* directory.
 - Add .so files to the *InstallDir/platform/lib* directory.
2. Restart the Enterprise Gateway.

Policy Studio

When adding third-party binaries to the Policy Studio, you must perform the following steps:

1. Add .jar files to the *InstallDir/plugins/thirdparty.runtime.dependencies_6.0.3* directory.
2. Restart the Policy Studio.

General Details

Configure the following general setting:

Name:

Enter an appropriate name for the filter.

Connection Details

This section enables you to specify a group of Access Manager servers to connect to in order to authenticate clients. You can select a group of Access Manager servers to provide failover in cases where one or more servers are not available.

Connection Group Type

The Enterprise Gateway can connect to a group of Access Manager **Authorization Servers** or **Dispatcher Servers**. When multiple Access Manager Authorization Servers are deployed for load-balancing purposes, the Enterprise Gateway should first connect to a Dispatcher Server, which returns a list of active Authorization Servers. An attempt is then made to connect to one of these Authorization Servers using round-robin DNS. If the first Dispatcher Server in the Connection Group is not available, the Enterprise Gateway attempts to connect to the Dispatcher Server with the next highest priority in the group, and so on.

If a Dispatcher Server has not been deployed, the Enterprise Gateway can connect directly to an Authorization Server. If

the Authorization Server with the highest priority in the Connection Group is not available, the Enterprise Gateway attempts to connect to the Authorization Server with the next highest priority, and so on.

Select the type of the Connection Group using the **Authorization Server** or **Dispatcher Server** radio button. All servers in the group must be of the same type.

Connection Group:

Select the **Connection Group** to use for authenticating clients. You can add Connection Groups on the **External Connections** tab in Policy Studio. Expand the **Connection Sets** node, right-click **RSA ClearTrust Connection Sets**, and select **Add a Connection Set**. For more details on adding and editing Connection Groups, see the [Configuring Connection Groups](#) topic

Authorization Details

This section describes the resource for which the user is requesting access.

- **Server:**
Enter the name of the server that is hosting the requested resource. The name entered must correspond to a pre-configured Server Name in Access Manager.
- **Resource:**
Enter the name of the requested resource. This resource must also have been pre-configured in Access Manager.

Alternatively, you can enter a property representing a message attribute in the **Resource** field. The Enterprise Gateway expands this property at runtime to the value of the corresponding message attribute. Enterprise Gateway properties take the following format:

```
${message.attribute}
```

The following example of a typical SOAP message received by the Enterprise Gateway shows how this works:

```
POST /services/timeservice HTTP/1.0
Host: localhost:8095
Content-Length: 374
SOAPAction: TimeService
Accept-Language: en-US
Content-Type: text/XML; utf-8

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns1:getTime xmlns:ns1="urn:timeservice">
      </ns1:getTime>
    </soap:Body>
  </soap:Envelope>
```

The following table shows an example of property expansion:

<i>Property</i>	<i>Expanded To</i>
<code>\${http.request.uri}</code>	<code>/services/timeservice</code>

Entrust GetAccess Authorization

Overview

Entrust GetAccess provides Identity Management and access control services for Web resources. It centrally manages access to Web applications, enabling users to benefit from a single sign-on capability when accessing the applications that they are authorized to use.

The Enterprise Gateway's **GetAccess** filter enables integration with Entrust GetAccess. This filter can query GetAccess for authorization information for a particular user for a given resource. In other words, the Enterprise Gateway asks GetAccess to make the authorization decision. If the user has been given authorization rights to the Web Service, the request is allowed through to the Service. Otherwise, the request is rejected.

GetAccess WS-Trust STS

This section configures how the Enterprise Gateway authenticates to the GetAccess WS-Trust Security Token Service (STS). You can configure the Enterprise Gateway to connect to a group of GetAccess STS servers in a round-robin fashion. This provides the necessary failover capability when one or more STS servers are not available.

Configure the following fields:

- **URL Group:**
Select an STS URL group from the drop-down list. This group consists of a number of GetAccess STS Servers to which the Enterprise Gateway round-robins connection attempts. You can add URL groups on the **External Connections** tab in Policy Studio. Expand the **URL Connection Sets** node, right-click **Entrust GetAccess URL Sets**, and select **Add a URL Set**. For more details on adding and editing URL groups, see the [Configuring URL Groups](#) topic.
- **Drift Time:**
Having successfully authenticated to a GetAccess STS server, the STS server issues a SAML authentication assertion and returns it to the Enterprise Gateway. When checking the validity period of the assertion, the specified **Drift Time** is used to account for a possible difference between the time on the STS server and the time on the machine hosting the Enterprise Gateway.
- **WS-Trust STS Attribute Field Name:**
Specify the field name for the `Id` field in the WS-Trust request. The default is `Id`.

GetAccess SAML PDP

When the Enterprise Gateway has successfully authenticated to a GetAccess STS server, it can then obtain authorization information about the end-user from the GetAccess SAML PDP. The authorization details are returned in a SAML authorization assertion, which is then validated by the Enterprise Gateway to determine whether the request should be denied.

Configure the following fields:

- **URL Group:**
Select a SAML PDP URL group in the **URL Group** field. This group consists of a number of GetAccess SAML PDP Servers on which the Enterprise Gateway round-robins authorization requests. You can add URL groups on the **External Connections** tab in Policy Studio. Expand the **URL Connection Sets** node, right-click **Entrust GetAccess URL Sets**, and select **Add a URL Set**. For more details on adding and editing URL groups, see the [Configuring URL Groups](#) topic.
- **Drift Time:**
The specified **Drift Time** is used to account for the possible difference between the time on the GetAccess SAML PDP and the time on the machine hosting the Enterprise Gateway. This comes into effect when validating the SAML

authorization assertion.

- **Resource:**

This is the resource for which the client is requesting access. You can enter a property representing a message attribute, which is looked up and expanded to a value at runtime. Properties have the following format:

`${message.attribute}`

For example, to specify the original path on which the request was received by the Enterprise Gateway as the resource, enter the following property:

`${http.request.uri}`

- **Actor/Role:**

To add the SAML authorization assertion to the downstream message, select a SOAP actor/role to indicate the WS-Security block where the assertion is added. By leaving this field blank, the assertion is not added to the message.

Insert SAML Authorization Assertion

Overview

After successfully authorizing a client, the Enterprise Gateway can insert a Security Assertion Markup Language (SAML) authorization assertion into the SOAP message. Assuming all other security filters in the policy are successful, the assertion will eventually be consumed by a downstream Web Service.

It may be useful to refer to the following example of a signed SAML authorization assertion when configuring this filter.

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://.../soap/envelope/">
  <soap:Header xmlns:wsse="http://.../secext">
    <wsse:Security>
      <saml:Assertion
        xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
        AssertionID="oracle-1056130475340"
        Id="oracle-1056130475340"
        IssueInstant="2003-06-20T17:34:35Z"
        Issuer="CN=Sample User,.....,C=IE"
        MajorVersion="1"
        MinorVersion="0">
        <saml:Conditions
          NotBefore="2003-06-20T16:20:10Z"
          NotOnOrAfter="2003-06-20T18:20:10Z"/>
        <saml:AuthorizationDecisionStatement
          Decision="Permit"
          Resource="http://www.oracle.com/service">
          <saml:Subject>
            <saml:NameIdentifier
              Format="urn:oasis:names:tc:SAML:1.0:assertion#X509SubjectName">
              sample
            </saml:NameIdentifier>
          </saml:Subject>
          </saml:AuthorizationDecisionStatement>
          <dsig:Signature xmlns:dsig="http://.../xmldsig#" id="Sample User">
            <!-- XML SIGNATURE INSIDE ASSERTION -->
          </dsig:Signature>
        </saml:Assertion>
      </wsse:Security>
    </soap:Header>
    <soap:Body>
      <ns1:getTime xmlns:ns1="urn:timeservice">
      </ns1:getTime>
    </soap:Body>
  </soap:Envelope>
```

General Configuration

Configure the following field:

Name:

Enter an appropriate name for the filter.

Assertion Details

Configure the following fields on the **Assertion Details** tab:

Issuer Name:

Select the certificate containing the Distinguished Name (DName) that you want to use as the Issuer of the SAML assertion. This DName is included in the SAML assertion as the value of the `Issuer` attribute of the `<saml:Assertion>` element. For an example, see the sample SAML assertion above.

Expire In:

Specify the lifetime of the assertion in this field. The lifetime of the assertion lasts from the time of insertion until the specified amount of time has elapsed.

Drift Time:

The **Drift Time** is used to account for differences in the clock times of the machine hosting the Enterprise Gateway (that generate the assertion) and the machines that consume the assertion. The specified time is subtracted from the time at which the Enterprise Gateway generates the assertion.

SAML Version:

You can create SAML 1.0, 1.1, and 2.0 attribute assertions. Select the appropriate version from the drop-down list.

Important Note:

SAML 1.0 recommends the use of the <http://www.w3.org/TR/2001/REC-xml-c14n-20010315> XML Signature Canonicalization algorithm. When inserting signed SAML 1.0 assertions into XML documents, it is quite likely that subsequent signature verification of these assertions will fail. This is due to the side effect of the algorithm including inherited namespaces into canonical XML calculations of the inserted SAML assertion that were not present when the assertion was generated.

For this reason, Oracle recommend that SAML 1.1 or 2.0 is used when signing assertions as they both uses the exclusive canonical algorithm <http://www.w3.org/2001/10/xml-exc-c14n#>, which safeguards inserted assertions from such changes of context in the XML document. Please see section 5.4.2 of the [oasis-sstc-saml-core-1.0.pdf](#) and section 5.4.2 of [sstc-saml-core-1.1.pdf](#) documents, both of which are available at <http://www.oasis-open.org>.

Resource:

Enter the resource for which you want to obtain the authorization assertion. You should specify the resource as a URI (for example, <http://www.oracle.com/TestService>). The name of the resource is then included in the assertion.

Action:

You can specify the operations that the user can perform on the resource using the **Action** field. This entry is a comma-separated list of permissions. For example, the following is a valid entry: `read,write,execute`.

Assertion Location

The options on the **Assertion Location** tab specify where the SAML assertion is inserted in the message. By default, the SAML assertion is added to the WS-Security block with the current SOAP actor/role. The following options are available:

Append to Root or SOAP Header:

Appends the SAML assertion to the message root for a non-SOAP XML message, or to the SOAP Header for a SOAP message. For example, this option may be suitable for cases where this filter may process SOAP XML messages or non-SOAP XML messages.

Add to WS-Security Block with SOAP Actor/Role:

Adds the SAML assertion to the WS-Security block with the specified SOAP actor (SOAP 1.0) or role (SOAP 1.1). By default, the assertion is added with the current SOAP actor/role only, which means the WS-Security block with no actor. You can select a specific SOAP actor/role when available from the drop-down list.

XPath Location:

If you wish to insert the SAML assertion at an arbitrary location in the message, you can use an XPath expression to

specify the exact location in the message. You can select XPath expressions from the drop-down list. The default is the `First WSSE Security Element`, which has an XPath expression of `//wsse:Security`. You can add, edit, or remove expressions by clicking the relevant button. For more details, see the [Configuring XPath Expressions](#) topic.

You can also specify how exactly the SAML assertion is inserted using the following options:

- **Append to node returned by XPath expression** (the default)
- **Insert before node returned by XPath expression**
- **Replace node returned by XPath expression**

Insert into Message Attribute:

Specify a message attribute to store the SAML assertion from the drop-down list (for example, `saml.assertion`). Alternatively, you can also enter a custom message attribute in this field (for example, `my.test.assertion`). The SAML assertion can then be accessed downstream in the policy circuit.

Subject Confirmation Method

The settings on the **Subject Confirmation Method** tab determines how the `<SubjectConfirmation>` block of the SAML assertion is generated. When the assertion is consumed by a downstream Web Service, the information contained in the `<SubjectConfirmation>` block can be used to authenticate either the end-user that authenticated to the Enterprise Gateway, or the issuer of the assertion, depending on what is configured.

The following is a typical `<SubjectConfirmation>` block:

```
<saml:SubjectConfirmation>
  <saml:ConfirmationMethod>
    urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
  </saml:ConfirmationMethod>
  <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
    <dsig:X509Data>
      <dsig:X509SubjectName>CN=oracle</dsig:X509SubjectName>
      <dsig:X509Certificate>
        MIICmzCCAY . . . . . mB9CJEw4Q=
      </dsig:X509Certificate>
    </dsig:X509Data>
  </dsig:KeyInfo>
</saml:SubjectConfirmation>
```

The following configuration fields are available on the **Subject Confirmation Method** tab:

Method:

The selected value determines the value of the `<ConfirmationMethod>` element. The following table shows the available methods, their meanings, and their respective values in the `<ConfirmationMethod>` element:

Method	Meaning	Value
Holder Of Key	The Enterprise Gateway includes the key used to prove that the Enterprise Gateway is the holder of the key, or it includes a reference to the key.	<code>urn:oasis:names:tc:SAML:1.0:cm:holder-of-key</code>

Method	Meaning	Value
Bearer	The subject of the assertion is the bearer of the assertion.	urn:oasis:names:tc:SAML:1.0:cm:bearer
SAML Artifact	The subject of the assertion is the user that presented a SAML Artifact to the Enterprise Gateway.	urn:oasis:names:tc:SAML:1.0:cm:artifact
Sender Vouches	Use this confirmation method to assert that the Enterprise Gateway is acting on behalf of the authenticated end-user. No other information relating to the context of the assertion is sent. It is recommended that both the assertion and the SOAP Body must be signed if this option is selected. These message parts can be signed by using the Sign Message filter.	urn:oasis:names:tc:SAML:1.0:cm:bearer

Note that you can also leave the **Method** field blank, in which case no <ConfirmationMethod> block is inserted into the assertion.

Holder-of-Key Configuration:

When you select **Holder-of-Key** as the SAML subject confirmation in the **Method** field, you must configure how information about the key is included in the message. There are a number of configuration options available depending on whether the key is a symmetric or asymmetric key.

Asymmetric Key:

If you want to use an asymmetric key as proof that the Enterprise Gateway is the holder-of-key entity, you must select the **Asymmetric Key** radio button, and then configure the following fields on the **Asymmetric** tab:

- **Private Key from Certificate Store:**
If you want to select a key that is stored in the Certificate Store, select this option, and click the **Signing Key** button. On the **Select Certificate** screen, select the box next to the certificate that is associated with the key that you want to use.
- **Private Key from Message Attribute:**
Alternatively, the key may have already been used by a previous filter in the circuit, for example to sign a part of the message. In this case, the key is stored in a message attribute. You can specify this message attribute in this field.

Symmetric Key:

If you want to use a symmetric key as proof that the Enterprise Gateway is the holder of key, select the **Symmetric Key** radio button, and then configure the fields on the **Symmetric** tab:

- **Generate Symmetric Key:**
If you select this option, the Enterprise Gateway generates a symmetric key, which is then included in the message before it is sent to the client.
- **Symmetric Key in Message Attribute:**
If a previous filter (for example, a **Sign Message** filter) has already used a symmetric key, you can reuse this key as proof that the Enterprise Gateway is the holder-of-key entity. You must enter the name of the message attribute in the field provided.
- **Encrypt using Certificate from Certificate Store:**

When a symmetric key is used, you must assume that the recipient has no prior knowledge of this key. It must, therefore, be included in the message so that the recipient can validate the key. To avoid meet-in-the-middle style attacks, where a hacker could eavesdrop on the communication channel between the Enterprise Gateway and the recipient and gain access to the symmetric key, the key must be encrypted so that only the recipient can decrypt the key. One way of doing this is to select the recipient's certificate from the Certificate Store. By encrypting the symmetric key with the public in the recipient's certificate, the key can only be decrypted by the recipient's private key, to which only the recipient has access. Select the **Signing Key** button and then select the recipient's certificate on the **Select Certificate** dialog.

- **Encrypt using Certificate from Message Attribute:**
Alternatively, if the recipient's certificate has already been used (perhaps to encrypt part of the message) this certificate is stored in a message attribute. You can enter the message attribute in this field.
- **Symmetric Key Length:**
Enter the length (in bits) of the symmetric key to use.
- **Key Wrap Algorithm:**
Select the algorithm to use to encrypt (*wrap*) the symmetric key.

Key Info:

The **Key Info** tab must be configured regardless of whether you have elected to use symmetric or asymmetric keys. It determines how the key is included in the message. The following options are available:

- **Do Not Include Key Info:**
Select this option if you do not wish to include a <KeyInfo> section in the SAML assertion.
- **Embed Public Key Information:**
If this option is selected, details about the key are included in a <KeyInfo> block in the message. You can include the full certificate, expand the public key, include the distinguished name, and include a key name in the <KeyInfo> block by selecting the appropriate boxes. When selecting the **Include Key Name** field, you must enter a name in the **Value** field, and then select the **Text Value** or **Distinguished Name Attribute** radio button, depending on the source of the key name.
- **Put Certificate in Attachment:**
Select this option to add the certificate as an attachment to the message. The certificate is then referenced from the <KeyInfo> block.
- **Security Token Reference:**
The Security Token Reference (STR) provides a way to refer to a key contained in a SOAP message from another part of the message. It is often used in cases where different security blocks in a message use the same key material and it is considered an overhead to include the key more than once in the message.
When this option is selected, a <wsse:SecurityTokenReference> element is inserted into the <KeyInfo> block. It references the key material using a URI to point to the key material and a **ValueType** attribute to indicate the type of reference used. For example, if the STR refers to an encrypted key, you should select **EncryptedKey** from the dropdown, whereas if it refers to a **BinarySecurityToken**, you should select **X509v3** from the dropdown. Other options are available to enable more specific security requirements.

Advanced

Configure the following fields on the **Advanced** tab:

Select Required Layout Type:

WS-Policy and SOAP Message Security define a set of rules that determine the layout of security elements that appear in the WS-Security header within a SOAP message. The SAML assertion is inserted into the WS-Security header according to the layout option selected here. The available options correspond to the WS-Policy Layout assertions of **Strict**, **Lax**, **LaxTimestampFirst**, and **LaxTimestampLast**.

Insert SAML Attribute Statement:

You can specify to insert a SAML attribute statement into the generated SAML authorization assertion. If this option is selected, a SAML attribute assertion is generated using attributes stored in the `attribute.lookup.list` message at-

tribute and subsequently inserted into the assertion. The `attribute.lookup.list` attribute must have been populated previously by an attribute lookup filter for the attribute statement to be generated successfully.

Indent:

Select this method to ensure that the generated signature is properly indented.

Security Token Reference:

The generated SAML authorization assertion can be encapsulated within a `<SecurityTokenReference>` block. The following example demonstrates this:

```
<soap:Header>
  <wsse:Security
    xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"
    soap:actor="oracle">
    <wsse:SecurityTokenReference>
      <wsse:Embedded>
        <saml:Assertion
          xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
          AssertionID="oracle-1056130475340"
          Id="oracle-1056130475340"
          IssueInstant="2003-06-20T17:34:35Z"
          Issuer="CN=Sample User,.....,C=IE"
          MajorVersion="1"
          MinorVersion="0">
          <saml:Conditions
            NotBefore="2003-06-20T16:20:10Z"
            NotOnOrAfter="2003-06-20T18:20:10Z"/>
          <saml:AuthorizationDecisionStatement
            Decision="Permit"
            Resource="http://www.oracle.com/service">
          <saml:Subject>
            <saml:NameIdentifier
              Format="urn:oasis:names:tc:SAML:1.0:assertion#X509SubjectName">
              sample
            </saml:NameIdentifier>
          </saml:Subject>
          </saml:AuthorizationDecisionStatement>
          <dsig:Signature xmlns:dsig="http://.../xmldsig#" id="Sample User">
            <!-- XML SIGNATURE INSIDE ASSERTION -->
          </dsig:Signature>
        </saml:Assertion>
      </wsse:Embedded>
    </wsse:SecurityTokenReference>
  </wsse:Security>
</soap:Header>
```

To add the SAML assertion to a `<SecurityTokenReference>` block as in the example above, select the **Embed SAML assertion within Security Token Reference** option. Otherwise, select **No Security Token Reference**.

RBAC Filter

Overview

Role-Based Access Control (RBAC) is used to protect access to the Enterprise Gateway management services. For example, management services are invoked when a user accesses the server using the Policy Studio or Service Manager, requests the Welcome page (<http://localhost:8090/>), or uses the Traffic Monitor or Real-time Monitoring tools. For more information, see [Configuring Role-Based Access Control](#).

The **RBAC** filter is used in the **Protect Management and Policy Director Interfaces** policy to perform the following tasks:

- Read the user roles from the configured message attribute (for example, `authentication.subject.role`).
- Determine which management service is currently being invoked (which URI, and which SOAP operation and namespace, where applicable).
- Return true if one of the roles has access to the management service currently being invoked, as defined in the `acl.policy` file.
- Otherwise, return false, and the **Return HTTP Error 403: Access Denied (Forbidden)** policy is called. The message content of this filter is shown when a valid user has logged into the browser, but their roles do not give them access to the URI they have invoked. For example, this occurs if a new user is created and they have not yet been assigned any roles.

Configuration

Name:

Enter an appropriate name for this filter.

Role Attribute:

Select or enter the message attribute that contains the user roles. Defaults to `authentication.subject.role`.

SAML Authorization Assertion

Overview

A Security Assertion Markup Language (SAML) authorization assertion contains proof that a certain user has been authorized to access a specified resource. Typically, such assertions are issued by a SAML Policy Decision Point (PDP) when a client requests access to a specified resource. The client must present identity information to the PDP, which ensures that the client does have permission to access the resource. The PDP then issues a SAML authorization assertion stating whether the client is allowed access the resource.

When the Enterprise Gateway receives a request containing such an assertion, it performs the following validation on the assertion details:

- Ensures the resource in the assertion matches that configured in the filter
- Checks the client's access permissions for the resource
- Ensures the assertion has not expired

If the information validates, the Enterprise Gateway authorizes the message for the resource specified in the assertion.

When configuring this filter, it may be useful to refer to the following SAML authorization assertion as an example:

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  MajorVersion="1" MinorVersion="0"
  AssertionID="192.168.0.131.1010924615489"
  Issuer="AA" IssueInstant="2002-03-26 16:23:35">
  <saml:Conditions NotBefore="2002-04-18T09:19:00Z"
    NotOnOrAfter="2003-06-28T09:21:00Z"/>
  <saml:AuthorizationDecisionStatement
    Resource="http://www.abc.org/services/getPrice"
    Decision="Permit">
    <saml:Action>Read</saml:Action>
  </saml:AuthorizationDecisionStatement>
</saml:Assertion>
```

General Settings

Configure the following field on the **SAML Authorization** screen:

Name:

Enter an appropriate name for the filter.

Details

The following fields are available on the **Details** tab:

SOAP Actor/Role:

There may be several authorization assertions contained in a message. You can identify the assertion to validate by entering the name of the SOAP actor/role of the WS-Security header that contains the assertion.

XPath Expression:

Alternatively, you can enter an XPath expression to locate the authorization assertion. You can configure XPath expressions using the **Add**, **Edit** and **Delete** buttons.

SAML Namespace:

Select the SAML namespace that must be used on the SAML assertion for this filter to succeed. If you do not wish to check the namespace, select the `Do not check version` option from the drop-down list.

SAML Version:

Enter the SAML Version that the assertion must adhere to by entering the major version in the first field, followed by the minor version in the second field. For example, for SAML version 2.0, enter 2 in the first field and 0 in the second field.

Drift Time:

The *drift time*, specified in seconds, is used when checking the validity dates on the authorization assertion. The drift time allows for differences between the clock times of the machine on which the assertion was generated and the machine hosting the Enterprise Gateway.

Remove Enclosing WS-Security Element on Successful Validation:

Select this checkbox if you wish to remove the WS-Security block that contains the SAML assertion after the assertion has been successfully validated.

Trusted Issuers

You can use the table on this tab to select the issuers that you consider trusted. In other words, this filter only accepts assertions that have been issued by the selected SAML Authorities.

Click the **Add** button to display the **Trusted Issuers** screen. Select the Distinguished Name of a SAML Authority whose certificate has been added to the Certificate Store, and click **OK**. Repeat this step to add more SAML Authorities to the list of trusted issuers.

Optional Settings

The optional settings enable further examination of the contents of the authorization assertion. The assertion can be checked to ensure that the authorized subject matches a specified value, and that the resource specified in the assertion matches the one entered here.

The Enterprise Gateway can verify that the subject in the SAML assertion (the `<NameIdentifier>`) matches one of the following options:

- **The subject of the authentication filter**
- **The following value:** (for example, `user@oracle.com`)
- **Neither of the above**

The Enterprise Gateway examines the `<Resource>` tag inside the SAML authorization assertion. By default, it compares the `<Resource>` to the `destination.uri` attribute that is set in the policy. If they are identical, this filter passes. Otherwise, it fails.

You can enter a value for the resource in the **Resource** field. The Enterprise Gateway then compares the `<Resource>` in the assertion to this value instead of the `destination.uri` attribute. The filter passes if the `<Resource>` value matches the value entered in the **Resource** field.

SAML PDP Authorization

Overview

The Enterprise Gateway can request an authorization decision from a SAML (Security Assertion Markup Language) PDP (Policy Decision Point) for an authenticated client using the SAML Protocol (SAML). In such cases, the Enterprise Gateway presents evidence to the PDP in the form of some user credentials, such as the Distinguished Name of a client's X.509 certificate.

The PDP decides whether the user is authorized to access the requested resource. It then creates an authorization assertion, signs it, and returns it to the Enterprise Gateway in a SAML Protocol response. The Enterprise Gateway can then perform a number of checks on the response, such as validating the PDP signature and certificate, and examining the assertion. It can also insert the SAML authorization assertion into the message for consumption by a downstream Web Service.

Request Configuration

This section describes how the Enterprise Gateway should package the SAML request before sending it to the SAML PDP.

You can configure a group of SAML PDPs to which the Enterprise Gateway connects in a round-robin fashion if one or more of the PDPs are unavailable. This is known as a SAML PDP URL Set. You can configure a SAML PDP URL Set on the **External Connections** tab in the Policy Studio. Expand the **URL Connection Sets** node, right-click **SAML PDP URL Set**, and select **Add a URL Set**. For more details, see the [Configuring URL Groups](#) topic.

When you have configured a group of SAML PDPs to connect to, you can configure the following general fields:

- **SAML PDP URL Set:**
Select a previously configured SAML PDP URL Set from the drop-down list. You can configure a SAML PDP URL Set on the **External Connections** tab.
- **SAML Version:**
Select the SAML version to use in the SAML request.
- **Signing Key:**
If the SAML request is to be signed, click the **Signing Key** button, and select the appropriate signing key from the Certificate Store.

SAML Subject tab

The details specified on the **SAML Subject** tab describe the *subject* of the SAML assertion. Complete the following fields:

- **Subject Attribute:**
Select the message attribute that contains the name of an authenticated username. By default, the `authentication.subject.id` message attribute is selected, which contains the username of the authenticated user.
- **Subject Format:**
Select the format of the message attribute selected in the **Subject Attribute** field above. You do not need to select a format if the **Subject Attribute** field is set to `authentication.subject.id`.

Subject Confirmation tab

The settings on the **Subject Confirmation** tab determine how the `<SubjectConfirmation>` block of the SAML assertion is generated. When the assertion is consumed by a downstream Web Service, the information contained in the `<SubjectConfirmation>` block can be used to authenticate the end-user that authenticated to the Enterprise Gateway, or the issuer of the assertion, depending on what is configured.

The following is a typical <SubjectConfirmation> block:

```
<saml:SubjectConfirmation>
  <saml:ConfirmationMethod>
    urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
  </saml:ConfirmationMethod>
  <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
    <dsig:X509Data>
      <dsig:X509SubjectName>CN=oracle</dsig:X509SubjectName>
      <dsig:X509Certificate>
        MIICmzCCAY . . . . . mB9CJEw4Q=
      </dsig:X509Certificate>
    </dsig:X509Data>
  </dsig:KeyInfo>
</saml:SubjectConfirmation>
</saml:SubjectConfirmation>
```

You must configure the following fields on the **Subject Confirmation** tab:

Method:

The selected value determines the value of the <ConfirmationMethod> element. The following table shows the available methods, their meanings, and their respective values in the <ConfirmationMethod> element:

Method	Meaning	Value
Holder Of Key	Inserts a <SubjectConfirmation> into the SAML request. The <SubjectConfirmation> contains a <dsig:KeyInfo> section with the certificate of the user selected to sign the SAML request. The user selected to sign the SAML request must be the authenticated subject (authentication.subject.id). Select the Include Certificate option if the signer's certificate is to be included in the SubjectConfirmation block. Alternatively, select the Include Key Name option if only the key name is to be included.	urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
Bearer	Inserts a <SubjectConfirmation> into the SAML request.	urn:oasis:names:tc:SAML:1.0:cm:bearer
SAML Artifact	Inserts a <SubjectConfirmation> into the SAML request.	urn:oasis:names:tc:SAML:1.0:cm:artifact
Sender Vouches	Inserts a <SubjectConfirmation> into the SAML request. The SAML request must be signed by a user.	urn:oasis:names:tc:SAML:1.0:cm:bearer

If the **Method** field is left blank, no `<ConfirmationMethod>` block is inserted into the assertion.

Include Certificate:

Select this option if you wish to include the SAML subject's certificate in the `<KeyInfo>` section of the `<SubjectConfirmation>` block.

Include Key Name:

Alternatively, if you do not want to include the certificate, you can select this option to only include the key name in the `<KeyInfo>` section.

Resource:

Enter the resource for which you want to obtain the authorization assertion. You should specify the resource as a URI (for example, `http://www.oracle.com/TestService`). The name of the resource is then included in the assertion.

Evidence:

The SAML Protocol stipulates that proof of identity in the form of a SAML authentication assertion must be presented to the SAML PDP as part of the SAML request. The Enterprise Gateway can either use an existing SAML authentication assertion that is already present in the message, or it can generate one based on the user that authenticated to it.

Select the **Use SAML Assertion in message** option to include an existing assertion in the SAML request. Specify the actor/role of the WS-Security block where the assertion can be found in the **SOAP Actor/Role** field.

Alternatively, select the **Create SAML Assertion from authenticated client** radio button to generate a new authentication assertion for inclusion in the SAML request. You can sign the newly generated assertion by selecting a key from the drop-down list, which shows all the keys from the Certificate Store.

The specified **Drift Time** is subtracted from the time at which the Enterprise Gateway generates the authentication assertion. This is to account for any possible difference in the times of the machines hosting the SAML PDP and the Enterprise Gateway.

Response

You can configure the Enterprise Gateway to perform a number of checks on the SAML Protocol response from the PDP by examining the contents of various key elements in the authorization assertion.

SOAP Actor/Role:

If the SAML response from the PDP contains a SAML authentication assertion, the Enterprise Gateway can extract it from the response and insert it into the downstream message. The SAML assertion is inserted into the WS-Security block identified by the specified SOAP actor/role.

Drift Time:

The SAML request to the PDP is timestamped by the Enterprise Gateway. To account for differences in the times on the machines running the Enterprise Gateway and the SAML PDP the specified time is subtracted from the time at which the Enterprise Gateway generates the SAML request.

Subject in the Assertion Must Match:

The authorization assertion can be checked to ensure that the authorized subject matches a specified value, and that the resource specified in the assertion matches the one entered here.

The Enterprise Gateway can verify that the subject in the SAML assertion (the `<NameIdentifier>`) matches one of the following options:

- **The subject of the authentication filter**
- **The following value** (for example, `CN=sample, O=Company, C=ie`)
- **Neither of the above**

Tivoli Integration

Overview

Oracle Enterprise Gateway is a dedicated network device for offloading processor-intensive tasks from applications running in general purpose application servers. The Enterprise Gateway performs application networking by routing traffic based on both content and sender. Its patented high performance XML acceleration engine, coupled with acceleration hardware ensures wirespeed network performance.

Tivoli Access Manager is a commonly used product for securing web resources. The Tivoli message filter allows the Enterprise Gateway to leverage existing Access Manager policies, thus avoiding the need to maintain duplicate policies in both products. At runtime, the Tivoli filters can delegate authentication and authorization decision to Access Manager, and can also retrieve user attributes. Therefore, the Enterprise Gateway integrates with Tivoli Access Manager by providing the following functionality:

- Connects to Tivoli Access Manager
- Authenticates a user against Access Manager
- Authorizes a user against Access Manager
- Retrieves user attributes from Access Manager

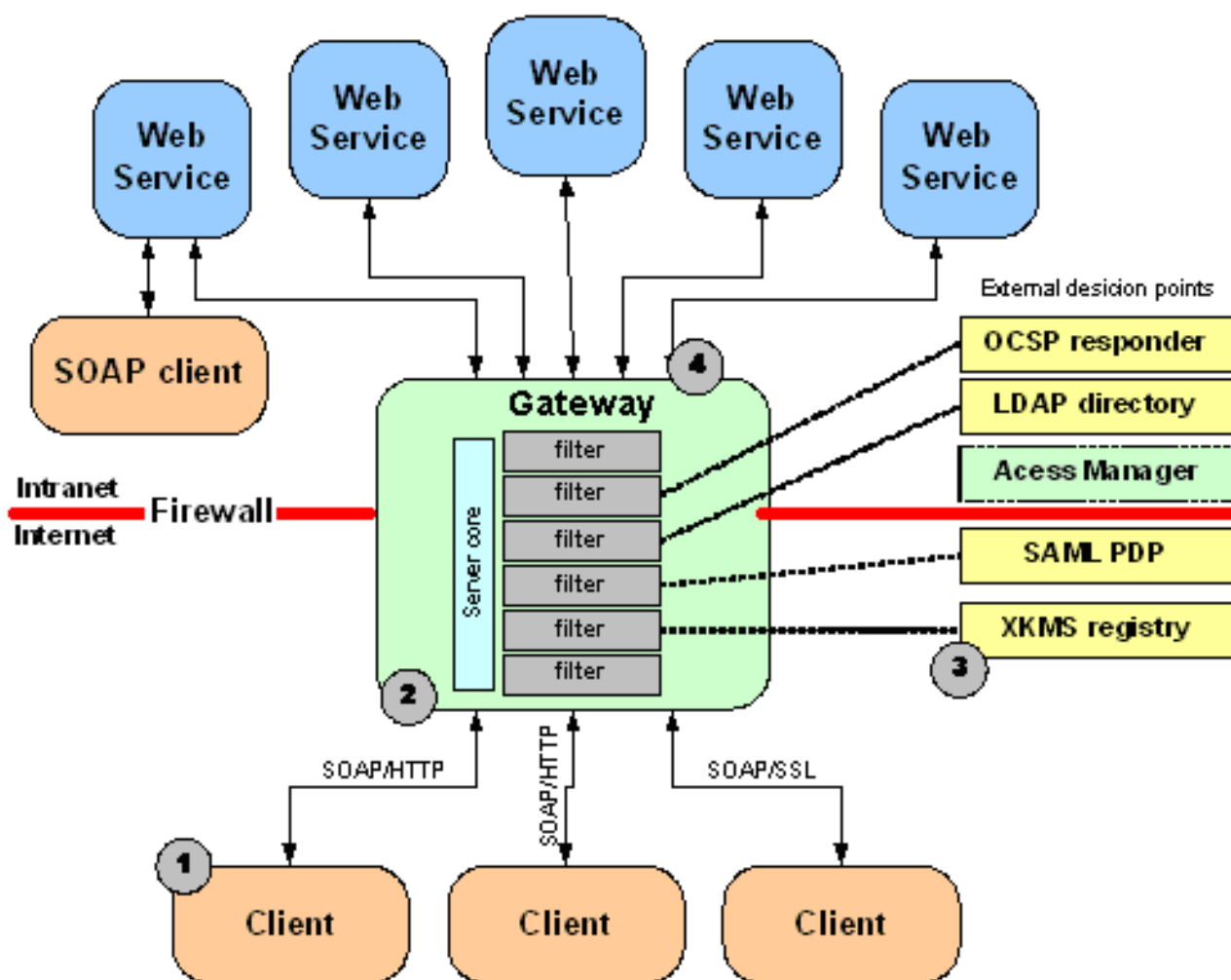
The Enterprise Gateway has been built to integrate with Tivoli Access Manager 6.0.

Integration Architecture

The Oracle Enterprise Gateway contains a set of message filters that directly or indirectly restrict access to Web Services. For example, filters that directly control access include XML-signature verification, CA certificate chain verification, and SAML assertion verification. With this class of filters, policy decisions are made and enforced within the Enterprise Gateway's core engine itself.

On the other hand, filters that indirectly control access offload the policy decision to an external system, such as Tivoli Access Manager. With indirect filters, the policy decision is made by the external system but then enforced by the Enterprise Gateway.

The objective of this integration solution is to implement a message filter that forwards policy decisions to IBM's Tivoli Policy Director/Access Manager. The architecture can be seen in the following diagram.



The following processing stages are executed:

1. The client sends an XML message to the Enterprise Gateway.
2. The Enterprise Gateway dispatches the message to the appropriate policy. The filters configured for that policy are then executed.
3. Assuming that the Tivoli filter is one of the filters that is configured for this policy, the Enterprise Gateway asks Tivoli Access Manager to authenticate, authorize, or retrieve attributes for a given user. Tivoli Access Manager makes its security decision and returns it to the Enterprise Gateway, where the decision is enforced.
4. If Tivoli Access Manager successfully authenticates or authorizes the user, or can retrieve attributes about that user, the message is routed on to the configured target system. Otherwise, the message is blocked and a SOAP Fault is returned to the client.

Prerequisites

IBM Tivoli integration requires the following:

Tivoli API:

IBM Tivoli Access Manager requires the IBM Tivoli Access Manager for e-business Authorization C API.

Enterprise Gateway

When adding third-party binaries to the Enterprise Gateway, you must perform the following steps:

1. Add the binary files as follows:
 - Add `.jar` files to the `InstallDir/ext/lib` directory.
 - Add `.dll` files to the `InstallDir\win32\lib` directory.
 - Add `.so` files to the `InstallDir/platform/lib` directory.
2. Restart the Enterprise Gateway.

Policy Studio

When adding third-party binaries to the Policy Studio, you must perform the following steps:

1. Add `.jar` files to the `InstallDir/plugins/thirdparty.runtime.dependencies_6.0.3` directory.
2. Restart the Policy Studio.

Tivoli Runtime:

The Tivoli Access Manager Runtime must be installed on the machine running the Enterprise Gateway. Note that the Tivoli Access Manager Runtime for Java is **not** required. The Tivoli runtime is not packaged with the Enterprise Gateway product, so the IBM installers need to be run to install the runtime.

Please note that the Tivoli Access Manager Runtime may be installed using the native utilities instead of the Installation Wizard. This is advised so that the IBM Java Runtime 1.4.2 does *not* get installed. The Java Runtime 1.4.2 is required by the Installation Wizard, but not by any of the runtime software.

Tivoli Configuration Files:

The Enterprise Gateway uses information stored in the Tivoli configuration files in order to connect to a Tivoli server. These configuration files can be generated using the **svrsslcfg** command line utility, which is shipped with the Tivoli Access Manager Runtime discussed in the previous section. The generated configuration files are then either uploaded to the Enterprise Gateway using Oracle Policy Studio or manually copied into a location on the Enterprise Gateway's file system.

The following example shows how to run the **svrsslcfg** utility (using Windows file paths):

```
svrsslcfg -config -f "C:\conf\config.conf" -d "C:\conf" -n Enterprise Gateway
-s remote -P passwd -S passwd -r 7777 -h test.vordel.com
```

The following list describes the available arguments:

- **-config**
Create the configuration files required for the Enterprise Gateway to communicate with Tivoli.
- **-f**
The name of the main Tivoli configuration file. This file is generated by the command.
- **-d**
The name of the directory that is to contain the SSL key file (`.kdb`) for the server. This command generates the key file.
- **-n**
The name of the application that is connecting to Tivoli (the Enterprise Gateway).

- **-s**
The mode in which the application (the Enterprise Gateway) runs. The most likely scenario is that the Enterprise Gateway runs remotely.
- **-P**
The administrator's password.
- **-S**
The password for the Enterprise Gateway.
- **-r**
The listening port for the Enterprise Gateway.
- **-h**
The name of the host on which the Enterprise Gateway is running.

After the **svrsslcfg** utility has been run with the **-config** option, the following command must be run:

```
svrsslcfg -add_replica -f "c:\conf\config.conf" -h tivoli.qa.vordel.com
```

The following list describes the available arguments:

- **-add_replica**
Add a Tivoli authorization server replica. The Enterprise Gateway contacts this server to make authorization decisions.
- **-f**
The name of the main Tivoli configuration file. This command adds settings to this file.
- **-h**
The name of the Tivoli authorization server.

Note that the following files are generated after running these commands:

- **c:\conf\config.conf** - The main Tivoli configuration file.
- **c:\conf\Enterprise Gateway.kdb** - The SSL key file.
- **c:\conf\Enterprise Gateway.sth** - The stash file for the SSL key file.
- **c:\conf\Enterprise Gateway.conf.obf** - The database configuration file.

Please note that depending on the version of the Enterprise Gateway you are running, the above file names may or may not have spaces in them.

Please refer to the Tivoli documentation for more information on running these command line utilities.

Creating a Tivoli Object Space:

An object space, user, and ACL (Access Control List) may be created within Tivoli using the **pdadmin** command as follows:

```
> login -a sec_master passw0rd
```

```
> objectspace create /vordel/test "For testing purposes" 9
> user create -gsouser jsmith "cn=John Smith, o=Vordel" "John Smith" Smith passw0rd
> user modify jsmith account-valid yes
> acl create VordelACL
> acl modify VordelACL set user jsmith brT
> acl attach /vordel/test VordelACL
```

The following commands allow you to view the details of the newly added user:

```
> user show jsmith
> objectspace list
> acl show VordelACL
```

Please refer to the Tivoli documentation for more information on running the **pdadmin** utility.

Global Tivoli Configuration

Policy Studio is used to configure all Tivoli connections and settings within the Enterprise Gateway. It can be run from the `/bin` directory of your product installation.

There are two global Tivoli-specific settings that can be configured:

- Tivoli Connections
- Tivoli Repositories

Tivoli Connections:

Tivoli Connections determine how a particular Enterprise Gateway Process connects to an instance of a Tivoli server. Each Process can connect to a single Tivoli server. This connection can be configured by right-clicking on the Process in the tree on the left hand side of the Policy Studio and clicking the **Tivoli** menu option.

Alternatively, you can add a global Tivoli Connection by right-clicking the **Tivoli Connection** node on the **External Connections** tab, and clicking the **Add a Tivoli Connection** option. The newly added connection can then be assigned to a particular Process.

In both cases, the **Tivoli Configuration** dialog is used to add the connection details required for the Enterprise Gateway to connect to the Tivoli server. As stated in the Prerequisites section above, the connection details are stored in the Tivoli configuration files that are generated by the **svrsslcfg** utility. This dialog allows you to upload these files to the Enterprise Gateway.

The Policy Studio can be used to upload the Tivoli configuration files to the machine on which the Enterprise Gateway is running or, alternatively, the configuration files may be copied manually using the tool of your choice onto the Enterprise Gateway's file system. This section now describes how to perform each of these methods in turn.

Complete the following steps to upload a configuration file to the Enterprise Gateway:

1. Enter a name on the **Tivoli Configuration** dialog. Note that a previously configured Tivoli Connection can be selec-

- ted in order to base the new configuration on.
2. Select the **Upload Tivoli configuration files** option.
 3. Select the version of the Tivoli server that this connection connects to. Both Tivoli 5.1 and 6.0 are supported.
 4. Check the **Connection enabled** checkbox if you want to immediately enable the connection. It can be disabled at a later stage by toggling this checkbox. Click the **Next** button.
 5. On the **Upload Tivoli configuration files** screen, click the **Load File** button and browse to the location of the main *Tivoli configuration* file. The contents of this file are then displayed in the text area. Note that any of the details can be edited in the text area at this stage if required.
For example, it may be necessary to change the file locations of the configuration files. This is because when you use the **Upload ...** option, the Enterprise Gateway writes out the files on startup and on server update to the following directory, where `PROCESS_NAME` is Enterprise Gateway and `INSTALL_DIR` refers to the root of your product installation:
`[INSTALL_DIR]\conf\plugin\tivoli\[PROCESS_NAME]`
 Note that spaces are substituted with `-` in the Process name.
 In addition, the Enterprise Gateway names each file as `config.[EXTENSION]`. For example, the directory, `[INSTALL_DIR]\conf\plugin\tivoli\Enterprise Gateway` contains `config.conf`, `config.kdb`, `config.sth`, and `config.conf.obf`. The Enterprise Gateway overwrites these files each time at startup or refresh (for example, when configuration updates are deployed). This means that any changes to the main configuration file must be made using the Policy Studio and not directly to the file on disk.
 6. Click the **Next** button.
 7. Click the **Load File** button and browse to the location of the *Tivoli SSL key file*. Once again, the contents of this file are displayed in the text area. Note that in this case, the (base-64 encoded) SSL keys can *not* be edited in the text area. Click the **Next** button.
 8. Click the **Load File** button and browse to the location of the *Tivoli SSL stash* file. Click the **Next** button.
 9. Click the **Load File** button and browse to the location of the *Tivoli Configuration database configuration* file. Click the **Finish** button to upload all the selected files.

Alternatively, the configuration files may be copied manually onto the Enterprise Gateway's file system. Having done this using some out-of-bounds method, complete the following steps to configure the Enterprise Gateway to pick up the uploaded files:

1. Enter a name on the **Tivoli Configuration** dialog.
2. Select the **Set file location for main Tivoli Configuration file** option and click the **Next** button. Click the **Next** button.
3. Enter the full path to the main Tivoli configuration file on the server's file system in the **Server-side Tivoli configuration** field. Click the **Finish** button.
4. If you have not already manually copied the configuration files on to the Enterprise Gateway's file system you should do so now. Please ensure that the settings contained in the main configuration file that point to other configuration filenames are set correctly. Note that when the **Set file location** option is selected, the Enterprise Gateway does not overwrite the files at startup or refresh time. You may edit the main configuration file directly using an editor of your choice.

Tivoli Repositories:

A Tivoli Repository is used to authenticate clients against a running instance of a Tivoli server. All authentication filters can pass identity credentials to the Tivoli Repository in order to authenticate clients. The Tivoli server decides whether or not to authenticate the client and the Enterprise Gateway subsequently enforces the decision.

Tivoli Repositories can be configured globally by right clicking on the **Tivoli Repositories** node in the tree on the Policy Studio and selecting the **Add** option.

Enter a name for the Repository in the **Repository Name** field on the **Authentication Repository** dialog. Select the **Finish** button to complete the configuration. You may specify Tivoli Connection details from the **Repository Configuration** screen or via the **Settings** button. On the **Tivoli Configuration** dialog, select the Process whose connection details you

want to configure, then follow the steps outlined above in the Tivoli Connections section.

When configuring an authentication filter, you can select this globally configured Tivoli Repository to authenticate clients against. The authentication filter uses the connection details of whatever Process was selected.

Tivoli Authorization

The **Tivoli Authorization** filter can be found in the **Authorization** category of filters. To configure this filter, drag and drop it onto the circuit editor and configure the following fields:

Name:

Enter a name for the Tivoli filter here.

Object Space:

The object space represents the resource for which the client must be authorized. Enter the name of the resource in the **Object Space** field.

You can also enter a property that represents the value of a message attribute. At runtime, the Enterprise Gateway expands the property to current value of the corresponding message attribute.

Properties have the following format:

```
${message.attribute}
```

For example, to specify the original path on which the request was received as the resource, enter the following property:

```
${http.request.uri}
```

Permissions:

Clients can access a resource with a number of permissions such as read, write, execute, and so on. A client is only authorized to access the requested resource if it has the relevant permissions checked in the **Action Bit** table. You can edit existing permissions by clicking the **Edit** button.

Attributes:

It is possible to specify a list of user attributes to retrieve from the Tivoli server. Individual attributes to be retrieved can be added by clicking the **Add** button and entering the attribute name in the dialog. If you want all attributes to be retrieved, simply leave the table blank and select the **Set attributes for SAML Attribute token** option. These attributes can then be made available to the Insert SAML Attribute Assertion filter at a later stage. If you do not require any attribute retrieval, then do not select the **Set attributes for SAML Attribute token** option.

Note:

The permissions for the `primary` action group are available by default. You can also configure custom action groups and make them available for selection in the filter. The groups created here reflect custom groups created on the Tivoli server. To create a new group with custom action bits, click the **Edit** button to display the **Tivoli Action Group** dialog.

Enter a name for the group in the **Name** field. Click the **Add** button to add a new action bit to the group. The **Tivoli Action** dialog is displayed. You must enter an **Action Bit** (for example, `r`) and a **Description** (for example, `Read permission`) for the new action bit. Click the **OK** button on the **Tivoli Action** dialog to return to the **Tivoli Action Group** dialog.

Add as many action bits as required to your new group before clicking **OK** on the **Tivoli Action Group** dialog. The new action bits are then available for selection in the table on the main filter screen.

Tivoli Configuration Files:

As stated earlier, a Tivoli configuration file that contains all the required connection details is associated with a particular Oracle Process (for example, `Enterprise Gateway`). Click the **Settings** button to display the **Tivoli Configuration** dialog.

On the **Tivoli Configuration** dialog, select the Process whose connection details you want to configure, then follow the steps as outlined above in the Tivoli Connections section.

Note that you do not have to configure the Tivoli Connection for each filter or Authentication Repository. The **Settings** button is placed on the filter screen as a convenient way to access the Tivoli Connection settings. The Tivoli Connection

needs to be configured once per Process.

Tivoli Authentication

It is possible to authenticate clients against a Tivoli Access Manager repository. In this way, the Enterprise Gateway can leverage existing Tivoli security policies without the need to duplicate policies across both products.

The Tivoli repository is available from all authentication filters. However, for demonstration purposes, assume that you want to use the HTTP Basic Authentication filter to authenticate a client against a Tivoli Repository using a username and password combination.

Drag the HTTP Basic filter from the Authentication category of filters and drop it onto the circuit editor of the Policy Studio. Complete the following fields:

Name:

Enter a name for the authentication filter here.

Realm:

The **Realm** entered here is presented to the client at the same time as they are entering their username and password. The client is then said to be logging into this realm. It is useful in cases where a given user might belong to many different realms, and so, by presenting the realm to the client, he can specify which realm he wants to log into.

Credential Format:

The username presented to the Enterprise Gateway during the HTTP Basic handshake can be of many formats, usually either username or distinguished name. Since the Enterprise Gateway has no way of inherently telling one format from the other (the client's username could be a DName), it is necessary to specify the format of the credential presented by the client.

Allow Client Challenge:

HTTP Basic Authentication can be configured to work in two ways:

1. **Direct Authentication:**
The client sends up the `Authorization` HTTP Basic Authentication header in its first request to the server.
2. **Challenge-Response Handshake:**
The client does *not* send the `Authorization` header when sending its request to the server (it does not know that the server requires HTTP Basic Authentication). The server responds with an `HTTP 401 response code`, instructing the client to authenticate to the server by sending up the `Authorization` header. The client then sends up a second request, this time including the `Authorization` header and the relevant username and password.

The first case is used mainly for machine-to-machine transactions in which there is no human intervention. The second case is typical of situations where a browser is talking to a Web Server. When the browser receives the `HTTP 401 response` to its initial request, it pops up a dialog to allow the user to enter the username and password combination.

If you wish to force clients to always send the `HTTP Basic Authorization` header to the Enterprise Gateway, disable the **Allow client challenge** checkbox. If, on the other hand, you wish to allow clients to engage in the HTTP Basic Authentication challenge-response handshake with the Enterprise Gateway, make sure this feature is enabled by checking this option.

Remove HTTP Authentication Header:

Select this checkbox to remove the `HTTP Authorization` header from the downstream message. If this option is left unchecked, the incoming `Authorization` header is forwarded onwards to the target system.

Repository Name:

The **Repository Name** field specifies the name of the **Authentication Repository** where all **User** profiles are stored. You can select a previously configured Tivoli repository by simply selecting the name of this repository from the **Repository Name** dropdown. Alternatively, a new repository can be added by clicking the **Add** button.

On the **Authentication Repository** dialog, select `Tivoli Repository` from the **Repository Type** field, and then enter

a name for this type of store in the **Repository Name** field.

Select the **OK** button on the **Authentication Repository** dialog and then **Finish** button on the **HTTP Basic** filter to complete the configuration.

Connections to Tivoli authentication repositories can be configured globally by expanding the **Authentication Repository Profiles** node in the Policy Studio, right-clicking on the **Tivoli Repositories** node and selecting the **Add a New Repository** menu option. The globally configured repository is then available for selection in authentication filters, such as the HTTP Basic authentication filter, as described above.

Tivoli Attribute Retrieval

The **Tivoli Attribute Retrieval** filter can be used in cases where you would like to retrieve user attributes independently from authorizing the user against Tivoli Access Manager. This filter can be found in the **Attributes** category of filters. The following fields should be configured:

Name:

Enter a name for the filter in this field.

User ID:

Enter the ID of a user to retrieve attributes for. You can enter this as a static username, Distinguished Name (DName), or property representing a message attribute. The property is expanded at runtime to the value of the message attribute.

For example, you can enter `${authentication.subject.id}` in this field. This means that the ID of the authenticated user, which is normally a DName, is used to retrieve attributes for. For this to work correctly, an authentication filter must have been configured to run before this filter in the circuit.

Attributes:

It is possible to specify a list of user attributes to retrieve from the Tivoli server. Individual attributes to be retrieved can be added by clicking the **Add** button and entering the attributes in the dialog. If you want all attributes to be retrieved, simply leave the table blank.

Tivoli Configuration Files:

A Tivoli configuration file that contains all the required connection details is associated with a particular Oracle Process (for example, `Enterprise Gateway`). Click the **Settings** button to display the **Tivoli Configuration** dialog.

On the **Tivoli Configuration** dialog, select the Process whose connection details you want to configure, then follow the steps as outlined above in the Tivoli Connections section.

Tivoli Authorization

Overview

Tivoli Access Manager provides authentication and access control services for Web resources. It also stores policies describing the access rights of users.

The Enterprise Gateway can integrate with this product through its Tivoli connector. The Enterprise Gateway Tivoli connector can query Tivoli for authorization information for a particular user on a given resource. In other words, the Enterprise Gateway asks Tivoli to make the authorization decision. If the user has been given authorization rights to the Web Service, the request is allowed through to the Service. Otherwise, the request is rejected.

For details on prerequisites for integration with IBM Tivoli, see the [Tivoli Integration](#) topic.

Adding a Tivoli Client

To add the machine running the Enterprise Gateway as a client of Tivoli, perform the following steps:

1. Open a terminal window on the machine running the Tivoli Authorization Server and Management Server.
2. Start the `pdadmin` tool using the following command, where `oracle` is the password for the Management Server:

```
C:\WINNT> pdadmin -a sec_master -p oracle
```

This starts the `pdadmin` terminal tool.

3. Use the `user create` command to add a user. The parameters are as follows:

```
pdadmin> user create <username> <dn> <cn> <sn> <password>
```

The following is an example where the Enterprise Gateway is running on a machine called `TEST_CLIENT` with an IP address of `192.168.0.100`:

```
pdadmin> user create TEST_CLIENT cn=PdPermission/192.168.0.100,o=Company,c=ie \
PdPermission/192.168.0.100 PdPermission myPass1234
```

Make sure the DName you assign the user is *exactly* identical to the DName in your user's certificate. This includes case and attribute order. Also make sure that you put the IP address or hostname in the CN.

4. Next you must activate the account for the new user. Use the following command:

```
pdadmin> user modify TEST_CLIENT account-valid yes
```

5. Finally, the user must be included in the remote Access Control List (ACL) client list:

```
pdadmin> group modify remote-acl-users add batman
```

The machine running the Enterprise Gateway has now been added as a client to Tivoli.

Adding Users and Web Services to Tivoli

To authorize a user to access a Web Service, you must first add the user to Tivoli as follows:

1. Add the user as before using the `user create` command as follows:

```
pdadmin> user create <username> <dn> <cn> <sn> <password>
```

Ensure that the DN you assign the user is identical to the DName in the user's certificate.

2. Next, you must insert the server that runs your Web Service into Tivoli's object space. Use the following command to do this:

```
pdadmin> object create /Enterprise Gateway/<object-name> <description> 9
```

Note

The 9 parameter indicates that you are adding a Web Resource. In addition, it is the responsibility of the Policy Decision Point (Enterprise Gateway) to map an attempt to access a Web Service to a given object. The Tivoli Authorization server does not contain any mapping between its object space nodes and URLs.

3. Finally, you must create a binding between the user and the object by creating an ACL for the object, and adding the user to that list:

```
pdadmin> acl create <acl-name>
pdadmin> acl modify <acl-name> set user <username> rx
pdadmin> acl attach <object-name> <acl-name>
```

Configuring Tivoli Authorization

Open the **Tivoli Authorization** screen, and configure the following fields:

Name:

Enter a name for the Tivoli filter here.

Object Space:

The object space represents the resource for which the client must be authorized. Enter the name of the resources in the **Object Space** field.

You can also enter properties that represent the values of message attributes. At runtime, the Enterprise Gateway expands the property to the current value of the corresponding message attribute.

Properties have the following format:

```
${message.attribute}
```

For example, to specify the original path on which the request was received by the Enterprise Gateway as the resource, enter the following property:

```
${http.request.uri}
```

Access Method:

Clients can access a resource with a number of permissions such as read, write, execute and so on. A client is only authorized to access the requested resource if he has the relevant permissions checked in the **Access Types** listbox.

Tivoli Connection Settings:

You must enter details on how the Enterprise Gateway should connect to the Tivoli Access Manager in this section. The Enterprise Gateway must have been added to Tivoli as a user for it to connect to the Access Manager. Consult your Tivoli administrator for more information on how to do this.

Important Note:

You must *never* allow more than one the Enterprise Gateway instance use the same account with the Tivoli server.

1. In the **Username** field, enter the username that the Enterprise Gateway uses to connect to the Tivoli server. This is the distinguished name of the Enterprise Gateway's X.509 certificate. You can use %IP% and %HOSTNAME% to generically represent the IP and hostname of the Enterprise Gateway instance. For example, the following entries are both valid:

```
cn=PdPermission/%IP%, o=Company, c=ie
```

```
cn=PdPermission/%HOSTNAME%, o=Company, c=ie
```

This means that multiple the Enterprise Gateway instances, each of which has been set up as a Tivoli user, can share this global setting. For example, one the Enterprise Gateway installation with cn=10.10.10.10 and another with cn=20.20.20.20, can both be represented by cn=PdPermission/%IP% in the **Tivoli Username**. Similarly, an Enterprise Gateway instance with cn=VS_1 and another with cn=VS_2 can both be represented by cn=PdPermission/%HOSTNAME%.

2. In the **Security Master Password** field, enter the master password.
3. In the **Management Server** field, enter the IP address or hostname of the Tivoli Management Server.
4. In the **Authorization Server** field, enter the IP address or hostname of the Tivoli Authorization Server.

Tivoli Authentication Refresh

At start up, the Enterprise Gateway contacts the Tivoli server over SSL, and authenticates using the Security Master password. If you change the Security Master password in the **Policy Studio**, you must stop and start the Enterprise Gateway for this change to be picked up.

CA SOA Security Manager Authorization

Overview

CA SOA Security Manager can authenticate end-users and authorize them to access protected Web resources. The Enterprise Gateway can interact directly with CA SOA Security Manager by asking it to make authorization decisions on behalf of end-users that have successfully authenticated to the Enterprise Gateway. CA SOA Security Manager decides whether to authorize the user, and relays the decision back to the Enterprise Gateway where the decision is enforced. The Enterprise Gateway, therefore, acts as a Policy Enforcement Point (PEP) in this situation, enforcing the authorization decisions made by the CA SOA Security Manager, which acts a Policy Decision Point (PDP).

Important Note:

A CA SOA Security Manager authentication filter must be invoked before a CA SOA Security Manager authorization filter in a given policy. In other words, the end-user must authenticate to CA SOA Security Manager before they can be authorized for a protected resource.

Prerequisites

CA SOA Security Manager integration requires CA TransactionMinder SDK version 6.0 or later.

Enterprise Gateway

When adding third-party binaries to the Enterprise Gateway, you must perform the following steps:

1. Add the binary files as follows:
 - Add `.jar` files to the `InstallDir/ext/lib` directory.
 - Add `.dll` files to the `InstallDir\win32\lib` directory.
 - Add `.so` files to the `InstallDir/platform/lib` directory.
2. Restart the Enterprise Gateway.

Policy Studio

When adding third-party binaries to the Policy Studio, you must perform the following steps:

1. Add `.jar` files to the `InstallDir/plugins/thirdparty.runtime.dependencies_6.0.3` directory.
2. Restart the Policy Studio.

Configuration

Configure the following fields on the **CA SOA Security Manager Authorization** filter:

Name:

Enter an appropriate name for the filter.

Attributes:

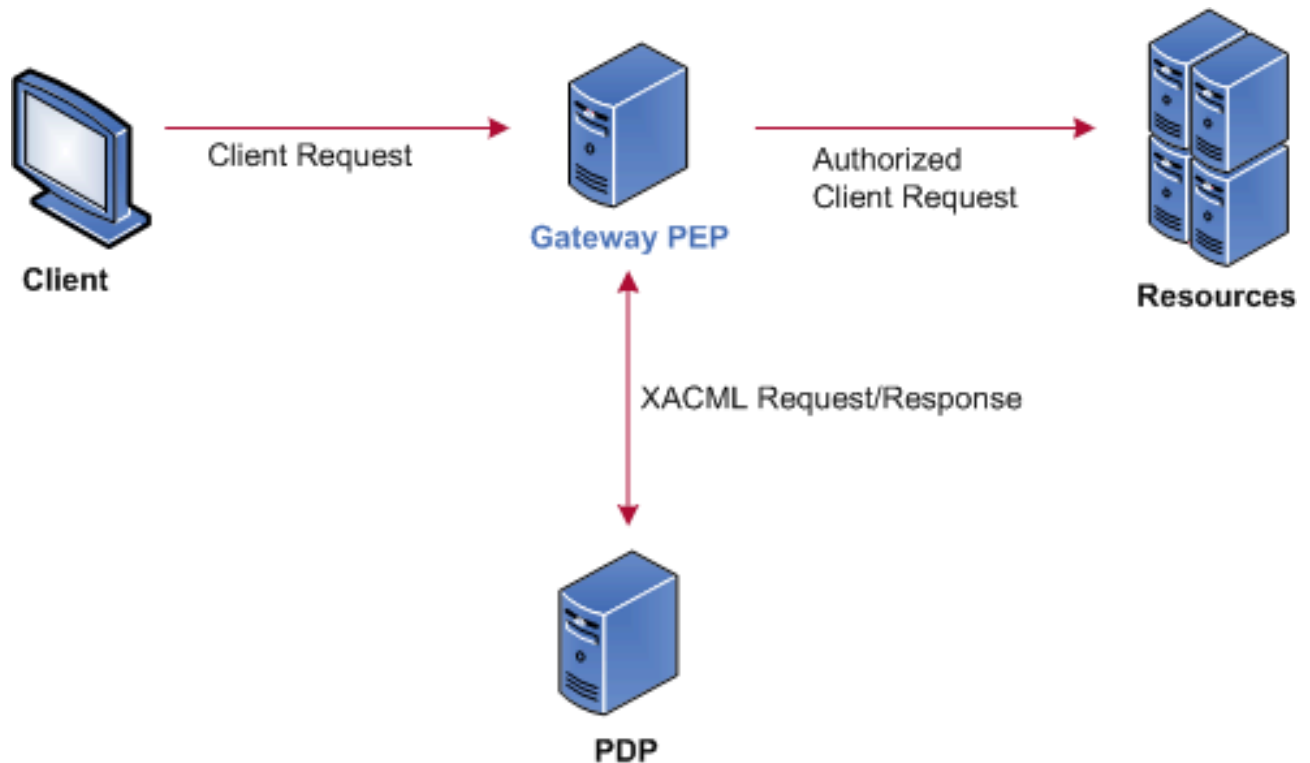
If the end-user is successfully authorized, the attributes listed here are looked up in CA SOA Security Manager, and returned to the Enterprise Gateway. These attributes are stored in the `attributes.lookup.list` message attribute. They can be retrieved at a later stage to generate a SAML attribute assertion.

Select the **Set attributes for SAML Attribute token** checkbox, and click the **Add** button to specify an attribute to fetch from CA SOA Security Manager.

XACML Policy Enforcement Point

Overview

The eXtensible Access Control Markup Language (XACML) Policy Enforcement Point (PEP) filter enables you to configure the Enterprise Gateway to act as a PEP. The Enterprise Gateway intercepts a user request to a resource, and enforces the decision from the Policy Decision Point (PDP). The Enterprise Gateway queries the PDP to see if the user has access to the resource, and depending on the PDP response, allows the filter to pass or fail. Possible PDP responses include `Permit`, `Deny`, `NotApplicable`, and `Indeterminate`.



Workflow

In more detail, when the **XACML PEP** filter is configured in the Enterprise Gateway, the workflow is as follows:

1. The client sends a request for the resource to the XACML PEP filter.
2. The PEP filter stores the original client request, and generates the XACML request.
3. The PEP filter delegates message-level security to the policies configured on the **XACML** tab.
4. The PEP filter routes the XACML request to the PDP using details configured on the **Policies** tab.
5. The PDP decides if access should be granted, and sends the XACML response back to the Enterprise Gateway.
6. The PEP filter validates the response from the PDP.
7. By default, if the response is `Permit`, the PEP filter passes, and the original client request for the resource is authorized, and the policy flow continues on the success path.

Further Information

For more details on XACML, see the XACML specification:

http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf

Example XACML Request

The following example XACML request is used to illustrate the XACML request configuration settings explained in this topic:

```
<Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Subject>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>admin</AttributeValue>
    </Attribute>
    <Attribute AttributeId="department" DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>sysadmin</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>http://localhost:8280/services/echo/echoString</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>read</AttributeValue>
    </Attribute>
  </Action>
  <Environment/>
</Request>
```

General Settings

In the **XACML PEP** filter screen, configure the following general field:

Name:

Enter an appropriate name for this filter.

XACML Settings

The **XACML** tab specifies configuration settings for the generated XACML request. Configure the following fields on this tab:

XACML Version:

Select the XACML version from the list. Defaults to XACML2_0.

Create XACML Request Assertion with the following attributes:

Click the **Add** button on the following tabs to add attributes to the XACML request:

Subject	Represents the entity making the access request (who wants access to the resource). The Subject element can contain multiple Attribute elements, which are used to identify the Subject. Each Attribute element has two attributes: AttributeId and DataType. You can define your own AttributeId or use those provided by the
----------------	--

	XACML specification. For more details on adding attributes, see the next subsection.
Resource	Defines the data, service, or system component that the Subject wants to access. The Resource element contains one or more attributes of the resource to which subjects request access. There can be only one Resource element per XACML request. A specific Resource is identified by the Attribute child element. In the Example XACML Request , the Subject wants to access the following Resource: ht-tp://localhost:8280/services/echo/echoString.
Action	Contains one or more attributes of the action that subjects wish to perform on the resource. There can be only one Action element per XACML request. A specific Action is identified by the Attribute child element. In the Example XACML Request , the Subject wants read access the following Resource: ht-tp://localhost:8280/services/echo/echoString.
Environment	A more complex request context may contain some attributes not associated with the Subject, Resource, or Action. These are placed in an optional Environment element after the Action element.

Adding Attributes

When you click the **Add** button on each tab, the **XACML** dialog is displayed to enable you to add attributes. Complete the following fields on this dialog:

Attribute ID	Enter a custom AttributeId or select one provided by the XACML specification from the list. For example, the XACML special identifiers defined for the Subject include the following: urn:oasis:names:tc:xacml:1.0:subject:authn-locality:dns-name urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address urn:oasis:names:tc:xacml:1.0:subject:authentication-method urn:oasis:names:tc:xacml:1.0:subject:authentication-time urn:oasis:names:tc:xacml:1.0:subject:key-info urn:oasis:names:tc:xacml:1.0:subject:request-time urn:oasis:names:tc:xacml:1.0:subject:session-start-time urn:oasis:names:tc:xacml:1.0:subject:subject-id ... In the Example XACML Request , the first attribute under
---------------------	--

	the Subject element uses the urn:oasis:names:tc:xacml:1.0:subject:subject-id identifier. The next is a custom department attribute. This can be any custom attribute (for example, mail, givenName, or accessList), which can be identified by the XACML policy defined where this request is evaluated.
Value(s)	Click the Add button to add an attribute value. Enter the value in the Add dialog, and click OK . You can add multiple values for a single attribute.
Type	Select the type of data that the AttributeValue element should contain from the list. For example, the set of data types defined in XACML includes the following: http://www.w3.org/2001/XMLSchema#string http://www.w3.org/2001/XMLSchema#boolean http://www.w3.org/2001/XMLSchema#integer http://www.w3.org/2001/XMLSchema#double http://www.w3.org/2001/XMLSchema#time http://www.w3.org/2001/XMLSchema#date http://www.w3.org/2001/XMLSchema#dateTime http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration http://www.w3.org/2001/XMLSchema#anyURI http://www.w3.org/2001/XMLSchema#hexBinary ... In the Example XACML Request , the Attributes are of type http://www.w3.org/2001/XMLSchema#string .
Issuer	Specify an optional issuer for the attribute. For example, this may be a Distinguished Name, or some other identifier agreed with the issuer.

AuthzDecisionQuery Settings

This section enables you to configure settings for the Authorization Decision Query, which is sent in the XACML request to the PDP. Complete the following fields in this group:

Decision based on external XACML attributes	If this is selected, the authorization decision must be made based only on the information contained in the XACML Authz Decision Query, and external XACML attributes must not be used. If this is unselected, the authorization decision can be made based on XACML attributes not contained in the XACML Authz Decision Query. This is unselected by default, which is equivalent to the following setting in the XACML Authz Decision Query: <InputContextOnly value="false">
Return Context	If this is selected, the PDP must include an <code>xacmlcontext:Request</code> instance in the <code>XACMLAuthzDecision</code> statement in the <code>XACMLAuthzDecision</code> response. The <code>xacmlcontext:Request</code> instance must include all attributes supplied by the PEP in the <code>xacml-sample:XACMLAuthzDecisionQuery</code> used to make the authorization decision. If this is unselected, the PDP must not

	include an <code>xacmlcontext:Request</code> instance in the <code>XACMLAuthzDecision</code> statement in the <code>XACMLAuthzDecision</code> response. This is unselected by default, which is equivalent to the following setting in the XACML request: <code><ReturnContext value="false"></code>
Combine Policies	If this is selected, the PDP must insert all policies passed in the <code>xacml:samlp:XACMLAuthzDecisionQuery</code> into the set of policies or policy sets that define the PDP. If this is unselected, there must be no more than one <code>xacml:Policy</code> or <code>xacml:PolicySet</code> passed in the <code>xacml-samlp:XACMLAuthzDecisionQuery</code> . This is selected by default, which is equivalent to the following setting in the XACML request: <code><CombinePolicies value="true"></code>

XACML Message Security

This section enables you to delegate message-level security to the configured custom security policies. Complete the following fields in this group:

XACML Request Security	Click the browse button on the right, select a policy in the XACML request security policy dialog, and click OK .
XACML Response Security	Click the browse button on the right, select a policy in the XACML response security policy dialog, and click OK .

XACML Response:

Select the **Required response decision** from the PDP that is required for this **XACML PEP** filter to pass. Defaults to `Permit`. Possible values are as follows:

- `Permit`
- `Deny`
- `Indeterminate`
- `NotApplicable`

Routing Settings

The **Routing** tab enables you to specify configuration settings for routing the XACML request to the PDP. You can specify a direct connection to the PDP using a URL. Alternatively, if the routing behavior is more complex, you can delegate to a custom routing policy, which takes care of the added complexity.

Use the following URL:

If you wish to route XACML requests to a URL, select this option, and enter the **URL**. You can also specify the URL as a property so that the URL is built dynamically at runtime from the specified message attributes. For example, `${host}:${port}`, or `${http.destination.protocol}://${http.destination.host}:${http.destination.port}`.

In both cases, you can configure the connection details (for example, SSL and other authentication schemes) using the fields in the **Connection Details** group below. For more details, see the [Connect to URL](#) topic.

Delegate routing to the following policy:

If you wish to use a dedicated routing policy to send XACML requests to the PDP, select this option. Click the browse button next to the **Routing Policy** field. Select the policy that you want to use to route XACML requests, and click **OK**.

Advanced Settings

Configure the following settings on the **Advanced** tab:

SOAP Settings:

The available SOAP settings are as follows:

SOAP version required	Specifies the SOAP version required when creating the XACML request message. The available options are as follows: <ul style="list-style-type: none"> SOAP1_1 SOAP1_2 NONE Defaults to SOAP1_1.
SOAP Operation	Specifies the SOAP operation name used in the XACML request message. Defaults to <code>XACMLAuthzDecisionQuery</code> .
Prefix	Specifies the prefix name used in the XACML request message. Defaults to <code>xacml-samlp</code> .
Namespace	Specifies the namespace used in the XACML request message. Defaults to <code>urn:oasis:xacml:2.0:saml:protocol:schema:os</code> .
SOAP Action	You can specify an optional <code>SOAPAction</code> field used in the XACML request header to indicate the intent of the request message.

Advanced Settings:

The available advanced settings are as follows:

Store and restore original message	Specifies whether to store the original client request before generating the XACML request, and then to restore the original client request after access is granted. This option is selected by default.
Split subject attributes into individual elements	Specifies whether to split <code>Subject</code> attributes into individual elements in the XACML request. This option is not selected by default.
Split resource attributes into individual elements	Specifies whether to split <code>Resource</code> attributes into individual elements in the XACML request. This option is not selected by default.

SiteMinder Certificate Authentication

Overview

CA SiteMinder can authenticate end-users and authorize them to access protected Web resources. When the Enterprise Gateway retrieves an X.509 certificate from a message or during an SSL handshake, it can authenticate to SiteMinder on behalf of the user using the certificate. SiteMinder decides whether the user should be authenticated, and the Enterprise Gateway then enforces this decision.

Prerequisites

CA SiteMinder integration requires CA SiteMinder SDK version 12.0-sp1-cr005 or later.

Enterprise Gateway

When adding third-party binaries to the Enterprise Gateway, you must perform the following steps:

1. Add the binary files as follows:
 - Add `.jar` files to the `InstallDir/ext/lib` directory.
 - Add `.dll` files to the `InstallDir\win32\lib` directory.
 - Add `.so` files to the `InstallDir/platform/lib` directory.
2. Restart the Enterprise Gateway.

Policy Studio

When adding third-party binaries to the Policy Studio, you must perform the following steps:

1. Add `.jar` files to the `InstallDir/plugins/thirdparty.runtime.dependencies_6.0.3` directory.
2. Restart the Policy Studio.

Configuration

Configure the following fields:

Name:

Enter an appropriate name for the filter.

Agent Name:

Select a previously configured agent to connect to SiteMinder in the **Agent Name** field. This name *must* correspond with the name of an agent previously configured in the SiteMinder **Policy Server**.

At runtime, the Enterprise Gateway connects as this agent to a running instance of SiteMinder. For details on how to configure a SiteMinder connection, see the [SiteMinder/SOA Security Manager Connection](#) topic.

Resource:

Enter the name of the protected resource for which the end-user must be authenticated. You can enter a property representing a message attribute, which is expanded to a value at runtime. Properties have the following format:

```
${message.attribute}
```

For example, to specify the original path on which the request was received by the Enterprise Gateway as the resource, enter the following property:

```
${http.request.uri}
```

Action:

The end-user must be authenticated for a specific action on the protected resource. By default, this action is taken from the HTTP verb used in the incoming request. You can use the following property to get the HTTP verb:

```
${http.request.verb}
```

Alternatively, any user-specified value can be entered.

Single Sign-On Token:

When a client has been authenticated for a given resource, SiteMinder can generate a *single sign-on token* and return it to the client. The client can then pass this token with future requests to the Enterprise Gateway. When the Enterprise Gateway receives such a request, it can validate the token using the **SiteMinder Session Validation** filter to authenticate the client. In other words, the client is authenticated for the entire lifetime of the token. As long as the token is still valid, the Enterprise Gateway does not need to authenticate the client against SiteMinder for every request, which increases throughput considerably.

In this section, you can instruct SiteMinder to generate a single sign-on token. The Enterprise Gateway can then store this token in a user-specified message attribute. By default, the token is stored in the `siteminder.session` message attribute.

Typically, the token is copied to the `attribute.lookup.list` message attribute using the **Copy / Modify Attributes** filter, before being inserted into a SAML attribute statement using the **Insert SAML Attribute Assertion** filter. The attribute statement is then returned to the client for use in subsequent requests.

Select the **Create single sign-on token** checkbox to instruct SiteMinder to generate the single sign-on token. Enter the name of the message attribute where the token is stored in the field provided.

SiteMinder Session Validation

Overview

CA SiteMinder can authenticate end-users and authorize them to access protected Web resources. When the Enterprise Gateway has authenticated successfully to SiteMinder on behalf of a user using the [SiteMinder Certificate Authentication](#) filter, SiteMinder can issue a *single sign-on* token and return it to the Enterprise Gateway. Typically, the Enterprise Gateway inserts this token into a SAML attribute assertion or an HTTP Header, and returns it to the client.

The client then sends the single-sign on token in subsequent requests to the Enterprise Gateway. The Enterprise Gateway extracts the single-sign on token from the message payload or HTTP headers, and stores it in a message attribute, usually the `siteminder.session` attribute.

The Enterprise Gateway can then use the **SiteMinder Session Validation** filter to ensure that the token is still valid, and hence, that the user is still authenticated. This means that the Enterprise Gateway does not have to authenticate every request to SiteMinder. By validating the token, the user can be authenticated, and therefore, unnecessary round-trips to SiteMinder can be avoided.

Prerequisites

CA SiteMinder integration requires CA SiteMinder SDK version 12.0-sp1-cr005 or later.

Enterprise Gateway

When adding third-party binaries to the Enterprise Gateway, you must perform the following steps:

1. Add the binary files as follows:
 - Add `.jar` files to the `InstallDir/ext/lib` directory.
 - Add `.dll` files to the `InstallDir\win32\lib` directory.
 - Add `.so` files to the `InstallDir/platform/lib` directory.
2. Restart the Enterprise Gateway.

Policy Studio

When adding third-party binaries to the Policy Studio, you must perform the following steps:

1. Add `.jar` files to the `InstallDir/plugins/thirdparty.runtime.dependencies_6.0.3` directory.
2. Restart the Policy Studio.

Configuration

Configure the following fields on the **SiteMinder Session Validation** screen:

Name:

Enter an appropriate name for the filter.

Agent Name:

Select the name of the agent to connect to SiteMinder in the **Agent Name** field. This name *must* correspond to the name of an agent previously configured in the SiteMinder **Policy Server**.

At runtime, the Enterprise Gateway connects as this agent to a running instance of SiteMinder. For details on how to configure a SiteMinder connection, see the [SiteMinder/SOA Security Manager Connection](#) topic.

Resource:

Enter the name of the protected resource for which the end-user must be authenticated. You can enter a property representing a message attribute, which is expanded to a value at runtime. Properties have the following format:

`${message.attribute}`

For example, to specify the original path on which the request is received by the Enterprise Gateway as the resource, enter the following property:

`${http.request.uri}`

Action:

The end-user must be authenticated for a specific action on the protected resource. By default, this action is taken from the HTTP verb used in the incoming request. You can use the following property to get the HTTP verb:

`${http.request.verb}`

Alternatively, any user-specified value can be entered here.

Message attribute containing session:

Enter the name of the message attribute that contains the single sign-on token generated by SiteMinder. By default, the token is stored in the `siteminder.session` message attribute, but can be stored in any attribute.

SiteMinder Logout

Overview

When the Enterprise Gateway authenticates to CA SiteMinder on behalf of a user, SiteMinder can issue a *single sign-on* token as evidence of the authentication event. The token is eventually returned to the client, which can then use it in subsequent requests to the Enterprise Gateway.

Instead of authenticating the client against SiteMinder for every request, the Enterprise Gateway need only validate the token. If the token validates, the client can be considered authenticated. If the token does not validate, the client is not considered authenticated.

You can use the **SiteMinder Logout** filter to invalidate a single sign-on token that was previously issued by SiteMinder. When the token has been invalidated, the client is no longer be considered authenticated.

Note:

You must have already validated the session before calling the **SiteMinder Logout** filter in your circuit. For more details, see the [SiteMinder Session Validation](#) topic.

Prerequisites

CA SiteMinder integration requires CA SiteMinder SDK version 12.0-sp1-cr005 or later.

Enterprise Gateway

When adding third-party binaries to the Enterprise Gateway, you must perform the following steps:

1. Add the binary files as follows:
 - Add `.jar` files to the `InstallDir/ext/lib` directory.
 - Add `.dll` files to the `InstallDir\win32\lib` directory.
 - Add `.so` files to the `InstallDir/platform/lib` directory.
2. Restart the Enterprise Gateway.

Policy Studio

When adding third-party binaries to the Policy Studio, you must perform the following steps:

1. Add `.jar` files to the `InstallDir/plugins/thirdparty.runtime.dependencies_6.0.3` directory.
2. Restart the Policy Studio.

Configuration

Enter a name for the filter in the **Name** field of the **SiteMinder Logout** screen.

SiteMinder Authorization

Overview

CA SiteMinder can authenticate end-users and authorize them to access protected Web resources. The Enterprise Gateway can interact directly with SiteMinder by asking it to make authorization decisions on behalf of end-users that have successfully authenticated to Enterprise Gateway. The Enterprise Gateway then enforces the decisions made by SiteMinder.

Important Note:

A SiteMinder authentication filter must be configured before a SiteMinder authorization filter is created. In other words, end-users must authenticate to SiteMinder before they can be authorized.

Prerequisites

CA SiteMinder integration requires CA SiteMinder SDK version 12.0-sp1-cr005 or later.

Enterprise Gateway

When adding third-party binaries to the Enterprise Gateway, you must perform the following steps:

1. Add the binary files as follows:
 - Add `.jar` files to the `InstallDir/ext/lib` directory.
 - Add `.dll` files to the `InstallDir\win32\lib` directory.
 - Add `.so` files to the `InstallDir/platform/lib` directory.
2. Restart the Enterprise Gateway.

Policy Studio

When adding third-party binaries to the Policy Studio, you must perform the following steps:

1. Add `.jar` files to the `InstallDir/plugins/thirdparty.runtime.dependencies_6.0.3` directory.
2. Restart the Policy Studio.

Configuration

Configure the following fields on the **SiteMinder Authorization** filter:

Name:

Enter an appropriate name for the filter.

Attributes:

If the end-user is successfully authorized, the attributes listed here are returned to the Enterprise Gateway and stored in the `attribute.lookup.list` message attribute. They can then be used by subsequent filters in a policy to make decisions on their values. Alternatively, they can be inserted into a SAML attribute assertion so that the target service can apply some business logic based on their values (for example, if role is CEO, escalate the request, and so on).

Select the **Retrieve attributes from CA SiteMinder** checkbox, and click the **Add** button to specify an attribute to fetch from SiteMinder. If you select the **Retrieve attributes from CA SiteMinder** checkbox, and do not specify attribute names to be retrieved, all attributes returned by SiteMinder are added to the `attribute.lookup.list` message attribute.

SiteMinder/SOA Security Manager Connection

Overview

This topic explains how to create connections to CA SiteMinder and CA SOA Security Manager. On the **External Connections** tab in the Policy Studio, right-click the **SiteMinder/SOA Security Manager Connection** node, and select **Add CA SiteMinder Connection** or **Add CA SOA Security Manager Connection**.

You can specify how the Enterprise Gateway connects to CA SiteMinder using the **SiteMinder Connection Details** dialog. You can specify how the Enterprise Gateway connects to CA SOA Security Manager using the **CA SOA Security Manager Connection Details** dialog. In both cases, the Enterprise Gateway must have already been set up as an agent in the **CA Policy Server**.

The connection details to be configured for the Enterprise Gateway are the same for both SiteMinder and SOA Security Manager, with an additional setting for SOA Security Manager.

SiteMinder and SOA Security Manager Connection Details

This section describes details that are common to both SiteMinder and CA SOA Security Manager connections.

Agent Name:

Enter the name of the agent to connect to SiteMinder or SOA Security Manager in the **Agent Name** field. This name *must* correspond to the name of an agent previously configured in the **CA Policy Server**.

Agent Configuration Object:

The name entered must match the name of the Agent Configuration Object (ACO) configured in the **CA Policy Server**. The Enterprise Gateway currently does not support any features represented by the ACO parameters except for the `PersistentIPCheck` setting. For example, the Enterprise Gateway ignores the `DefaultAgent` parameter, and uses the agent value it collects separately during agent registration.

When the `PersistentIPCheck` ACO parameter is set to `yes`, this instructs the Enterprise Gateway to compare the IP address from the last request (stored in a persistent cookie) with the IP address in the current request to see if they match. If the IP addresses do not match, the Enterprise Gateway rejects the request. If this parameter is set to `no`, this check is disabled.

Connection Details:

The Enterprise Gateway host machine must be registered with SiteMinder or SOA Security Manager. To register the host machine, you must use the `smregghost` tool on the Enterprise Gateway machine. The `smregghost` tool creates a file called `SmHost.conf`. You must then use the **Browse** button to upload this file into the Enterprise Gateway configuration.

If you have already generated a suitable `SmHost.conf` file, and have copied it to the machine on which you are running the Policy Studio, you can browse to the location of this file using the **Browse** button at the bottom right of the **Connection Details** text area. After selecting the configuration file, the connection details are displayed in this text area.

If you do not have a suitable `SmHost.conf` file, you can generate one by running the `smregghost` command on the machine running the Enterprise Gateway. Complete the following steps:

1. You need to run the `smregghost` command on the machine on which you have installed the Enterprise Gateway. The `smregghost` tool is found in the following location, depending on your target platform:

Windows: `/win32/lib`

Linux: `/Linux.i386/bin`

Solaris: `/SunOS.sun4u-32/bin`

Open a command prompt at this directory, and run the `smregghost` command. You must pass the appropriate command-line arguments, depending on the `hostname` and `hostconfigobject` configured to represent the Enter-

- prise Gateway in the CA **Policy Server**. Similarly, you must specify the hostname/IP and port of the CA Policy Server.
2. The `smregghost` tool writes its output to a `SmHosts.conf` file in the same directory. You must manually copy this file from the machine running the Enterprise Gateway to the machine running the Policy Studio.
 3. Browse to the location of this file using the **Browse** button on the connection details dialog.

SOA Security Manager Connection Details Only

This section describes details that are specific to CA SOA Security Manager connections only. In addition to the fields already described in the previous section, you must also configure the following field on the **CA SOA Security Manager Connection Details** dialog.

XMLSDKAcceptSMSessionCookie:

This setting controls whether the CA SOA Security Manager authentication filter accepts a single sign-on token for authentication purposes. The single sign-on token must reside in the HTTP header field named `SMSESSION` to authenticate using this mechanism. This token is created and updated when the CA SOA Security Manager authorization filter runs successfully.

When this checkbox is selected, the authentication filter allows authentication using a single sign-on token. Note that if no single sign-on token is present in the message, the authentication filter authenticates fully by gathering credentials from the request in whatever manner has been configured in the CA SOA Security Manager. When this checkbox is unselected, the authentication filter authenticates fully (it never allows authentication using a single sign-on token).

Static CRL Certificate Validation

Overview

A Certificate Authority (CA) may wish to publish a Certificate Revocation List (CRL) to a file. In such cases, the Enterprise Gateway can load the revoked certificates from the file-based CRL and validate user certificates against it.

Because the CRL is typically signed by the CA that owns it, the certificate of the CA that issued the CRL *must* be imported into the **Certificate Store** before this filter can work correctly. In addition, the **Static CRL Certificate Validation** filter requires the `certificates` message attribute to be set by a predecessor.

Important Note:

Typically, a CA publishes a new CRL, containing the most up-to-date list of revoked certificates at regular intervals. However, the **Static CRL Certificate Validation** filter does not automatically update the CRL when it is loaded from a local file. If you need to automatically retrieve updated CRLs from a particular URL, you should use the **Dynamic CRL Certificate Validation** filter.

Configuration

Enter a name for the filter in the **Name** field, and click the **Load CRL** button to browse to the location of the CRL file. When the CRL has been loaded from the selected location, read-only information regarding revoked certificates and update dates is displayed in the other fields on the screen.

Certificate CRL - Dynamic

Overview

This filter is responsible for validating certificates against a Certificate Revocation List (CRL) that has been published by a Certificate Authority (CA). The CRL is retrieved from the specified URL and is cached by the server for certificate validation. The filter automatically fetches a potentially updated CRL from this URL when the criteria specified in the **Automatic CRL Update Preferences** section are met.

Configuration

Configure the following fields on the **Certificate CRL - Dynamic** screen:

Name:

Enter an appropriate name for the filter.

CRL Import URL:

Enter the full URL of the CRL to use to validate the certificate. Alternatively, you can browse to the location of the CRL by clicking the button.

Automatic CRL Update Preferences:

Typically, a CA publishes an updated CRL at regular intervals. You can configure the filter to dynamically pull down the latest CRL published by the CA at specified intervals. Select the appropriate update option from the following:

- **Do not update:**
The filter never attempts to automatically retrieve the latest CRL.
- **Update on "next update" date:**
The CRL published by the CA contains a *Next Update* date, which indicates the next date on which the CA publishes the CRL. You can choose to dynamically retrieve the updated CRL on the Next Update date by selecting this option. This effectively synchronizes the server with the CA updates.
- **Update every number of days:**
The filter retrieves the CRL every number of days specified.
- **Trigger update on cron expression:**
You can enter a cron expression to determine when to perform the automatic update.

CRL LDAP Validation

Overview

A CRL (Certificate Revocation List) is a signed list indicating a set of certificates that are no longer considered valid (revoked certificates) by the certificate issuer. The Enterprise Gateway can query a CRL to find out if a given certificate has been revoked. If the certificate is present in the CRL, it should not be trusted.

To validate a certificate using a CRL lookup, the certificate's issuing CA certificate should be trusted by the Enterprise Gateway. This is because for a CRL lookup, the CA public key is needed to verify the signature on the CRL. The issuing CA public key is not always included in the certificates that it issues, so it is necessary to retrieve it from the Enterprise Gateway's certificate store instead.

Configuration

The **Name** and **URL** of all currently configured LDAP directories are displayed in the table on the **CRL Certificate Validation** screen. The Enterprise Gateway checks the CRL of all selected LDAP directories to validate the client certificate. The filter fails as soon as the Enterprise Gateway determines that one of the CRLs has revoked the certificate.

To configure LDAP connection information, complete the following fields:

Name:

Enter an appropriate name for the filter.

LDAP Configuration:

To configure the Enterprise Gateway to check the CRL of a configured LDAP directory, select the checkbox next to the directory entry in the table on the main **Certificate Validation - CRL** screen.

You can add, edit, or delete LDAP Connections on the **External Connections** tab in Policy Studio. For example, right-click **LDAP Connections**, and select **Add a LDAP Connection**. For more information on configuring LDAP connections, see the [LDAP Configuration](#) topic.

CRL Responder

Overview

This filter enables the Enterprise Gateway to behave as Certificate Revocation List (CRL) responder, which returns CRLs to clients. This filter imports the CRL from a specified URL. You can also configure it to periodically retrieve the CRL from this URL to ensure that it always has the latest version.

Configuration

Configure the following fields on the **CRL Responder** screen:

Name:

Enter an appropriate name for the filter.

CRL Import URL:

Enter the full URL of the CRL that you want to return to clients. Alternatively, browse to the location of the CRL file by clicking the browse button on the right.

Automatic CRL Update Preferences:

Because keeping up-to-date with the latest list of revoked certificates is crucial in any *trust network*, it is important that you configure the filter to retrieve the latest version of the CRL on a regular basis. The following automatic update options are available:

- **Do not update:**
The CRL is not automatically updated.
- **Update on "next update" date:**
The CRL published by the CA contains a *Next Update* date, which indicates the next date on which the CA publishes the CRL. You can choose to dynamically retrieve the updated CRL on the Next Update date by selecting this option. This effectively synchronizes the server with the CA updates.
- **Update every number of days:**
The CRL is updated after the specified number of days has elapsed (for example, every 3 days).
- **Trigger update on cron expression:**
You can enter a cron expression to determine when to perform the automatic update.

Create Thumbprint from Certificate

Overview

This filter can be used to create a human-readable thumbprint (or fingerprint) from the X.509 certificate that is stored in the `certificate` message attribute. The generated thumbprint is stored in the `certificate.thumbprint` attribute.

Configuration

Configure the following fields on this filter:

Name:

Enter a name for this filter.

Digest Algorithm:

Select the digest algorithm to create the thumbprint of the certificate from the drop-down list.

Certificate Validity

Overview

The validity period of an X.509 certificate is encoded in the certificate. The **Certificate Validity** performs a simple check on a certificate to ensure that it has not expired.

By default, the **Certificate Validity** filter searches for the X.509 certificate in the `certificate` message attribute, which must be set by a predecessor filter in the circuit (for example, by an **SSL Authentication** filter).

Configuration

Configure the following fields on the **Certificate Validity** screen:

Name:

Enter an appropriate name for the filter.

Certificate Attribute:

Enter or select the name of the message attribute that you expect to hold the certificate. The filter checks the validity of the certificate contained in this attribute. If no certificate is found, the filter returns an error.

Find Certificate

Overview

The **Find Certificate** filter locates a certificate and sets it in the message for use by other certificate-based filters. Certificates can be extracted from the **User Store**, message attributes, HTTP headers, or attachments.

Configuration

By default, the Enterprise Gateway stores the extracted certificate in the `certificate` message attribute. However, it can store the certificate in any message attribute, including any arbitrary attribute specified by the user (for example, a `user_certificate` attribute). The certificate can then be extracted from this attribute by a successor filter in the policy.

Name:

Enter a name for the filter in the **Name** field.

Attribute Name:

Enter or select the name of the message attribute to store the extracted certificate in.

When the target message attribute has been selected, the next step is to specify the location of the certificate from one of the following options:

User:

Select a **User** whose certificate is extracted from the **Certificate Store** and set to the message.

Certificate Store:

Click the **Select** button and select a certificate from the Trusted Certificate Store.

User or Wildcard:

This field represents an alternative way to specify what user's certificate is used. Either an explicitly named **User's** certificate is used, or you can specify a property to locate a **User** name or DName that can then be used to locate the certificate.

You can specify a property by enclosing the message attribute that contains the user name or DName in curly brackets, and prefixing this with the `$` sign. For example:

```
${authentication.subject.id}
```

This property means that the Enterprise Gateway uses the certificate belonging to the subject of the authentication event in subsequent certificate-related filters. The certificate is set to the `certificate` message attribute.

Using properties is a more generic way of locating certificates than specifying the **User** directly.

Message Attribute Name:

Enter the name of the message attribute that contains the certificate.

HTTP Header Name:

Enter the name of the HTTP header that contains the certificate.

Attachment Name:

Specify the name of the attachment (`Content-Id`) that contains the certificate. You can enter a wildcard in this field to

represent the value of a message attribute.

Extract Certificate Attributes

Overview

You can use the **Extract Certificate Attributes** filter to extract the X.509 attributes from a certificate stored in a specified Oracle message attribute.

Typically, this filter is used in conjunction with the **Find Certificate** filter, which is found in the **Certificates** category of message filters. In this case, the **Find Certificate** filter can locate a certificate from one of many possible sources (for example, the message itself, an HTTP header, or the Enterprise Gateway Certificate Store), and store it in a message attribute, which is usually the `certificate` attribute.

The **Extract Certificate Attributes** filter can then retrieve this certificate and extract the X.509 attributes from it. For example, you can then use a **Validate Message Attribute** filter to check the values of the attributes.

Generated Message Attributes

The **Extract Certificate Attributes** filter extracts the X.509 certificate attributes and populates a number of Oracle message attributes with their respective values. The following table lists the message attributes that are generated by this filter, and shows what each of these attributes contains after the filter has executed:

Generated Message Attribute	Contains
<code>attribute.lookup.list</code>	This user attribute list contains an attribute for each Distinguished Name (DName) attribute for the subject (<code>cn</code> , <code>o</code> , <code>l</code> , and so on). The user attributes are named <code>cn</code> , <code>o</code> , and so on.
<code>attribute.subject.id</code>	The DName of the subject of the cert.
<code>attribute.subject.format</code>	Set to <code>X509DName</code> .
<code>cert.basic.constraints</code>	If the subject is a Certificate Authority (CA), and the <code>BasicConstraints</code> extension exists, this field gives the maximum number of CA certificates that may follow this certificate in a certification path. A value of zero indicates that only an end-entity certificate may follow in the path. This contains the value of <code>pathLenConstraint</code> if the <code>BasicConstraints</code> extension is present in the certificate and the subject of the certificate is a CA, otherwise its value is -1. If the subject of the certificate is a CA and <code>pathLenConstraint</code> does not appear, there is no limit to the allowed length of the certification path.
<code>cert.extended.key.usage</code>	A String representing the OBJECT IDENTIFIERS of the <code>ExtKeyUsageSyntax</code> field of the extended key usage extension (OID = 2.5.29.37). It indicates a purpose for which the certified public key may be used, in addition to, or instead of, the basic purposes indicated in the key usage extension field.
<code>cert.hash.md5</code>	An MD5 hash of the certificate.
<code>cert.hash.shal</code>	A SHA1 hash of the certificate.
<code>cert.issuer.alternative.name</code>	An alternative name for the certificate issuer from the <code>IssuerAltName</code> extension (OID = 2.5.29.18).
<code>cert.issuer.id</code>	The DName of the issuer of the certificate.
<code>cert.issuer.id.c</code>	The <code>c</code> attribute of the issuer of the certificate, if it exists.

Generated Message Attribute	Contains
<code>cert.issuer.id.cn</code>	The <code>cn</code> attribute of the issuer of the certificate, if it exists.
<code>cert.issuer.id.emailaddress</code>	The <code>email</code> or <code>emailaddress</code> attribute of the issuer of the certificate, if it exists.
<code>cert.issuer.id.l</code>	The <code>l</code> attribute of the issuer of the certificate, if it exists.
<code>cert.issuer.id.o</code>	The <code>o</code> attribute of the issuer of the certificate, if it exists.
<code>cert.issuer.id.ou</code>	The <code>ou</code> attribute of the issuer of the certificate, if it exists.
<code>cert.issuer.id.st</code>	The <code>st</code> attribute of the issuer of the certificate, if it exists.
<code>cert.key.usage.cRLSign</code>	Set to <code>true</code> or <code>false</code> if the key can be used for <code>crlSign</code> .
<code>cert.key.usage.dataEncipherment</code>	Set to <code>true</code> or <code>false</code> if the key can be used for <code>dataEncipherment</code> .
<code>cert.key.usage.decipherOnly</code>	Set to <code>true</code> or <code>false</code> if the key can be used for <code>decipherOnly</code> .
<code>cert.key.usage.digitalSignature</code>	Set to <code>true</code> or <code>false</code> if the key can be used for <code>digitalSignature</code> .
<code>cert.key.usage.encipherOnly</code>	Set to <code>true</code> or <code>false</code> if the key can be used for <code>encipherOnly</code> .
<code>cert.key.usage.keyAgreement</code>	Set to <code>true</code> or <code>false</code> if the key can be used for <code>keyAgreement</code> .
<code>cert.key.usage.keyCertSign</code>	Set to <code>true</code> or <code>false</code> if the key can be used for <code>keyCertSign</code> .
<code>cert.key.usage.keyEncipherment</code>	Set to <code>true</code> or <code>false</code> if the key can be used for <code>keyEncipherment</code> .
<code>cert.key.usage.nonRepudiation</code>	Set to <code>true</code> or <code>false</code> if the key can be used for <code>nonRepudiation</code> .
<code>cert.not.after</code>	Not after validity period date.
<code>cert.not.before</code>	Not before validity period date.
<code>cert.serial.number</code>	Certificate serial number.
<code>cert.signature.algorithm</code>	The signature algorithm for certificate signature.
<code>cert.subject.alternative.name</code>	An alternative name for the subject from the <code>SubjectAltName</code> extension (OID = 2.5.29.17).
<code>cert.subject.id</code>	The <code>DName</code> of the subject of the certificate.
<code>cert.subject.id.c</code>	The <code>c</code> attribute of the subject of the certificate, if it exists.
<code>cert.subject.id.cn</code>	The <code>cn</code> attribute of the subject of the certificate, if it exists.
<code>cert.subject.id.emailaddress</code>	The <code>email</code> or <code>emailaddress</code> attribute of the subject of the certificate, if it exists.
<code>cert.subject.id.l</code>	The <code>l</code> attribute of the subject of the certificate, if it exists.
<code>cert.subject.id.o</code>	The <code>o</code> attribute of the subject of the certificate, if it exists.
<code>cert.subject.id.ou</code>	The <code>ou</code> attribute of the subject of the certificate, if it exists.
<code>cert.subject.id.st</code>	The <code>st</code> attribute of the subject of the certificate, if it exists.
<code>cert.version</code>	The certificate version.

Configuration

Name:

Enter a name for the filter.

Certificate Attribute:

The **Extract Certificate Attributes** filter extracts the attributes from the certificate contained in the message attribute selected or entered here. The selected attribute must contain a single certificate only.

Include Distribution Points:

If the certificate contains CRL Distribution Point X.509 extension attributes (which point to the location of the certificate issuer's CRL), you can also extract these and store them in message attributes by selecting this checkbox. The extracted distribution points are stored in message attributes that are prefixed by:

`distributionpoint.`

Certificate Chain Check

Overview

It is a trivial task for a user to generate a structurally sound X.509 certificate, and use it to negotiate mutually authenticated connections to publicly available services. However, this scenario is a security nightmare for IT administrators. You can not allow every user to generate their own certificate and use it on the Internet. For this reason, the Enterprise Gateway can establish the authenticity of the client certificate by ensuring that the certificate originated from a trusted source. To do this, a server can perform a *certificate chain check* on the client certificate.

The main purpose of certificate chain validation is to ensure that a certificate has been issued by a trusted source. Typically, in a Public Key Infrastructure (PKI), a Certificate Authority (CA) is responsible for issuing and distributing certificates. This infrastructure is based on the premise of *transitive trust*—if everybody trusts the CA, everybody transitively trusts the certificates issued by that CA. If entities only trust certificates that have been issued by the CA, they can reject certificates that have been self-generated by clients.

When a CA issues a certificate, it digitally signs the certificate and inserts a copy of its own certificate into it. This is called a certificate chain. Whenever an application (such as the Enterprise Gateway) receives a client certificate, it can extract the issuing CA certificate from it, and run a certificate chain check to determine whether it should trust the CA. If it trusts the CA, it also trusts the client certificate.

The Enterprise Gateway maintains a repository of trusted CA certificates, which is known as the **Certificate Store**. To *trust* a specific CA, that CA certificate must be imported into the **Certificate Store**.

Configuration

You can configure the following settings on the **Certificate Chain Check** screen:

Name:

Enter an appropriate name for this filter.

Certificates Message Attribute:

You can specify a message attribute that contains the certificate or certificates to check. The message attribute type can be an `X509Certificate` object, or an `ArrayList` of `X509Certificate` objects.

Distinguished Name:

This table lists the Distinguished Names of the certificates currently in the **Certificate Store**. Select the checkbox beside a CA to enable this filter to consider it as *trusted* when performing the certificate chain check. You can select multiple CAs in the table.

OCSP Certificate Validation

Overview

Online Certificate Status Protocol (OCSP) is an automated certificate checking network protocol. The Enterprise Gateway can query an OCSP responder for the status of a certificate. The responder returns whether the certificate is still trusted by the CA that issued it.

To validate a certificate using an OCSP lookup, the issuing CA certificate should be trusted by the Enterprise Gateway. This is because for an OCSP request, the protocol stipulates that the CA public key must be submitted as part of the request. The issuing CA public key is not always included in the certificates that it issues, so it is necessary to retrieve it from the Enterprise Gateway's certificate store instead. For more information on how to trust CA certificates, see the [Certificate](#) tutorial.

Configuration

The table on the **Certificate Validation - OCSP** screen lists the currently available global OCSP Connections. You can add OCSP Connections on the **External Connections** tab in Policy Studio.

Configure the following fields on the **Certificate Validation - OCSP** dialog:

Name:

Enter an appropriate name for this OCSP filter.

OCSP Connection:

Select one or more global OCSP Connections from the table. To add a global OCSP Connection, on the **External Connections** tab, right-click the **OCSP Connections** node, and select **Add an OCSP Connection**. For more information on configuring these connections, see the [OCSP Connection](#) topic.

Validate Certificates in Gateway's Store

Overview

This filter checks the Enterprise Gateway's certificate store for certificates that are due to expire before a specified number of days. This enables you to monitor the certificates that the Enterprise Gateway is running with.

For example, you can configure a policy that includes a **Validate Certificates in Gateway's Store** filter and an **Alert** filter, which sends an email alert when it finds certificates that are due to expire. You can also configure this policy to run at regular intervals using the policy execution scheduler provided with the Enterprise Gateway.

Configuration

Configure the following fields on the **Validate Gateway's Certificate Store** screen:

Name:

Enter an appropriate name for the filter.

Days before expires:

Enter the number of days before the certificates are due to expire.

Check Gateway's Certificate Store:

Select whether to check the certificates in the Enterprise Gateway's Certificate store. This is selected by default.

Check Gateway's Java Keystore:

Select whether to check the certificates in the Enterprise Gateway's Java Keystore. This is not selected by default. When selected, you must enter the **Password** for this keystore. The default password is `changeit`.

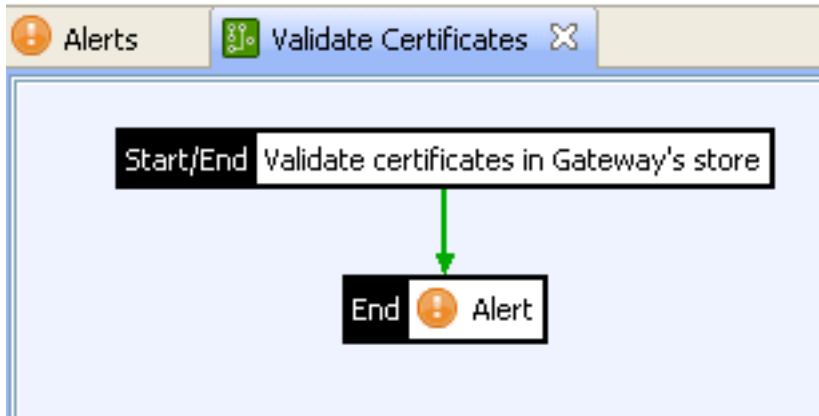
Check Java Keystore:

Select whether to check the certificates in the specified Java Keystore. This is not selected by default. When selected, you must configure the following fields:

Keystore Location	Specify the path to this keystore (for example, <code>/home/oracle/osr-client.jks</code>).
Password	Enter the password for this keystore.

Deployment Example

The following example shows a **Validate Certificates** policy that includes a **Validate Certificates in Gateway's Store** filter and an **Alert** filter. This policy sends an email alert when it finds certificates that are due to expire:



Configuring an Email Alert

When this filter is successful, and finds certificates that are due to expire, it generates an `expired.certs.summary` attribute, which contains a summary of certificates due to expire. You can then use this attribute in the **Alert** filter to send an email alert to the Enterprise Gateway administrators, as shown in the following example:

Alert filter

Configure when and where to alert



Name:

Message

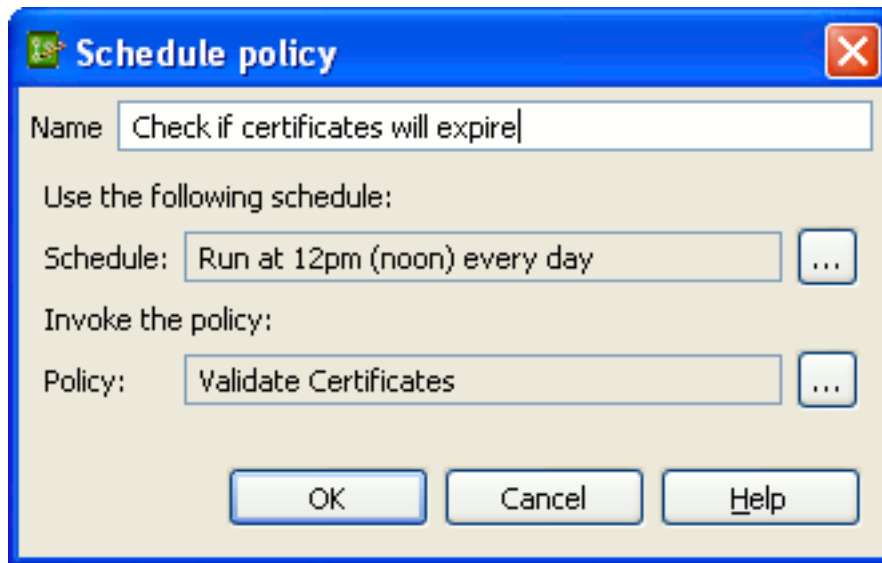
Alert Type: ☒ Error ☐ Warn ☐ Info

Alert message:

You must also select a pre-configured email alert destination on the **Destination** tab (for example, **Email Gateway Administrators**). For more details on configuring email alert destinations, see the [Alerts](#) topic.

Configuring a Policy Execution Schedule

You can configure this policy to run at regular intervals (for example, once every day) using the policy scheduler provided with the Enterprise Gateway. On the **Services** tab, right-click the Enterprise Gateway process node, and select **Add policy execution scheduler**. The following example runs the policy at 12 noon every day:



For more details, see the [Policy Execution Scheduling](#) topic.

Example Email Alert

An email alert is sent if any certificates that are due to expire are detected. The contents of the email are obtained from the `expired.certs.summary` message attribute. For example:

Oracle Enterprise Gateway running on Roadrunner contains certificates that will expire in 730 da

2 expired certificates in Gateway certificate store:

1. Cert details:

```
Cert issued to: CN=CA
Cert issued by: CN=CA
SHA1 fingerprint: 72:04:35:7C:A1:B1:C2:F5:E2:86:75:C4:83:12:9C:70:A8:D6:21:8E
MD5 fingerprint: 82:23:6F:59:F2:8F:C3:95:56:87:70:B5:51:3F:53:05
Subject Key Identifier (SKI): dfABenFoM0r7iJ3E1ZqU7HmKiyY=
Expires on: 2012-04-20
```

2. Cert details:

```
Cert issued to: CN=John Doe
Cert issued by: CN=CA
SHA1 fingerprint: 83:32:EB:3F:9C:15:87:FB:81:E1:D5:AC:CC:35:C3:F8:21:BB:DF:CD
MD5 fingerprint: 48:02:F6:3F:B9:64:EB:DA:DF:CF:F9:82:AC:CC:13:AB
Subject Key Identifier (SKI): HabJNMjAsBAWp4AcCq8yZkTEJKQ=
Expires on: 2012-04-20
```

XKMS Certificate Validation

Overview

XML Key Management Specification (XKMS) is an XML-based protocol that enables you to establish the trustworthiness of a certificate over the Internet. The Enterprise Gateway can query an XKMS responder to determine whether a given certificate can be trusted.

Configuration

The table on the **Certificate Validation - XKMS** screen lists the currently available global XKMS Connections. You can add XKMS Connections on the **External Connections** tab in Policy Studio.

You can configure the following fields on the **Certificate Validation - XKMS** screen.

Name:

Enter an appropriate name for this XKMS filter.

XKMS Connection:

Select one or more global XKMS Connections from the table. To add a global XKMS Connection, on the **External Connections** tab, right-click the **XKMS Connections** node, and select **Add an XKMS Connection**. For more information on configuring these connections, see the [XKMS Connection](#) topic.

Cache Attribute

Overview

The **Cache Attribute** filter allows you to configure what part of the message you want to cache. Typically, response messages are cached and so this filter is usually configured *after* the routing filters in a circuit. In this case the **content.body** attribute stores the response message body from the Web Service and so this message attribute should be selected in the **Attribute Name to Store** field.

For more information on how to configure this filter in the context of a "caching circuit", please refer to the [Global Caches](#) tutorial.

Configuration

Name:

Enter a name for this filter here.

Select Cache to Use:

The list of currently configured caches will be displayed in the tree. Select the name of the cache to store the attribute value in. Global caches (both local and distributed) can be added from the **Caches** top level node in the tree view of the Policy Studio.

Attribute Key:

The value of the message attribute entered here acts as the key into the cache. In the context of a "caching circuit", it *must* be the same as the attribute specified in the **Attribute containing key** field on the **Is Cached?** filter.

Attribute Name to Store:

The value of the Oracle message attribute entered here will be cached in the cache specified in the **Cache to use** field above.

Create Key

Overview

The **Create Key** filter is used to identify the part of the message that determines whether a message is unique. For example, you can use the request message body to determine uniqueness so that if two successive identical message bodies are received by the Enterprise Gateway, the response for the second request is taken from the cache.

You can also use other parts of the request to determine uniqueness (for example, HTTP headers, client IP address, client SSL certificate, and so on). This means that you can use the **Create Key** filter to create keys for a range of different caching scenarios (for example, caching a user's role, or caching a session for a user).

For more information on how to configure this filter in the context of a caching circuit, see the [Global Caches](#) tutorial. This shows the order in which caching filters such as the **Create Key** filter are placed in an example caching circuit.

Configuration

Name:

Enter a name for this filter here.

Attribute Name:

Enter the name of the Oracle message attribute to use to determine whether an incoming request is unique or not. For example, if **http.request.clientcert** (the client SSL certificate) is selected, the Enterprise Gateway takes a cached response for successive requests in which the client SSL certificate is the same.

Is Cached?

Overview

The **Is Cached?** filter looks up a named cache to see if a specified message attribute has already been cached. A message attribute - usually **message.key** - is used as the key to search for in the cache. If the lookup succeeds, the retrieved value overrides a specified message attribute, which is usually the **content.body** attribute.

For example, if a response message for a particular request has already been cached, the response message overrides the request message body so that it can be simply returned to the client using the **Reflect** filter.

For more information on how to configure this filter in the context of a "caching circuit", please refer to the [Global Caches](#) tutorial.

Configuration

Name:

Enter a name for this filter here.

Cache to Use:

The list of currently configured caches will be displayed in the tree. Select the name of the cache to lookup to find the attribute specified in the **Attribute containing key** field below. Global caches (both local and distributed) can be added from the **Caches** top level node in the tree view of the Policy Studio.

Attribute containing key:

The message attribute entered here will be used as the key to lookup in the cache. In the context of a "caching circuit", the attribute entered here *must* be the same as the attribute specified in the **Attribute key** field on the **Cache Attribute** filter.

Overwrite Attribute Name if Found:

Usually the **content.body** is selected here so that value retrieved from the cache (which is usually a response message) overrides the request **content.body** with the cached response, which can then be returned to the client using the **Reflect** filter.

Removed Cached Attribute

Overview

The **Remove Cached Attribute** filter allows you to delete a message attribute value that has been stored in a cache. Each cache is essentially a map of name-value pairs, where each value is keyed on a particular message attribute. For example, it is possible to store a cache of request messages according to their message ID. In this case the message's `id` attribute would be the key into the cache, which would store the value of the request message's `content.body` message attribute.

In this example, the **Remove Cached Attribute** filter can be used to remove a particular entry from the cache based on the run-time value of a particular message attribute. By specifying the `id` message attribute to remove, the Enterprise Gateway will look up the cache based on the value of the `id` message attribute. When it finds a matching message ID in the cache, it will remove the corresponding entry from the cache.

The example described above may be useful in cases where a request message may need to be cached and stored until the request has been fully processed and a response returned to the client. For example, if the request must be routed on to a back-end Web Service, but that Web Service is temporarily unavailable, it may be possible to configure the circuit to re-send the cached request instead of forcing the client to retry.

For more information on how to configure a "caching circuit", please refer to the [Global Caches](#) tutorial.

Configuration

Name:

Enter a name for this filter here.

Select Cache to Use:

The list of currently configured caches will be displayed in the tree. Select the name of the cache that contains the cached values that have been keyed according to the message attribute specified below. Global caches (both local and distributed) can be added from the **Caches** top level node in the tree view of the Policy Studio.

Attribute Key:

Enter the message attribute that is used as the key into the cache in this field. At run-time, the Enterprise Gateway will populate the value of this message attribute, which will then be used to lookup the cache selected in the table above. If a match is found in the cache, the corresponding entry will be deleted from the cache.

Content Validation

Overview

This tutorial describes how the Enterprise Gateway can examine the contents of an XML message to ensure that it meets certain criteria. It uses boolean XPath expressions to evaluate whether or not a specific element or attribute contains has a certain value.

For example, it is possible to configure XPath expressions to make sure the value of an element matches a certain string, to check the value of an attribute is greater (or less) than a specific number, or that an element occurs a fixed amount of times within an XML body.

There are two ways to configure XPath expressions on this screen. Please click on the appropriate link below:

- [Manual XPath Configuration](#)
- [XPath Wizard](#)

Manual XPath Configuration

To manually configure a **Content Validation** rule using XPath:

1. Enter a meaningful name for this XPath content filter.
2. Click the **Add** button to add a new XPath expression. Alternatively, you can select a previously configured XPath expression from the dropdown.
3. In order to resolve any prefixes within the XPath expression, the namespace mappings (i.e. **Prefix, URI**) should be entered in the table.

As an example of how this screen should be configured, consider the following SOAP message:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" id="sig1">
      .....
      .....
      .....
      .....
    </dsig:Signature>
  </soap:Header>
  <soap:Body>
    <prod:product xmlns:prod="http://www.company.com">
      <prod:name>SOA Product</prod:name>
      <prod:company>Company</prod:company>
      <prod:description>WebServices Security</prod:description>
    </prod:product>
  </soap:Body>
</soap:Envelope>
```

The following XPath expression evaluates to true if the *<company>* element contains the value "Company":
XPath Expression: //prod:company[text()='Company']

In this case it is necessary to define a mapping for the *prod* namespace as follows:

Prefix	URI
prod	http://www.company.com

Let's look at another example. This time the element that is to be examined by the XPath expression belongs to a default namespace. Consider the following SOAP message:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" id="sig1">
      .....
      .....
      .....
      .....
    </dsig:Signature>
  </soap:Header>
  <soap:Body>
    <product xmlns="http://www.company.com">
      <name>SOA Product</name>
      <company>Company</company>
      <description>Web Services Security</description>
    </product>
  </soap:Body>
</soap:Envelope>
```

The following XPath expression evaluates to true if the *<company>* element contains the value *Company*.

XPath Expression: `//ns:company[text()='Company']`

Because the *<company>* element belongs to the default (xmlns) namespace (*http://www.company.com*), you must make up an arbitrary prefix (*ns*) for use in the XPath expression, and assign it to *http://www.company.com*. This is necessary to distinguish between potentially several default namespaces which may exist throughout the XML message. The following mapping illustrates this:

Prefix	URI
ns	http://www.company.com

XPath Wizard

The **XPath Wizard** assists administrators in creating correct and accurate XPath expressions. The wizard allows administrators to load an XML message and then run an XPath expression on it to determine what nodes are returned. To launch the **XPath Wizard**, click on the **XPath Wizard Button** on the **XPath Expression** dialog.

To use the XPath Wizard, simply enter (or browse to) the location of an XML file in the **File** field. The contents of the XML file will appear in the main window of the wizard. Enter an XPath expression in the **XPath** field and click the **Evaluate** button to run the XPath against the contents of the file. If the XPath expression returns any elements (or returns true), those elements will be highlighted in the main window.

If you are not sure how to write the XPath expression yourself, you can simply select an element in the main window. An XPath expression to isolate this element is automatically generated and displayed in the **Selected** field. If you wish to use this expression, select the **Use this path** button, and click **OK**.

Message Size

Overview

It is sometimes useful to filter incoming messages based, not only on the content of the message, but on external characteristics of the message. To this end, the Enterprise Gateway can be configured to reject messages that are greater or less than a specified size.

Configuration

To configure the Enterprise Gateway to block messages of a certain size, complete the following fields:

- Enter the size (in bytes) of the smallest message that should be processed in the **At least** field. Messages smaller than this size will be rejected.
- Enter the size (in bytes) of the largest message that should be processed in the **At most** field. Messages larger than the size entered here will be rejected.
- The **Use in Size Calculation** options are used to specify the portion of the message that is to be used when calculating the size of the message.
 - If the **Root body only** option is selected, the Enterprise Gateway will calculate the size of the message body excluding all other MIME parts, i.e. attachments.
 - If the **Attachments only** option is selected, the Enterprise Gateway will only calculate the size of all attachments to the message. It will exclude the size of the root body payload from its calculation.
 - Finally, if the **Root body and attachments** option is selected, the Enterprise Gateway will include the root body together with all other MIME parts when it calculates the size of the message.

It is important to note the following:

- The message size measured by the Enterprise Gateway does **not** include HTTP headers.

Throttling

Overview

The **Throttling** filter can protect a Web Service or Service Oriented Architecture (SOA) from message flooding. You can configure the filter to only allow a certain number of messages from a specified client in a given time frame through to the protected system.

If the number of messages exceeds the specified limit, the filter fails for the excess messages. It is important to note that the filter still succeeds for incoming messages that meet the specified constraints. For example, if the filter is configured to allow 20 messages through per second, it fails for the 21st message, but passes for the first 20 incoming messages.

The Enterprise Gateway's behavior in the case of a breach in the configured constraints is determined by the filter that is next in the failure path for the **Throttling** filter in the policy. Typically, an **Alert**, **Trace**, or **Log** filter is configured as the successor filter in the failure path in the policy.

An example use-case of this filter would be to protect a Web Service that can only handle a maximum of 20 messages per client per second. If the filter detects a higher number of incoming requests, it blocks the messages.

General Settings

Name:

Enter an appropriate name for the filter.

Number of Messages:

If the Enterprise Gateway receives more than this number of messages during the time interval specified in the field below, the filter fails. Otherwise, the filter passes.

Time Period:

If the Enterprise Gateway receives more than the number of messages specified in the above field in the time interval specified here, the filter fails. Otherwise, the filter passes. The time period depends on the value selected below (seconds, minutes, hours, days, or weeks). For example, if you enter 10 as the **Time Period** and select **Minutes** from the **Time Period Unit** drop-down list below, the time period lasts 10 minutes.

Time Period Unit:

The unit that the time period is measured in must be selected from the following options: *Second*, *Minute*, *Hour*, *Day*, or *Week*. The value selected together with that entered above determines the time period.

Alternatively, you can specify the time period unit using a property to represent the value of a message attribute, which is looked up and expanded to a value at runtime. Use the following syntax to specify the property: `${name_of_message_attribute}`. Allowed values are *Second*, *Minute*, *Hour*, *Day*, or *Week*.

Note:

When one of *Second*, *Minute*, or *Hour* is specified, you do not need to configure the **Time Period Commences on Hour/Day** fields. The time period commences when a message is received and lasts for the time period (for example, 10 minutes). When this time period is up, the message count is reset, and the counter starts again when another message is received. The sections that follow explain how the time period is measured when the **Time Period Unit** is set to *Day* or *Week*.

Time Period Commences on Hour:

This field must be configured if you select either *Day* or *Week* as the **Time Period Unit** above. For example, if you select *Day* above and enter 00:00 in this field, this means that only the specified number of messages can be received in a one day period starting from midnight tonight until midnight the next day.

Time Period Commences on Day:

This field must only be configured if you select *Week* as the **Time Period Unit** above. For example, if you select *Week* and 00:00 in the fields above, and enter *Sunday*, this means that the time period commences next Sunday at midnight

and will last for one week exactly. The time period is reset on midnight of the next Sunday.

Cache Settings

Track per Key:

Select this box if you wish to configure the Enterprise Gateway to keep track of request messages based on a specific key value. In other words, if more messages that match this key are received than are allowed, the filter fails.

Key Value:

You can use the **Track per Key** option to perform message filtering based on a particular key. This key can be used to lookup entries in the cache selected below. The key entered can be a combination of a fixed string value and/or Enterprise Gateway message attribute. For example, the following key could be used to keep track of the number of times a particular URI is requested:

```
MSG_COUNT-${http.request.uri}
```

Select Cache to Use:

In cases where multiple Enterprise Gateways are deployed for load balancing purposes and you want to maintain a single count of all messages processed by all the Enterprise Gateway instances, you can configure a distributed cache to cache request messages.

For example, assume the intention is to prevent a burst of more than 50 messages per second from reaching the back-end Web Service. Also assume that a load balancer is deployed in front of two instances of the Enterprise Gateway and round-robins requests between these two instances. By caching request messages in a global distributed cache, which is inherently replicated across all Enterprise Gateway instances, the **Throttling** filter can compute the total number of messages in the distributed cache and hence, the total number of messages processed by all Enterprise Gateway instances.

The **Throttling** filter uses the pre-configured **Local maximum messages** cache by default. You can configure more caches using the [Global Cache](#) interface.

Using Multiple Maximum Message Filters

If you want two or more **Throttling** filters to maintain separate message counts, you must use a different key into the cache for each filter, or use different caches for each filter:

- Use a Different Key per Filter:**
 With this approach you can use a unique **Track per key** value in each **Throttling** filter. The easiest way to do this is to prepend the default `${http.request.clientaddr}` value with the filter name, for example:

```
${filterName} ${http.request.clientaddr}
```


 This ensures that each filter maintains its own separate message count in the selected cache.
- Use a Unique Cache per Filter:**
 Alternatively, you can use a unique cache to store the message count of each **Throttling** filter. With this solution, you must configure a separate cache for each **Throttling** filter that you have configured throughout *all* policies running on the Enterprise Gateway.

Content Type Filtering

Overview

The *SOAP Messages with Attachments* specification introduced a standard for transmitting arbitrary files along with SOAP messages as part of a multipart MIME message. In this way, both XML and non-XML data, including binary data, can be encapsulated in a SOAP message. The more recent Direct Internet Message Encapsulation (DIME) specification describes another way of packaging attachments with SOAP messages.

The Enterprise Gateway can accept or block multipart messages with certain MIME or DIME content types. For example, you can configure a filter that blocks multipart messages that contain parts that are of type `image/jpeg`.

Allow or Deny Types

The **Content Type Filtering** screen lists the content types that are allowed or denied by this filter.

Allow Content Types:

Use this option if you wish to *accept* most content types, but only want to reject a few specific types. To allow or deny incoming messages based on their content types, complete the following steps:

1. Select the **Allow content types** radio button to allow multipart messages to be routed onwards. If you wish to allow all content types, you do not need to select any of the MIME types in the list.
2. To deny multipart messages with certain MIME or DIME types as parts, select the checkbox next to those types. Multipart messages containing parts of the MIME or DIME types selected here will be rejected.

Deny Content Types:

If you wish to *block* multipart messages containing most content types, but want to allow a small number of content types, select this option. To reject multipart messages based on the content types of their parts, complete the following steps:

1. Select the **Deny content types** radio button to reject multipart messages. If you wish to block all multipart messages, you do not need to select any of the MIME or DIME types in the list.
2. To allow messages with parts of a certain MIME or DIME type, select the checkbox next to those types. Multipart messages with parts of the MIME or DIME types selected here will be allowed. All other MIME or DIME types will be denied.

MIME and DIME types can be added by clicking the **MIME/DIME Registered Types** button. The next section describes how to add, edit, and remove MIME/DIME types.

Configuring MIME/DIME Types

The **MIME/DIME Settings** dialog enables you to configure new and existing MIME types. When a type has been added, you can configure the Enterprise Gateway to accept or block multipart messages with parts of this type.

Click the **Add** button to add a new MIME/DIME type, or highlight a type in the table, and select the **Edit** button to edit an existing type. To delete an existing type, select that type in the list, and click the **Remove** button. You can edit or add types using the **Configure MIME/DIME Type** dialog.

Enter a name for the new type in the **MIME or DIME Type** field, and the corresponding file extension in the **Extension** field.

HTTP Header Validation

Overview

The Enterprise Gateway can check HTTP header values for threatening content. This ensures that only properly configured name-value pairs appear in the HTTP request headers. *Regular expressions* are used to test HTTP header values. This enables you to make decisions on what to do with the message (for example, if the HTTP header value is *x*, route to service *x*).

You can configure the following sections on the **Validate HTTP Headers** screen:

- **Enter Regular Expression:**
HTTP header values can be checked using regular expressions. You can select regular expressions from the global **White list** or enter them manually. For example, if you know that an HTTP header must have a value of *ABCD*, a regular expression of *^ABCD\$* is an exact match test.
- **Enter Threatening Content Regular Expression:**
You can select threatening content regular expressions from the global **Black list** to run against all HTTP headers in the message. These regular expressions identify common attack signatures (for example, SQL injection attacks).

You can configure the global **White list** and **Black list** libraries of regular expressions on the **Policies** tab in the Policy Studio.

Configuring HTTP Header Regular Expressions

The **Enter Regular Expression** table displays the list of configured HTTP header names together with the **White list** of regular expressions that restrict their values. For this filter to run successfully, *all* required headers must be present in the request, and *all* must have values matching the configured regular expressions.

The **Name** column shows the name of the HTTP header. The **Regular Expression** column shows the name of the regular expression that the Enterprise Gateway uses to restrict the value of the named HTTP header. A number of common regular expressions are available from the global **White list** library.

Configuring a Regular Expression

You can configure regular expressions by selecting the **Add**, **Edit**, and **Delete** buttons. The **Configure Regular Expression** dialog enables you to add or edit regular expressions to restrict the values of HTTP headers. To configure a regular expression, perform the following steps:

1. Enter the name of the HTTP header in the **Name** field.
2. Select whether this header is **Optional** or **Required** using the appropriate radio button. If it is **Required**, the header *must* be present in the request. If the header is not present, the filter fails. If it is **Optional**, the header does not need to be present for the filter to pass.
3. You can enter the regular expression to restrict the value of the HTTP header manually or select it from the global **White list** library of regular expressions in the **Expression Name** drop-down list. A number of common regular expressions are provided (for example, alphanumeric values, dates, and email addresses).
You can use properties representing the values of message attributes to compare the value of an HTTP header with the value contained in a message attribute. Enter the *\$* character in the **Regular Expression** field to view a list of available attributes. At runtime, the property is expanded to the corresponding attribute value, and compared to the HTTP header value that you want to check.
4. You can add a regular expression to the library by selecting the **Add/Edit** button. Enter a **Name** for the expression followed by the **Regular Expression**.

Advanced Settings

The **Advanced** section enables you to extract a portion of the header value which is run against the regular expression.

The extracted substring can be Base64 decoded if necessary. This section is specifically aimed towards HTTP Basic authentication headers, which consist of the `Basic` prefix (with a trailing space), followed by the Base64-encoded username and password. The following is an example of the HTTP Basic authentication header:

```
Authorization: Basic dXNlcjplc2Vy
```

The Base64-encoded portion of the header value is what you are interested in running the regular expression against. You can extract this by specifying the string that occurs directly before the substring you want to extract, together with the string that occurs directly after the substring.

To extract the Base64-encoded section of the `Authorization` header above, enter `Basic` (with a trailing space) in the **Start substring** field, and leave the **End substring** field blank to extract the entire remainder of the header value.

Important Note:

You must select the start and end substrings to ensure that the exact substring is extracted. For example, in the HTTP Basic example above, you should enter `Basic` (with a trailing space) in the **Start substring** field, and *not* `Basic` (with no trailing space).

By specifying the correct substrings, you are left with the Base64-encoded header value (`dXNlcjplc2Vy`). However, you still need to Base64 decode it before you can run a regular expression on it. Make sure to select the **Base64 decode** checkbox. The Base64-decoded header value is `user:user`, which conforms to the standard format of the `Authorization` HTTP header. This is the value that you need to run the regular expression against.

The following example shows an example of an HTTP Digest authentication header:

```
Authorization: Digest username="user", realm="oracle.com", qop="auth",
algorithm="MD5", uri="/editor", nonce="Id-00000109924ff10b-0000000000000091",
nc="1", cnonce="ael22a8b549af2f0915de868abff55bacd7757ca",
response="29224d8f870a62ce4acc48033c9f6863"
```

You can extract single values from the header value. For example, to extract the `realm` field, enter `realm="` (including the `"` character), in the **Start substring** field and `"` in the **End substring** field. This leaves you with `oracle.com` to run the regular expression against. In this case, there is no need to Base64 decode the extracted substring.

Note:

If both **Start substring** and **End substring** fields are blank, the regular expression is run against the entire header value. Furthermore, if both fields are blank and the **Base64 decode** checkbox is selected, the entire header value is Base64 encoded before the regular expression is run against it.

While the above examples deal specifically with the HTTP authentication headers, the interface is generic enough to enable you to extract a substring from other header values.

Configuring Threatening Content Regular Expressions

The regular expressions entered in this section guard against the possibility of an HTTP header containing malicious content. The **Enter Threatening Content Regular Expression** table lists the **Black list** of regular expressions to run to ensure that the header values do not contain threatening content.

For example, to guard against an SQL `DELETE` attack, you can write a regular expression to identify SQL syntax and add

it to this list. The **Threatening Content Regular Expressions** are listed in a table. *All* of these expressions are run against *all* HTTP header values in an incoming request. If the expression matches *any* of the values, the filter fails.

Important Note:

If any regular expressions are configured in the [Configuring HTTP Header Regular Expressions](#) section, these expressions are run *before* Threatening Content Regular Expressions (TCRE) are run. For example, if you already configured a regular expression to extract the Base64-decoded value of the `Authentication` header value in the example above, the TCRE is run against this value instead of the attribute value that appears in the HTTP header.

You can add threatening content regular expressions using the **Add** button. You can edit or remove existing expressions by selecting them in the drop-down list, and clicking the **Edit** or **Delete** button.

You can enter the regular expressions manually or select them from the global **Black list** library of threatening content regular expressions. This library is pre-populated with a number of regular expressions that scan for common attack signatures. These include expressions to guard against common SQL injection-style attacks (for example, SQL `INSERT`, SQL `DELETE`, and so on), buffer overflow attacks (content longer than 1024 characters), and the presence of control characters in attribute values (ASCII control characters).

Enter or select an appropriate regular expression to restrict the value of the specified HTTP header. You can add a regular expression to the library by selecting the **Add/Edit** button. Enter a **Name** for the expression followed by the **Regular Expression**.

Query String Validation

Overview

The Enterprise Gateway can check the request *query string* to ensure that only properly configured name and value pairs appear. *Regular expressions* are used to test the attribute values. This enables you to make decisions on what to do with the message (for example, if the query string value is x, route to service x)

You can configure the following sections on the **Validate Query String** screen:

- **Enter Regular Expression:**
Query string values can be checked using regular expressions. You can select regular expressions from the global **White list** or enter them manually. For example, if you know that a query string must have a value of ABCD, a regular expression of `^ABCD$` is an exact match test.
- **Enter Threatening Content Regular Expression:**
You can select threatening content regular expressions from the global **Black list** to run against all query string names and values. These regular expressions identify common attack signatures (for example, SQL injection attacks).

You can configure the global **White list** and **Black list** libraries of regular expressions on the **Policies** tab in the Policy Studio.

Request Query String

The request query string is the portion of the URL that comes after the `?` character, and contains the request parameters. It is typically used for HTTP `GET` requests in which form data is submitted as name-value pairs on the URL. This contrasts with the HTTP `POST` method where the data is submitted in the body of the request. The following example shows a request URL that contains a query string:

```
http://hostname.com/services/getEmployee?first=john&last=smith
```

In this example, the query string is `first=john&last=smith`. Query strings consist of attribute name-value pairs, and each name-value pair is separated by the `&` character.

The **Query String Validation** filter can also operate on the form parameters submitted in an HTTP Form `POST`. Instead of encoding the request parameters in the query string, the client uses the `application/x-www-form-urlencoded` content-type, and submits the parameters in the HTTP `POST` body, for example:

```
POST /services/getEmployee HTTP/1.1
Host: localhost:8095
Content-Length: 21
SOAPAction: HelloService
Content-Type: application/x-www-form-urlencoded

first=john&last=smith
```

If the Enterprise Gateway receives an HTTP request body such as this, the **Query String Validation** filter can validate the form parameters.

Configuring Query String Attribute Regular Expressions

The **Enter Regular Expression** table displays the list of configured query string names together with the white list of regular expressions that restrict their values. For this filter to run successfully, *all* required attributes must be present in the request, and *all* must have the correct value.

The **Name** column shows the name of the query string attribute. The **Regular Expression** column shows the name of the regular expression that the Enterprise Gateway uses to restrict the value of the named query string attribute. A number of common regular expressions are available from the global **White list** library.

If the **Allow unspecified names** checkbox is selected, additional unnamed query string attributes are not filtered by the Enterprise Gateway. For example, this is useful if you are interested in filtering the content of only a small number of query string attributes but the request may contain many attributes. In such cases, you only need to filter those few attributes, and by selecting this checkbox, the Enterprise Gateway ignores all other query string attributes.

Configuring a Regular Expression

You can configure regular expressions by selecting the **Add**, **Edit**, and **Delete** buttons. The **Configure Regular Expression** dialog enables you to add or edit regular expressions to restrict the values request query string attributes. To configure a regular expression, perform the following steps:

1. Enter the name of the query string attribute in the **Name** field.
2. Select whether this request parameter is **Optional** or **Required** using the appropriate radio button. If it is **Required**, the parameter name *must* be present in the request. If the parameter is not present, the filter fails. If it is **Optional**, the attribute does not need to be present for the filter to pass.
3. You can enter the regular expression to restrict the value of the query string attribute manually or select it from the global **White list** library of regular expressions in the **Expression Name** drop-down list. A number of common regular expressions are provided (for example, alphanumeric values, dates, and email addresses).
You can use properties representing the values of message attributes to compare the value of the query string attribute with the value contained in a message attribute. Enter the \$ character in the **Regular Expression** field to view a list of available attributes. At runtime, the property is expanded to the corresponding attribute value, and compared to the query string attribute value that you want to check.
4. You can add a regular expression to the library by selecting the **Add/Edit** button. Enter a **Name** for the expression followed by the **Regular Expression**.

Advanced Settings

The **Advanced** section enables you to extract a portion of the query string attribute value that is run against the regular expression. The extracted substring can also be Base64 decoded if necessary. The following is an example of a URL containing a query string. The value of the `password` attribute is Base64 encoded, and must be extracted from the query string and decoded before it is run against the regular expression.

```
http://oracle.com/services?username=user&password=dXNlcg0K&dept=eng
```

You can extract the encoded value of the `password=` attribute value by specifying the string that occurs directly before the substring you want to extract, together with the string that occurs directly after the substring. Enter `password=` in the **Start substring** field, and `&` in the **End substring** field.

Important Note:

You must select the start and end substrings to ensure that the exact substring is extracted. For example, in this ex-

ample, `password=` (including the equals sign) should be entered in the **Start substring** field, and **not** `password` (without the equals sign).

By specifying the correct substrings, you are left with the Base64-encoded attribute value (`dXN1cg0K`). However, you still need to Base64 decode it before you can run a regular expression on it. Make sure to select the **Base64 decode** checkbox. The Base64-decoded password value is simply `user`. This is the value that you want to run the regular expression against.

By specifying the correct substrings, you are left with the Base64-encoded attribute value (`dXN1cg0K`). However, you still need to Base64 decode it before you can run a regular expression on it. Make sure to select the **Base64 decode** checkbox. The Base64-decoded password value is `user`. This is the value that you need to run the regular expression against.

Note:

If both **Start substring** and **End substring** fields are blank, the regular expression is run against the entire attribute value. Furthermore, if both fields are blank and the **Base64 decode** checkbox is selected, the entire attribute value is Base64 encoded before the regular expression is run against it.

Configuring Threatening Content Regular Expressions

The regular expressions entered in this section guard against the possibility of a query string attribute containing malicious content. The **Enter Threatening Content Regular Expression** table lists the **Black list** of regular expressions to run to ensure that the header values do not contain threatening content.

For example, to guard against an SQL `DELETE` attack, you can write a regular expression to identify SQL syntax and add it to this list. The **Threatening Content Regular Expressions** are listed in a table. *All* of these expressions are run against *all* attribute values in the query string. If the expression matches *any* of the values, the filter fails.

Important Note:

If any regular expressions are configured in the [Configuring Query String Regular Expressions](#) section, these expressions are run *before* the Threatening Content Regular Expressions (TCRE) are run. For example, if you have already configured a regular expression to extract the Base64-decoded value of the `password` query string attribute as in the example above, the TCRE is run against this value instead of the attribute value that appears in the query string.

You can add threatening content regular expressions using the **Add** button. You can edit or remove existing expressions by selecting them in the drop-down list and clicking the **Edit** or **Delete** button.

You can enter the regular expressions manually or select them from the global **Black list** library of threatening content regular expressions. This library is pre-populated with regular expressions that guard against common attack signatures. These include a expressions to guard against common SQL injection style attacks (for example, SQL `INSERT`, SQL `DELETE`, and so on), buffer overflow attacks (content longer than 1024 characters), and the presence of control characters in attribute values (ASCII Control Character).

Enter or select an appropriate regular expression to restrict the value of the specified query string. You can add a regular expression to the library by selecting the **Add/Edit** button. Enter a **Name** for the expression followed by the **Regular Expression**.

Schema Validation

Overview

The Enterprise Gateway can check that XML messages conform to the structure or format expected by the Web Service by validating those requests against XML Schemas. An XML Schema precisely defines the elements and attributes that constitute an instance of an XML document. It also specifies the data types of these elements to ensure that only appropriate data is allowed through to the Web Service.

For example, an XML Schema might stipulate that all requests to a particular Web Service must contain a `<name>` element, which contains at most a ten character string. If the Enterprise Gateway receives a message with an improperly formed `<name>` element, it rejects the message.

You can find the **Schema Validation** filter in the **Content Filtering** category of filters in the Policy Studio. Drag and drop the filter on to the policy where you want to perform schema validation. The **Schema Validation** dialog has three tabs, which are explained in this topic.

Schema to Use

Select one of the following options:

Use Schema from WSDL of Web Service:

Instead of selecting a specific XML schema to run in this filter, you can select this option to dynamically use the appropriate SOAP operation schema from the current Web Service Context. When this option is selected, this filter has an additional required message attribute named `webservice.context`, which must be provided. This enables you to share this filter to perform validation across multiple Web Services.

Select which XML Schema to validate messages with:

This is the default option. The tree hierarchy contains two top-level nodes: **Schema Cache** and **Web Services Repository**. When these nodes are expanded, the contained XML Schemas are displayed in the tree. Each schema has an associated checkbox, which you can use to select the schema to validate the incoming message. The following tutorials show how to import schemas into these global stores so that they can be selected in the **Schema Validation** filter:

- [Global Schema Cache](#)
- [Web Service Repository](#)

Note:

If you have a WSDL file that contains an XML Schema, and want to use this schema to validate the message, you can import the WSDL file into the **Web Services Repository**. The **WSDL Import Wizard** automatically creates a **Schema Validation** filter and incorporates it into the auto-generated policy. In this case, the top-level schema in the WSDL, which is imported into the **Web Services Repository**, is selected by default in the filter. In this way, if the schema imports other schemas, they are available to the filter at runtime when validating the message. For more details on importing WSDL files, see the [Web Services Repository](#) topic.

Part of Message to Match

A portion of the XML message can be extracted using an XPath expression. The Enterprise Gateway can then validate this portion against the specified XML Schema. For example, administrators may only want to validate the SOAP Body part of a SOAP message. In this case, they should enter or select an XPath expression that identifies the SOAP Body of the message. This portion should then be validated against an XML Schema that defines the structure of the SOAP Body for that particular message.

Click the **Add** or **Edit** buttons to add or edit an XPath expression using the **Enter XPath Expression** dialog. You can remove expressions by selecting the expression in the **XPath Expression** drop-down and clicking the **Delete** button.

On the **Enter XPath Expression** dialog, there are two ways to configure XPath expressions. For more details, see the following links:

- [Manual Configuration](#)
- [XPath Wizard](#)

The **XPath Wizard** assists administrators in creating correct and accurate XPath expressions. The wizard enables administrators to load an XML message and then run an XPath expression on it to determine what nodes are returned.

Advanced

The following settings are available on the **Advanced** tab:

Allow RPC Schema Validation:

When the **Allow RPC Schema Validation** checkbox is selected, the filter makes a *best attempt* to validate an RPC-encoded SOAP message. An RPC-encoded message is defined in the WSDL as having an operation with the following characteristics:

- The `style` attribute of the `<soap:operation>` element is set to `document`.
- The `use` attribute of the `<soap:body>` element is set to `rpc`.

For details on the possible values for these attributes, see [Section 3.5](#) [http://www.w3.org/TR/wsdl#_soap:body] of the WSDL specification.

The problem with RPC-encoded SOAP messages in terms of schema validation is that the schema contained in the WSDL file does not necessarily fully define the format of the SOAP message, unlike with `document-literal` style messages. With an RPC-encoded operation, the format of the message can be defined by a combination of the SOAP operation name, WSDL message parts, and schema-defined types. As a result, the schema extracted from a WSDL file may not be able to validate a message.

Another problem with RPC-encoded messages is that type information is included in each element that appears in the SOAP message. For such element definitions to be validated by a schema, the type declarations must be removed, which is precisely what the **Schema Validation** filter does if the checkbox is selected on this tab. It removes the type declarations and then makes a *best attempt* to validate the message.

However, as explained earlier, if some of the elements in the SOAP message are taken from the WSDL file instead of the schema (for example, when the SOAP operation name in the WSDL file is used as the wrapper element beneath the SOAP Body instead of a schema-defined type), the schema is **not** able to validate the message.

Inline MTOM Attachments into Message:

Message Transmission Optimization Mechanism (MTOM) provides a way to send binary data to Web Services in standard SOAP messages. MTOM leverages the include mechanism defined by XML Optimized Packaging (XOP), whereby binary data can be sent as a MIME attachment (similar to SOAP with Attachments) to a SOAP message. The binary data can then be referenced from within the SOAP message using the `<xop:Include>` element.

The following SOAP request contains a binary image that has been Base64-encoded so that it can be inserted as the contents of the `<image>` element:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <uploadGraphic xmlns="www.example.org">
```

```

    <image>/aWKKapGGyQ=</image>
  </uploadGraphic>
</soap:Body>
</soap:Envelope>

```

When this message is converted to an MTOM message by the Enterprise Gateway (for example, using the [Extract MTOM Content](#) filter) the Base64-encoded content from the `<image>` element is replaced with an `<xop:Include>` element. This contains a reference to a newly created MIME part that contains the binary content. The following request shows the resulting MTOM message:

```

POST /services/uploadImages HTTP/1.1
Host: SOAPbox
Content-Type: Multipart/Related;boundary=MIME_boundary;
    type="application/xop+xml";
    start="<mymessage.xml@example.org>";
    start-info="text/xml"

--MIME_boundary
Content-Type: application/xop+xml;
    charset=UTF-8;
    type="text/xml"
Content-Transfer-Encoding: 8bit
Content-ID: <mymessage.xml@example.org>

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <uploadGraphic xmlns="www.example.org">
      <image>
        <xop:Include xmlns:xop='http://www.w3.org/2004/08/xop/include'
          href='cid:http://example.org/myimage.gif' />
      </image>
    </uploadGraphic>
  </soap:Body>
</soap:Envelope>

--MIME_boundary
Content-Type: image/gif
Content-Transfer-Encoding: binary
Content-ID: <http://example.org/myimage.gif>

// binary octets for image

--MIME_boundary

```

When attempting to validate the MTOM message with an XML Schema, it is crucial that you are aware of the format of the `<image>` element. Will it contain the Base64-encoded binary data, or will it contain the `<xop:include>` element with a reference to a MIME part?

For example, the XML Schema definition for `<image>` element might look as follows:

```
<xsd:element name="image" maxOccurs="1" minOccurs="1"
  type="xsd:base64Binary"
  xmime:expectedContentTypes="*/*"
  xsi:schemaLocation="http://www.w3.org/2005/05/xmlmime"
  xmlns:xmime="http://www.w3.org/2005/05/xmlmime"/>
```

In this case, the XML Schema validator expects the contents of the `<image>` element to be `base64Binary`. However, if the message has been formatted as an MTOM message, the `<image>` element contains a child element, `<xop:Include>` that the schema knows nothing about. This causes the schema validator to report an error and the schema validation fails.

To resolve this issue, select the **Inline MTOM Attachments into Message** checkbox on the **Advanced** tab. At runtime, the schema validator replaces the value of the `<xop:Include>` element with the Base64-encoded contents of the MIME part to which it refers. This means that the message now adheres to the definition of the `<image>` element in the XML Schema (the element contains data of type `base64Binary`).

This standard procedure of interpreting XOP messages is described in [Section 3.2 Interpreting XOP Packages](http://www.w3.org/TR/2004/CR-xop10-20040826/#interpreting_xop_packages) [http://www.w3.org/TR/2004/CR-xop10-20040826/#interpreting_xop_packages] of the XML-binary Optimized Packaging (XOP) specification.

Reporting Schema Validation Errors

When a Schema validation check fails, the validation errors are stored in the `xsd.errors` Oracle message attribute. You can return an appropriate SOAP Fault to the client by writing out the contents of this message attribute.

For example, you can do this by configuring a [Set Message](#) filter to write a custom response message back to the client. Place the **Set Message** filter on the failure path of the **Schema Validation** filter. You can enter the following sample SOAP Fault message in the **Set Message** filter. Notice the use of the `${xsd.errors}` message attribute in the `<Reason>` element.

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Receiver</env:Value>
        <env:Subcode>
          <env:Value xmlns:fault="http://www.Oracle.com/soapfaults">
            fault:MessageBlocked
          </env:Value>
        </env:Subcode>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en">
          ${xsd.errors}
        </env:Text>
      </env:Reason>
      <env:Detail xmlns:fault="http://www.Oracle.com/soapfaults"
        fault:type="faultDetails">
      </env:Detail>
    </env:Fault>
```

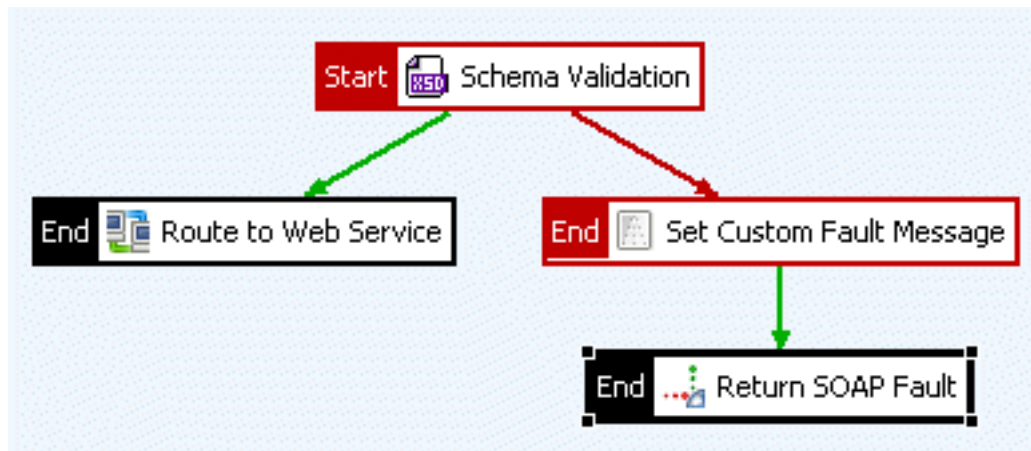


```
</env:Body>>
</env:Envelope>
```

At runtime, the error reported by the schema validator is set in the message. You can then return the SOAP Fault to the client using a [Reflect](#) filter. The following example shows a SOAP Fault containing a typical schema validation error:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Receiver</env:Value>
        <env:Subcode>
          <env:Value xmlns:fault="http://www.Oracle.com/soapfaults">
            fault:MessageBlocked
          </env:Value>
        </env:Subcode>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en">
          [XSD Error: Unknown element 'id' (line: 2, column: 8)]
        </env:Text>
      </env:Reason>
      <env:Detail xmlns:fault="http://www.Oracle.com/soapfaults"
        fault:type="faultDetails">
      </env:Detail>
    </env:Fault>
  </env:Body>>
</env:Envelope>
```

The following screenshot shows how to use the **Set Message** filter to return a customized SOAP Fault in a policy. If the **Schema Validation** filter succeeds, the message is routed on to the target Web Service. However, if the schema validation fails, the **Set Message** filter (named **Set Custom Fault Message**) is invoked. The filter sets the contents of the `xsd.errors` message attribute (the schema validation errors) to the custom SOAP Fault message as shown in the example error. The **Reflect** filter (named **Return SOAP Fault**) then writes the message back to the client.



Validate Message Attributes

Overview

Filters configured in a circuit before the **Validate Message Attributes** filter can generate message attributes and store them in the message. The **Validate Message Attributes** filter can use regular expressions to check the values of these message attributes. This enables you to make decisions on what to do with the message (for example, if the attribute value is *x*, route to service *x*)

You can configure the following sections on the **Validate Message Attributes** screen:

- **Enter Regular Expression:**
You can configure a list of message attributes so that each is checked against a regular expression from the global **White list** library or against a manually configured expression. This check ensures that the value of the message attribute is acceptable. For example, if you know that a message attribute must have a value of *ABCD*, a regular expression of *^ABCD\$* is an exact match test.
- **Enter Threatening Content Regular Expression:**
You can select threatening content regular expressions from the global **Black list** to run against each message attribute. These regular expressions identify common attack signatures (for example, SQL injection attacks, ASCII control characters, XML entity expansion attacks, and so on).

You can configure the global **White list** and **Black list** libraries of regular expressions on the **Policies** tab in the Policy Studio.

Configuring Message Attribute Regular Expressions

The **Enter Regular Expression** table displays the list of configured message attribute names together with the **White list** of regular expressions that restrict their values. For this filter to run successfully, *all* configured attribute checks must have values matching the configured regular expressions.

The **Name** column shows the name of the attribute. The **Regular Expression** column shows the name of the regular expression that the Enterprise Gateway uses to restrict the value of the named attribute. A number of common regular expressions are available from the global **White list** library.

Configuring a Regular Expression

You can configure regular expressions by selecting the **Add**, **Edit**, and **Delete** buttons. The **Configure Regular Expression** dialog enables you to add or edit regular expressions to restrict the values of message attributes. To configure a regular expression, perform the following steps:

1. Enter the name of the message attribute in the **Name** field.
2. Select whether this attribute is **Optional** or **Required** using the appropriate radio button. If it is **Required**, the attribute *must* be present in the request. If the attribute is not present, the filter fails. If it is **Optional**, the attribute does not need to be present for the filter to pass.
3. You can enter the regular expression to restrict the value of the attribute manually or select it from the global **White list** library of regular expressions in the **Expression Name** drop-down list. A number of common regular expressions are provided (for example, alphanumeric values, dates, and email addresses).
You can use a property representing the value of a message attribute to compare the value of a message attribute with another attribute. Enter the *\$* character in the **Regular Expression** field to view a list of available attributes. At runtime, the property is replaced by the corresponding attribute value and compared to the message attribute value that you want to check.
4. You can add a regular expression to the library by selecting the **Add/Edit** button. Enter a **Name** for the expression followed by the **Regular Expression**.

Advanced Settings

The **Advanced** section enables you to extract a portion of the attribute value which is run against the regular expression. The extracted substring can also be Base64 decoded if necessary.

Threatening Content Regular Expressions

The regular expressions entered in this section guard against the possibility of a message attribute containing malicious content. The **Enter Threatening Content Regular Expression** table lists the **Black list** of regular expressions that are run against all message attributes.

For example, to guard against a SQL `DELETE` attack, you can write a regular expression to identify SQL syntax and add to this list. The **Threatening Content Regular Expressions** are listed in a table. *All* of these expressions are run against *all* message attributes configured in the **Regular Expression** table above. If the expression matches *any* attribute values, the filter fails.

Important Note: If any regular expressions are configured in the [Message Attribute Regular Expressions](#) section, these expressions are run *before* the Threatening Content Regular Expressions (TCRE) are run. For example, if you have already configured a regular expression to extract the Base64-decoded attribute value, the TCRE is run against this value instead of the attribute value stored in the message.

You can add threatening content regular expressions using the **Add** button. You can edit or remove existing expressions by selecting them in the drop-down list, and clicking the **Edit** or **Delete** button.

You can enter the regular expressions manually or select them from the global **Black list** library of threatening content regular expressions. This library is pre-populated with regular expressions that scan for common attack signatures. These include expressions to guard against common SQL injection-style attacks (for example, SQL `INSERT`, SQL `DELETE`, and so on), buffer overflow attacks (content longer than 1024 characters), and the presence of control characters in attribute values (ASCII control characters).

Enter or select an appropriate regular expression to scan all message attributes for threatening content. You can add a regular expression to the library by selecting the **Add/Edit** button. Enter a **Name** for the expression followed by the **Regular Expression**.

Validate Timestamp

Overview

You can use the **Validate Timestamp** filter to validate a timestamp that has been stored in a message attribute by a previous filter in a policy.

For example, you can extract the value of a `wsu:Created` element from a WS-Security token and store it in a created attribute using the **Retrieve from Message** filter in the **Attributes** category. You can then use the **Validate Timestamp** filter to ensure that the created timestamp is not *after* the current time.

Similarly, you can use the **Retrieve from Message** filter to extract the value of the `wsu:Expires` element and store it in a timestamp message attribute. You can use the **Validate Timestamp** filter to check that the timestamp is not *before* the current time.

This ensures that the current time is between the `Created` time and the `Expires` time. By taking into account the drift time (to resolve discrepancies between clock times on the machine that generated the timestamp, and the machine running the Enterprise Gateway), this ensures that the current time is after the `Created` time minus the drift time, and before the `Expires` time plus the drift time. The current time is within the following timeframe:

```
[Created Time - Drift, Expiry Time + Drift]
```

Important Note:

If you wish to validate the timestamp stored in a WS-Security Username Token or SAML assertion, you can use the **WS-Security Username Token Authentication**, **SAML Authentication**, **SAML Authorization**, or **SAML Attribute** filters to perform this validation. You can use the **Validate Timestamp** filter to validate non-standard timestamps, such as those not transmitted in WS-Security tokens or SAML assertions.

The **Validate Timestamp** filter does not require an entire WS-Utility Timestamp element (unlike the **Insert Timestamp** filter). Instead, this filter requires a simple date-formatted string.

Configuration

Complete the following fields to configure the Enterprise Gateway to validate a timestamp that has been stored in a message attribute:

Name:

Enter a name for the filter.

Attribute Containing Timestamp:

Enter the name of the message attribute that contains the value of the timestamp in this field. You must configure a predecessor of this filter to extract the timestamp from the message and store it in the specified attribute (for example, the **Retrieve from Message** filter in the **Attributes** filter).

Format of Timestamp:

Enter the format of the timestamp that is contained in the specified message attribute. The default date/time format is `yyyy-MM-dd'T'HH:mm:ss.SSS'Z'`, which can be altered if necessary. For more information on how to use this format, see the Javadoc for the [java.text.SimpleDateFormat](http://java.sun.com/javase/6/docs/api/index.html) [http://java.sun.com/javase/6/docs/api/index.html] class.

Timezone:

Select the time zone to use to interpret the time stored in the message attribute selected above. The default option is GMT.

Drift (secs):

Specify the drift time to use when determining whether or not the current time falls within a certain time interval. The drift time can be used to account for differences in the clock times of the machine running the Enterprise Gateway and the machine on which the timestamp was generated.

Timestamp must be in the past:

The time in the timestamp must be *before* the time at which the server validates the timestamp. This is used for validating a timestamp that represents a `Created` time (the created time must be before the validation time).

Timestamp must be in the future:

The time in the timestamp must be *after* the time at which the server validates the timestamp. This is used for validating a timestamp that represents an `Expires` time (the expiry time must be some time in the future relative to the validation time).

Validate REST Request

Overview

The **Validate REST Request** filter enables you to validate the following aspects of a REST request:

- The HTTP method used in the request.
- Each of the query string parameters against a set of restrictive conditions called a *Request Parameter Restriction*.

For example, a Request Parameter Restriction enables you to specify the expected data type of a named parameter, a regular expression for the parameter value, the minimum and maximum length of a string parameter, the minimum and maximum value of a numeric parameter, and so on.

This filter is found in the **Content Filtering** category in the Policy Studio. For details on how to create a REST request, see the [Create REST Request](#) filter.

General Configuration

Complete the following fields on the **Validate REST Request** screen:

Name:

Enter an appropriate name for the filter.

HTTP Method:

Enter or select the HTTP method of the incoming message (for example, POST, GET, DELETE, and so on). The HTTP method of the incoming request must match the method specified here.

REST Request Parameter Restrictions

Click the **Add** button to configure restrictions on the values of query string parameters. You can configure the following settings in the **REST Request Parameter Restrictions** dialog:

REST Request Parameter Details

Complete the following fields:

Description	The description entered here is displayed in the REST Request Parameter Restriction table on the main filter screen (for example, Name parameter must be string no longer than 10 characters). This field is mandatory.
Request Parameter Name	The name of the query string parameter to validate (for example, name). This field is mandatory.
Request Parameter Type	The data type of the query string parameter (for example, string or integer). You can enter a value or select from the drop-down list. This field is mandatory.
Fail if request parameter not found	Select this option if the specified request parameter must be present in the request query string. The filter fails if the parameter is not found.

Request Parameter Restrictions

Complete the following fields:

Min Length	Specifies the minimum number of characters or list items allowed (for example, 0). The default value of -1 means that this restriction is ignored.
Max Length	Specifies the exact number of characters or list items allowed (for example, 10). The default value of -1 means that this restriction is ignored.
Regular Expression	Specifies the exact sequence of characters that are permitted using a regular expression (for example, <code>^[a-zA-Z\s]*\$</code>).
Enumeration	Specifies a list of permitted values. Click Add to enter an item in the list, and Click OK . Repeat as necessary to add multiple values.

Advanced Restrictions

Complete the following fields:

Greater than	Specifies that the value entered in the Minimum Value field represents an exclusive lower bound (the value must greater than this).
Greater than or Equal to	Specifies that the value entered in the Minimum Value field represents an inclusive lower bound (the value must greater than or equal to this).
Minimum Value	Specifies the lower bounds for numeric values (for example, the value must greater than 20).
Less than	Specifies that the value entered in the Maximum Value field represents an exclusive lower bound (the value must less than this).
Less than or Equal to	Specifies that the value entered in the Maximum Value field represents an inclusive lower bound (the value must less than or equal to this).
Maximum Value	Specifies the upper bounds for numeric values (for example, the value must less than or equal to 30).
Max Total Digits for Number	Specifies the maximum number of digits allowed for a numeric data type. The default value of -1 means that this restriction is ignored.
Max Digits in Fraction Part of Number	Specifies the exact number of digits allowed in the fraction part of a numeric type. For example, the number 1.23 has two fraction digits (two numbers after the decimal point). The default value of -1 means that this restriction is ignored.
Whitespace	<p>Specifies how white space is handled (for example, line feeds, tabs, spaces, and carriage returns). You can enter one of the following values:</p> <ul style="list-style-type: none"> <code>Preserve</code> means the XML processor preserves (does not remove) any white space characters. <code>Replace</code> means the XML processor replaces any white space characters (line feeds, tabs, and carriage returns) with spaces.

	<ul style="list-style-type: none">Remove means the XML processor removes any white space characters (line feeds, tabs, spaces, carriage returns are replaced with spaces, leading and trailing spaces are removed, and multiple spaces are reduced to a single space).
--	--

Fail if unspecified request parameter found:

If a request parameter is found on the incoming query string that has not been specified in the **REST Request Parameter Restrictions** table, this filter fails. You can use this option to guard against processing a query string containing a potentially malicious request parameter (for example, `/uri?number=2&badParam=System.exit(1);`).

XML Complexity

Overview

Parsing XML documents is a notoriously processor-intensive activity. This can be exploited by hackers by sending large and complex XML messages to Web Services in a type of denial-of-service attack attempting to overload them. The **XML Complexity** filter can protect against such attacks by performing the following checks on an incoming XML message:

- Checking the total number of nodes contained in the XML message.
- Ensuring that the message does not contain deeply nested levels of XML nodes.
- Making sure that elements in the XML message can only contain a specified maximum number of child elements.
- Making sure that each element can have a maximum number of attributes.

By performing these checks, the Enterprise Gateway can protect back-end Web Services from having to process large and potentially complex XML messages.

You can also use the **XML Complexity** filter on the Web Service response to prevent a dictionary attack. For example, if the Web Service is a phone book service, a `name=*` parameter could return all entries.

Configuration

The **XML Complexity** filter should be configured as the first filter in the policy that processes the XML body. This enables this filter to block any excessively large or complex XML message *before* any other filters attempt to process the XML.

To configure the **XML Complexity** filter, complete the following fields:

Name:

Enter an appropriate name for this filter.

Maximum Total Number of Nodes:

Specify the maximum number of nodes that you want to allow in an XML message. Note that this number does not include text nodes or comments. You can use the [Message Size](#) filter to stop large text nodes or comments.

Maximum Number of Levels of Descendant Nodes:

Enter the maximum number of descendant nodes that an element is allowed to have. Again, this number does not include text nodes or comments.

Maximum Number of Child Nodes per Node:

Enter the maximum number of child nodes that an element in an XML message is allowed to have.

Maximum Number of Attributes per Node:

Enter the maximum number of attributes that an element is allowed to have.

Threatening Content

Overview

The **Threatening Content** filter can run a series of regular expressions that identify different attack signatures against request messages to check if they contain threatening content. Each expression identifies a particular attack signature, which can run against different parts of the request, including the request body, HTTP headers, and the request query string. In addition, you can configure the MIME types on which the **Threatening Content** filter operates.

The threatening content regular expressions are stored in the global **Blacklist** library, which is displayed as a top-level node on the **Policies** tab in Policy Studio. By default, this library contains regular expressions to identify SQL syntax to guard against SQL injection attacks, DOCTYPE DTD references to avoid against DTD expansion attacks, Java exception stack trace information to prevent call stack information getting returned to the client, and expressions to identify other types of attack signature.

The **Threatening Content** filter is available from the **Content Filtering** category of filters. Drag and drop this filter on to the circuit editor, and enter a name for the filter in the **Name** field. The next sections describe how to configure the other tabs on this filter screen.

Scanning Details

To configure the scanning details, complete the following sections:

Additional message parts to scan:

This section configures what parts of the incoming request are scanned for threatening content. By default, the **Threatening Content** filter acts on the request body. However, it can also scan the HTTP headers and the request query string for threatening content. Select the appropriate checkboxes to indicate what additional parts of the request message you want to scan.

Blacklist:

The table lists all the regular expressions that have been added to the global **Blacklist**. These regular expressions are used to identify threatening content. For example, there are regular expressions to match SQL syntax, ASCII control characters, and XML processing instructions, all of which can be used to attack a Web Service. For more information on how to configure these global regular expressions, see the [Blacklist](#) topic.

Select the regular expressions that you want to run against incoming requests using the checkboxes in the table. You can add new expressions using the **Add** button. It is important to note that when adding new regular expressions on the **Add Regular Expression** dialog, the expressions are added to the global **Blacklist** library.

You can edit or remove existing regular expressions by selecting the expression in the tree, and selecting the **Edit** or **Delete** button.

MIME Types

The **MIME Types** tab lists the MIME types to be scanned for incoming messages. By default, all text- and XML-related types are scanned for threatening content. However, you can select any type from the list.

Similar to the way in which the **Blacklist** regular expressions are global, so too are the MIME types. You can add these globally by selecting **Settings** -> **Settings** -> **MIME/DIME** from the main menu.

You can add new types by selecting the **Add** button and entering a type name and corresponding extension on the **Configure MIME/DIME Type** dialog. You can enter a list of extensions by separating them with spaces. You can edit or delete existing types by selecting the **Edit** and **Delete** buttons.

WS-Security Policy Layout

Overview

Web Services can use the WS-Policy specification to advertise the security requirements that clients must adhere to in order to successfully connect and send messages to the service. For example, a typical WS-Policy would mandate that the SOAP request Body be signed and encrypted (using XML Signature and XML Encryption) and that a signed WS-Utility Timestamp must be present in a WS-Security header.

To guarantee that the security tokens used to *protect* the message are added to the request in the most efficient and interoperable manner, WS-Policy uses the `<wsp:Layout>` assertion. The semantics of this assertion are implemented by this filter and are outlined in the configuration details in the next section.

Configuration

To check a SOAP message for a particular WS-Policy layout, complete the following fields:

Name:

Enter an intuitive name for this filter (for example, `Check SOAP Request for Lax Layout`).

Actor:

Enter the name of the SOAP Actor/Role where the security tokens are present.

Select Required Layout Type:

Select the required layout from the following WS-Policy options:

- **Strict:**
Select this option to check that a SOAP message adheres to the WS-Policy strict layout rules. For more information, see the WS-Policy specification.
- **Lax:**
Select this option if you want to ensure that the security tokens in the SOAP header have been inserted according to the Lax WS-Policy layout rules. The WS-Policy Lax rules are effectively identical to those stipulated by the SOAP Message Security specification.
- **LaxTimestampFirst:**
This layout option ensures that the WS-Policy Lax rules have been followed, but also checks to make sure that the WS-Utility Timestamp is the first security token in the WS-Security header.
- **LaxTimestampLast:**
This option ensures that the WS-Utility Timestamp is the last security token in the WS-Security header and that all other Lax layout rules have been followed.

WS-Security Version:

The layout rules for WS-Security versions 1.0 and 1.1 are slightly different. Select the version of the layout rules that you wish to apply to SOAP requests. For details on the differences between these versions, see the WS-Security specifications.

McAfee Virus Scanner

Overview

The **McAfee Anti-Virus** filter scans incoming HTTP requests and their attachments for viruses and exploits. For example, if a virus is detected in a MIME attachment or in the XML message body, the Enterprise Gateway can reject the entire message and return a SOAP Fault to the client. In addition, this filter supports cleaning of messages from infections such as viruses and exploits. It also provides scan type presets for different detection levels, and reports overall message status after scanning.

Important: This filter is currently available on Windows and Linux only.

Prerequisites

McAfee virus scanner integration requires the McAfee 5400 Scan Engine.

Enterprise Gateway

When adding third-party binaries to the Enterprise Gateway, you must perform the following steps:

1. Add the binary files as follows:
 - Add .dat files from the McAfee 5400 Scan Engine to the `install-dir/conf/plugin/mcafee/datv2` directory, except for `config.dat` which must be added to `install-dir/platform/lib` or `install-dir\win32\lib`.
 - Add .jar files to the `install-dir/ext/lib` directory.
 - Add .dll files to the `install-dir\win32\lib` directory.
 - Add .so files to the `install-dir/platform/lib` directory.
2. Restart the Enterprise Gateway.

Policy Studio

When adding third-party binaries to the Policy Studio, you must perform the following steps:

1. Add .jar files to the `install-dir/plugins/thirdparty.runtime.dependencies_6.0.3` directory.
2. Restart the Policy Studio.

Configuring a McAfee Anti-Virus Filter

To configure the **McAfee Anti-Virus** filter, perform the following steps:

1. Enter an appropriate name in the **Name** field.
2. Select a **Scan type** from the drop-down box. The available options are as follows:

Normal	Processes the entire message detecting exploits and viruses in the message headers, macros, multi-file archives, executables, MIME-encoded/UU-encoded/XX-encoded/BinHex and TNEF/IMC format files. Performs heuristic analysis to find new viruses and potentially unwanted programs. This is the default scan type.
Fast	Detects infections in the top level of each message part, such as exploits that use headers and multiple bodies. The

	detection is less precise, but the performance is better if the top-level object is infected.
Multi-pass	Combines the Normal and Fast scan types. The Fast scan (pass 1) runs first on the whole message with no cleaning. The scanner stops if it finds an infected object, and if the clean type is set to No cleaning , the scanner reports the infection, or otherwise deletes the message. If pass 1 does not detect any virus or exploit, the Normal scan (pass 2) runs with the specified clean type and provides more precise detection.
Custom	Enables you to set the Custom options described in the next section. This provides compatibility with previous Enterprise Gateway versions. Note: When existing circuits are upgraded to the current Enterprise Gateway version, the McAfee Anti-Virus filter scan type is set to Custom and the clean type is set to No cleaning for backward compatibility.

3. Select a **Clean type** from the drop-down box. The available options are as follows:

No cleaning	Fails if any infection is detected. This is the default clean type.
Always remove infected parts	Removes the infected message part, and does not try to repair it.
Attempt to repair infected parts	Attempts to repair the found infection (if repairable), otherwise deletes the infected message part.

Configuring Custom Options

When you configure a custom scan type, the following **Custom options** are available:

Decompress Archives:

This instructs the filter to scan each file in an archive for viruses. Types of archived files include the ZIP, JAR, TAR, ARJ, LHA, PKARC, PKZIP, RAR, WinACE, BZip, and Zcompress formats.

Decompress Executables:

Executables are sometimes compressed to decrease overall message size. In such cases, any embedded viruses are also compressed and may be missed by conventional scans. If this option is selected, the filter decompresses the executable before scanning it for viruses.

Fail Any Macros:

A *macro* is a series of commands that can be invoked in a single command or keystroke. While calling the macro can appear to be harmless, the initiated command sequence may be harmful. Macros are usually configured to run automatically when the host document is opened. When this option is selected, the Enterprise Gateway fails if any macro is detected in a compound document (whether it matches a virus signature or not). An appropriate SOAP Fault is returned to the client.

Heuristic Program Analysis:

A heuristic virus detection algorithm runs a series of probing tests on a file in an attempt to solicit virus-like behavior from it. Based on the results of these tests, the algorithm can then make an educated guess on whether the file represents a potential threat or not. For example, programs that attempt to modify or delete files, invoke email clients, or replicate themselves all display virus-like behavior and so may be treated as viruses by the scanner.

The major advantage of this type of analysis is that new viruses can be detected. With the signature detection method, the scanner attempts to find a fixed number of known virus signatures in a file. Because the number of known signatures is fixed, new or unknown viruses can not be detected. If this option is selected, the filter runs heuristic analysis on executables only.

Heuristic Macro Analysis:

When this option is enabled, the filter runs heuristic detection analysis on macros contained in any body parts of the message. If any viruses are detected, the message is blocked. If this option is selected, the Enterprise Gateway searches for virus signatures in the respective body parts of a MIME message. However, it can only search for *known* viruses using this method. It is important to note that macros embedded in MIME parts are also scanned for virus signatures.

Scan Embedded Scripts:

The Enterprise Gateway can scan MIME parts, such as HTML documents, for embedded scripts. If this option is selected, the filter scans for embedded scripts.

Scan for Test Files:

When this option is selected, the Enterprise Gateway fails if it encounters an anti-virus test file (for example, `eicar.com`). This is a convenient way to check that the anti-virus filter successfully detects known viruses.

Reporting Message Status

When the scan is complete, the **McAfee Anti-Virus** filter reports the overall message status in the `mcafee.status` message attribute, which is generated by the filter. This reflects the overall status of the scan for all message parts, and includes one of the following values:

NOVIRUS	No virus or exploit detected in the message.
INFECTED	Infection detected in the message.
REPAIRED	Message repaired.
REMOVED	Some or all message parts successfully removed.
REPAIRED, REMOVED	Some message parts successfully repaired and some others removed.

Loading McAfee Updates

When the **McAfee Anti-Virus** filter has been loaded, it searches for virus definitions in the `install-dir\conf\plugin\mcafee\datv2` directory. When these have been loaded, it periodically checks for the presence of an `install-dir\conf\plugin\mcafee\datv2.new` directory.

If the `datv2.new` directory is found, the scanner is stopped and the `datv2` directory is renamed to `datv2.0`. If a `datv2.0` directory already exists from a previous rollover, a `datv2.1` directory is created instead, and so on, until an unused index is used. This means that the server never deletes the old files, and rolls them out of the way.

When the engine is stopped and restarted, any messages that require scanning are suspended until the restart completes. In addition, an initiated reload is suspended until all currently active scans are completed.

Important Note

Like all file system scanning approaches, there is an inherent ordering problem. If you create the `datv2.new` directory

before copying the files into the directory, the scanner may pick up the new directory before it is ready to be used. For example, on Windows, you may experience problems if you enter the following commands from the *install-dir\conf\plugin\mcafee* directory:

```
mkdir datv2.new
copy c:\mcafee\newfiles\*. * datv2.new
```

You can use the following commands to prevent this problem:

```
mkdir datv2.tmp
copy c:\mcafee\newfiles\*. * datv2.tmp
rename datv2.tmp datv2.new
```

In other words, create a temporary folder, copy the files into this folder, and then rename the temporary folder to *datv2.new*. In this way, the scanner is guaranteed to pick up the virus definition files when it detects the new directory.

On Linux and Solaris, the same approach applies, but the location of the file and the commands used are different. For example, enter the following commands from the *install-dir/conf/plugin/mcafee* directory:

```
mkdir datv2.tmp
cp /var/tmp/mcafee/newfiles/*. * datv2.tmp
mv datv2.tmp datv2.new
```


ClamAV Anti-Virus

Overview

The Enterprise Gateway can check messages for viruses by connecting to a ClamAV daemon running on network. The ClamAV daemon inspects the message and if the daemon finds a virus, it returns a corresponding response to the Enterprise Gateway, which can then block the message, if necessary.

Configuration

Complete the following fields to configure the **ClamAV Anti-Virus** filter:

Name:

Enter an appropriate name for this filter.

ClamAV Daemon Host:

Enter the host name of the machine on which the ClamAV daemon is running.

ClamAV Daemon Port Number:

Enter the port on which the ClamAV daemon is listening.

Sophos Anti-Virus Filter

Overview

The **Sophos Anti-Virus** filter uses the Sophos Anti-Virus Interface (SAVI) to screen messages for viruses. You can configure the behavior of the Sophos library using configuration options available in the **Sophos Anti-Virus** filter.

Important Note:

Because the Enterprise Gateway does not ship with any Sophos binaries, the Enterprise Gateway must be installed on the same machine as the Sophos AV distribution. On Linux or Solaris, before starting the Enterprise Gateway, ensure that the Sophos AV `lib` directory is on your `LD_LIBRARY_PATH`. Similarly, on Windows, this directory must be on the system `PATH` (the Sophos installation automatically puts this directory on the system path).

Prerequisites

Sophos integration requires the Sophos SAV Interface version 4.8.

Enterprise Gateway

When adding third-party binaries to the Enterprise Gateway, you must perform the following steps:

1. Add the binary files as follows:
 - Add `.jar` files to the `InstallDir/ext/lib` directory.
 - Add `.dll` files to the `InstallDir\win32\lib` directory.
 - Add `.so` files to the `InstallDir/platform/lib` directory.
2. Restart the Enterprise Gateway.

Policy Studio

When adding third-party binaries to the Policy Studio, you must perform the following steps:

1. Add `.jar` files to the `InstallDir/plugins/thirdparty.runtime.dependencies_6.0.3` directory.
2. Restart the Policy Studio.

General Settings

Specify the following general setting:

Name:

Enter an appropriate name for this filter.

Sophos Configuration Settings

All SAVI configuration options take the form of a name-value pair. Each name is unique and its corresponding value controls specific behavior in the Sophos anti-virus library (for example, decompress `.zip` files to examine their content). You can specify these SAVI configuration settings in the **Sophos configuration settings** section:

Name:

The **Sophos Anti-Virus** filter ships with two sets of default configuration settings: one for UNIX-based platforms, and the other for Windows platforms. Select the appropriate configuration settings for your target platform from the drop-down list.

You can create a new set of configuration options by clicking the **Add** button, and adding the name-value pairs to the table provided. For convenience, you can base a new configuration set on a previously existing one, including the default Windows and UNIX sets. In this way, you can create a new configuration set that *inherits* from the default set, and then

adds more configuration options.

To add a new configuration name-value pair, click the **Add** button under the table, and configure the following fields in the dialog:

Name:

Enter a name for the SAVI configuration option. This name must be available in the version of the SAVI library that is used by the Enterprise Gateway. Please refer to your SAVI documentation for a complete reference on available options.

Value:

Enter an appropriate value for the SAVI configuration option entered above. Please refer to your SAVI documentation for more information on acceptable values for this configuration option.

Type:

Select the appropriate type of this configuration option from the drop-down list. Please refer your SAVI documentation for more information on the type of the value for this configuration option.

Add HTTP Header

Overview

The Enterprise Gateway can add HTTP headers to a message as it passes through a policy. It can also set a Base64-encoded value for the header. For example, you can use the **Add HTTP Header** filter to add a message ID to an HTTP header. This message ID can then be forwarded to the destination Web Service, where messages can be indexed and tracked by their IDs. In this way, you can create a complete *audit trail* of the message from the time it is received by the Enterprise Gateway, until it is processed by the back-end system.

Each message being processed by the Enterprise Gateway is assigned a unique transaction ID, which is stored in the `id` message attribute. You can use the `${id}` property to represent the value of the unique message ID. At runtime, this property is expanded to the value of the `id` message attribute.

Configuration

To configure the **Add HTTP Header** filter, complete the following fields:

Name:

Enter an appropriate name for the filter.

HTTP Header Name:

Specify the name of the HTTP header to add to the message.

HTTP Header Value:

Enter the value of the new HTTP header. You can enter properties to represent message attributes. At runtime, the Enterprise Gateway expands these properties to the current value of the corresponding message attribute. For example, the `${id}` property is replaced by the value of the current message ID. Properties have the following syntax:

`${message_attribute}`

Base64 Encode:

Select this checkbox to Base64 encode the HTTP header value. For example, you should use this if the header value is an X.509 certificate.

Add header to body:

Select this radio button to add the HTTP header to the message body.

Add header to HTTP headers attribute:

Select this radio button to add the HTTP header to the `http.headers` message attribute.

Add XML Node

Overview

You can use this filter to add an XML element, attribute, text, comment, or CDATA section node to an XML message. The new node is inserted into the location specified by an XPath expression or a SOAP actor/role. XPath is a query language that enables you to select nodes in an XML document. A SOAP actor/role provides a way of identifying a particular WS-Security block in a message.

General Configuration

You can configure the following general setting:

Name:

Enter a suitable name that reflects the role of the filter. For example, if the purpose of this filter is to add an <Address> element to the message, it would be appropriate to name this filter **Add Address Element**.

Configure where to Insert the New Nodes

You can insert the new node into a location specified by an XPath expression or into a WS-Security header for the specified SOAP actor or role. Select one of the following options:

Insert using XPath:

Select or enter an XPath expression to specify where to insert the new node. If this expression returns more than one node, the first one is used. If the expression returns no nodes, the filter returns false. You can add, edit, or delete XPath expressions using the buttons provided.

Select one of the following options to determine where the new node is placed relative to the node(s) returned by the XPath expression.

- **Append:**
The new node is appended as a child node of the node returned by the XPath expression. If there are already child nodes of the node returned by the XPath expression, the new node is added as the last child node.
- **Before:**
The new node is inserted as a sibling node before the node returned by the XPath expression.
- **Replace:**
The node pointed to by the XPath expression is completely replaced by the new node.

Insert into WS-Security element for SOAP Actor/Role:

Select or enter the name of the SOAP actor/role that specifies the WS-Security element into which the XML node is inserted. A SOAP actor/role provides a way of distinguishing a particular WS-Security block from others that may be present in the message. Actors belong to the SOAP 1.1 specification, and were replaced by roles in SOAP 1.2. For example, this setting is useful if there is no SOAP header or WS-Security element in the message because these are created when this option is specified.

Node Source

Select one of the following options for the source of the new node:

- **Create a new node:**
If this option is selected, a new node is created and inserted into the location pointed to by the XPath expression configured above.
- **Insert previously removed nodes:**
You can configure a **Remove XML Node** filter to remove XML nodes from the message and store them in the de-

leted.node.list message attribute. You can then use the **Add XML Node** filter to re-insert these nodes back into a different location inside the message, effectively moving the deleted nodes in the message. To use this option, select the **Save deleted nodes** option on a **Remove XML Node** filter that is configured to run before the **Add XML Node** filter in the policy.

- **Message attribute:**
If this option is selected, a new node is created from the contents of the selected message attribute.

Configure New Node Details

Configure the following node details:

Node Type:

Select the type of the new node from the drop-down list. It is important to note that this selection determines where you can insert the new node according to the following rules:

- You can only append a node to a **Node Type** of `Element` or `Text`.
- If you append to a `Text` node, you must append another `Text` node.
- If you add a new `Attribute` node using the **Replace** option, it must replace an existing `Attribute` node.

Node Content:

This field contains the node to be inserted into the message. The **Node Content** field must contain valid XML when the **Node Type** is set to `Element`. You can also enter wildcards, which are populated at runtime by the Enterprise Gateway.

Attribute Node Details

The following attribute-related fields are only enabled when you select `Attribute` from the **Node Type** drop-down list:

Attribute Name:

Enter the name of the attribute in this field.

Attribute Namespace URL:

Enter the namespace of the attribute in this field.

Attribute Namespace Prefix:

Specify the prefix to use to map the element to the namespace entered above in this field. The new attribute is prefixed by this prefix.

Examples

The following are some examples of using the **Add XML Node** filter to replace and add attributes and elements.

Replacing an attribute value

To replace an attribute value, perform the following steps:

1. In the **Configure where to insert the new nodes** section, select **Insert using XPath**.
2. Select a value from the drop-down list (for example, SOAP Header "mustUnderstand" attribute).
3. Select the **Replace** option.
4. In the **Configure new node details** section, select `Attribute` from the **Node Type** list.
5. Enter the **Node Content** in the text box (for example, in this case, 1 or 0).
6. In the **Attribute Details** section, you must enter the **Attribute Name**.

Adding an attribute

To add an attribute to an element, perform the following steps:

1. In the **Configure where to insert the new nodes** section, select **Insert using XPath**.
2. Select a value from the drop-down list (for example, SOAP Header "mustUnderstand" attribute).
3. Select the **Append** option.
4. In the **Configure new node details** section, select `Attribute` from the a **Node Type** list.
5. Enter the **Node Content** in the text box (for example, in this case 1 or 0).
6. In the **Attribute Details** section, you must enter the **Attribute Name**.

Adding an element

To add an element, perform the following steps:

1. In the **Configure where to insert the new nodes** section, select **Insert using XPath**.
2. Select a value from the drop-down list (for example, SOAP Header Element).
3. Select the **Append** option.
4. In the **Configure new node details** section, select `Element` from the a **Node Type** list.
5. Enter the **Node Content** in the text box (for example, in this case, the contents of the SOAP header).

Replacing an element

To replace element A with new element B, perform the following steps:

1. In the **Configure where to insert the new nodes** section, select **Insert using XPath**.
2. Select a value from the drop-down list (for example, SOAP Method Element).
3. Select the **Replace** option.
4. In the **Configure new node details** section, select `Element` from the a **Node Type** list.
5. Enter the **Node Content** in the text box (for example, in this case, the contents of the SOAP method).

Contivo Transformation

Overview

The **Contivo Transformation** filter uses the Liaison Contivo engine to transform messages into an alternative format. First you use the Contivo Analyst tool to define the mappings required for your transformation. For example, a specific XML element is placed into a specific field in a COBOL message. In Contivo Analyst, when your mappings are defined, you automatically generate Java code, which must be placed on the Enterprise Gateway classpath. This topic explains how to add the generated Java code to the Enterprise Gateway classpath.

Configuration

Complete the following fields to configure the **Contivo Transformation** filter:

Name:

Enter an appropriate name for this filter.

Transform Class Name:

Contivo Analyst generates a Java class file that performs the transformation. You must place this class on the Enterprise Gateway classpath by copying the associated class file into the `INSTALL_DIR/ext/lib` directory, where `INSTALL_DIR` points to the root of your product installation. Enter the name of the class in this field.

Source FFD:

Contivo Analyst generates a Fixed File Descriptor (FFD) that describes the format of the *source* message to be transformed by the Contivo engine. Browse to the location of this file using the **Browse** button.

Target FFD:

Contivo Analyst also generates a Fixed File Descriptor (FFD) for the *target* message, which is the message generated by the Contivo engine. Browse to the location of this file using the **Browse** button.

Transformed Content Type:

Enter the Content-type of the target message used by the Enterprise Gateway when routing the output of the transformation on to the target Web Service.

Multipart Bodypart Conversion

Overview

The **Multipart Bodypart Conversion** filter is typically used to compress a MIME message before FTPing a message to an FTP server. A simple policy containing a **Multipart Bodypart Conversion** filter as a predecessor of an **FTP Upload** filter could be written to achieve this functionality.

Configuration

The following fields must be configured on this filter screen:

Name:

Enter a name for this filter in the field provided.

Multipart Content Type:

Specify the MIME content type that you want to compress the multipart message to. To compress a multipart message to a ZIP file, enter "multipart/x-zip" in this field.

Create REST Request

Overview

Representational State Transfer (REST) is a client-server architectural style used to represent the state of application resources in distributed systems. Typically, servers expose resources using a URI, and clients access these resources using HTTP verbs such as HTTP GET, HTTP POST, HTTP DELETE, and so on.

The **Create REST Request** filter enables you to create HTTP requests to RESTful Web Services. You can also configure the query string parameters that are sent with the REST request. For example, for an HTTP GET request, the parameters are URL-encoded and appended to the request URI as follows:

```
GET /translate_a/t?client=t&sl=en&tl=ga&text=Hello
```

For an HTTP POST request, the parameters are URL-encoded and added to the request body as follows:

```
POST /webservicetempconvert.aspx/CelsiusToFahrenheit
Host: ww.w3schools.com
Accept-charset: en
Celsius=200
```

This filter is found in the **Conversion** category in the Policy Studio. For details on how to extract REST request attributes from a message, see the [Extract REST Request Attributes](#) filter. For details on how to validate a REST request, see the [Validate REST Request](#) filter.

Configuration

Complete the following fields on the **Create REST Request** screen:

Name:

Enter an appropriate name for the filter.

HTTP Method:

Enter or select an HTTP method from the drop-down list (for example, POST, GET, DELETE, and so on).

REST Request Parameters:

You can add query string parameters to the REST request. These are simple name-value pairs (for example, Name=Joe Bloggs). To add query string parameters, click the **Add** button, and enter the name-value pair in the **Configure REST Request Parameters** dialog. Repeat to add multiple parameters.

Add attributes stored in attribute lookup list to REST request:

If you have populated the `attribute.lookup.list` message attribute using a previous filter in a policy circuit, you can select this checkbox to include these message attributes in the serialized query string that is written to the request.

Set HTTP Verb

Overview

You can use the **Set HTTP Verb** filter to explicitly set the HTTP verb in the message that is sent from the Enterprise Gateway. By default, all messages are routed onwards using the HTTP verb that the Enterprise Gateway received in the request from the client. If the message originated from a non-HTTP client (for example, JMS), the messages are routed using the HTTP POST verb.

Configuration

Complete the following fields on the **Set HTTP Verb** filter screen:

Name:

Enter a name for the filter.

HTTP Verb:

Specify the HTTP verb to use in the message that is routed onwards.

Insert MTOM Attachment

Overview

Message Transmission Optimization Mechanism (MTOM) provides a way to send binary data to Web Services in standard SOAP messages. MTOM leverages the include mechanism defined by XML Optimized Packaging (XOP) whereby binary data can be sent as a MIME attachment (similar to SOAP with Attachments) to a SOAP message. The binary data can then be referenced in the SOAP message using the `<xop:Include>` element.

The following MTOM message contains a binary image encapsulated in a MIME part. It is important to note that the MIME part that contains the binary image is referenced in the SOAP request body using the `<xop:Include>` element. The `href` attribute of this element refers to the `Content-ID` HTTP header of the MIME part.

```
POST /services/uploadImages HTTP/1.1
Host: SOAPbox
Content-Type: Multipart/Related;boundary=MIME_boundary;
    type="application/xop+xml";
    start="<mymessage.xml@example.org>";
    start-info="text/xml"

--MIME_boundary
Content-Type: application/xop+xml;
    charset=UTF-8;
    type="text/xml"
Content-Transfer-Encoding: 8bit
Content-ID: <mymessage.xml@example.org>

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <uploadGraphic xmlns="www.example.org">
      <image>
        <xop:Include xmlns:xop='http://www.w3.org/2004/08/xop/include'
          href='cid:http://example.org/myimage.gif' />
      </image>
    </uploadGraphic>
  </soap:Body>
</soap:Envelope>

--MIME_boundary
Content-Type: image/gif
Content-Transfer-Encoding: binary
Content-ID: <http://example.org/myimage.gif>

// binary octets for image

--MIME_boundary
```

When the Enterprise Gateway receives this request, the **Insert MTOM Attachment** filter can be used to read the binary data in the MIME parts pointed to by the `<xop:Include>` elements embedded in the SOAP request. The binary data is then Base64-encoded and inserted into the message in place of the `<xop:Include>` elements. The resulting message is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <uploadGraphic xmlns="www.example.org">
      <image>/aWKKapGGyQ=</image>
    </uploadGraphic>
  </soap:Body>
</soap:Envelope>
```

Configuration

Complete the following fields to configure this filter:

Name:

Enter a name for the filter.

XPath Location:

Use an XPath expression to point to the location of the `<xop:Include>` element that refers to the binary attachment. The specified XPath expression can point to multiple `<xop:Include>` elements if necessary. For example, an XPath expression of `//xop:Include` returns *all* `<xop:Include>` elements in the SOAP Envelope. For more information, see the [Configuring XPath Expressions](#) topic.

Remove attachments once they have been included in the message:

Select this option if you wish to remove the MIME parts that contain the actual binary content from the message after they have been inserted into the message.

Extract MTOM Attachment

Overview

Message Transmission Optimization Mechanism (MTOM) provides a way to send binary data to Web Services within standard SOAP messages. MTOM leverages the include mechanism defined by XML Optimized Packaging (XOP) whereby binary data can be sent as a MIME attachment (similar to SOAP with Attachments) to a SOAP message. The binary data can then be referenced in the SOAP message using the `<xop:Include>` element.

The following MTOM message contains a binary image that has been Base64-encoded so that it can be inserted as the contents of the `<image>` element:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <uploadGraphic xmlns="www.example.org">
      <image>/aWKKapGGyQ=</image>
    </uploadGraphic>
  </soap:Body>
</soap:Envelope>
```

When the Enterprise Gateway receives this request, the **Extract MTOM Content** filter can be used to extract the Base64-encoded content from the `<image>` element, replace it with an `<xop:Include>` element, which contains a reference to a newly created MIME part that contains the binary content. The following request shows the resulting MTOM message:

```
POST /services/uploadImages HTTP/1.1
Host: SOAPbox
Content-Type: Multipart/Related;boundary=MIME_boundary;
  type="application/xop+xml";
  start="<mymessage.xml@example.org>";
  start-info="text/xml"

--MIME_boundary
Content-Type: application/xop+xml;
  charset=UTF-8;
  type="text/xml"
Content-Transfer-Encoding: 8bit
Content-ID: <mymessage.xml@example.org>

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <uploadGraphic xmlns="www.example.org">
      <image>
        <xop:Include xmlns:xop='http://www.w3.org/2004/08/xop/include'
          href='cid:http://example.org/myimage.gif' />
      </image>
    </uploadGraphic>
  </soap:Body>
</soap:Envelope>
```

```
--MIME_boundary
Content-Type: image/gif
Content-Transfer-Encoding: binary
Content-ID: <http://example.org/myimage.gif>

// binary octets for image

--MIME_boundary
```

It is important to note the following points:

- The Base64-encoded contents of the `<image>` element have been replaced by the `<xop:Include>` element.
- The `<xop:Include>` element points to a MIME part using the `href` attribute.
- The value of the `href` attribute corresponds to the value of the `Content-ID` HTTP header of the MIME part that contains the binary octets of the actual image file.

Configuration

Complete the following fields to configure this filter:

Name:

Enter an appropriate name for the filter.

XPath Location:

Use an XPath expression to locate the encoded data elements. For example, in the sample SOAP request message above, you would configure an XPath expression to point to the `<image>` element. For more information, see the [Configuring XPath Expressions](#) topic.

Load File

Overview

The **Load file** filter enables you to load the contents of the specified file, and set them as message content to be processed. When the contents of the file are loaded, they can be passed to the core message pipeline for processing by the appropriate message filters. For example, the **Load file** filter might be used in cases where an external application drops XML files on to the file system to be validated, modified, and potentially routed on over HTTP, JMS, or simply stored to a directory where the application can access them again.

For example, this sort of protocol mediation can be useful when integrating legacy systems. Instead of making drastic changes to the legacy system by adding an HTTP engine, the Enterprise Gateway can load files from the file system, and route them on over HTTP to another back-end system. The added benefit is that messages are exposed to the full compliment of message processing filters available in the Enterprise Gateway. This ensures that only properly validated messages are routed on to the target system.

Configuration

To configure the **Load file** filter, specify the following fields:

Name	Name of the filter to be displayed in a circuit. Defaults to Load file .
File	Specify the path to the file that the content is loaded from (for example, <code>c:/workspace/example_file.xml</code>).
Output directory	Specify the directory to write the output to at the end of processing (for example, <code>c:/my_app/output_dir</code>).
File contents	<p>Choose one of the following options:</p> <ul style="list-style-type: none">• Raw File• HTTP Message including HTTP Headers <p>This enables you to load the contents of a raw file, or to load archived HTTP messages from disk.</p>
Finishing processing	<p>Specify the action to take when processing is complete:</p> <ul style="list-style-type: none">• Do nothing• Delete loaded file• Move loaded file to following directory <p>If you select Move loaded file to following directory, the Directory field is enabled to allow you to specify the directory that the file is moved to. The default is Do nothing.</p>

Remove Attachments

Overview

This filter can be used to remove **all** attachments from either a request or a response message, depending on where the filter is placed in the policy.

Configuration

Simply enter a name for this filter in the **Name** field.

Remove HTTP Header

Overview

The Enterprise Gateway can strip a named HTTP header from the message as it passes through a policy. This is especially useful in cases where end-user credentials are passed to the Enterprise Gateway in an HTTP header. After processing the credentials, you can use the **Remove HTTP Header** filter to strip the header from the message to ensure that it is not forwarded on to the destination Web Service.

Configuration

To configure the **Remove HTTP Header** filter, perform the following steps:

1. Enter an appropriate name for this filter in the **Name** field.
2. Specify name of the HTTP header to remove in the **HTTP Header Name** field.
3. Select the **Fail if header is not present** checkbox to configure the Enterprise Gateway to abort the filter if the message does not contain the named HTTP header.

Remove XML Node

Overview

You can use this filter to remove an XML element, attribute, text, or comment node from an XML message. You can specify the node to remove using an XPath expression. The XPath query language enables you to select nodes in an XML document.

Configuration

To configure this filter, specify the following fields:

Name:

Enter a suitable name that reflects the role of the filter. For example, if the purpose of this filter is to remove an <ID> element from the message, it would be appropriate to name this filter **Remove ID Element**.

XPath Location:

Specify an XPath expression to indicate the node to remove. When the expression is configured correctly, you can remove an element, attribute, text, or comment node. If this expression returns more than one node, all returned nodes are removed.

You can select XPath expressions from the drop-down list, and edit or add expressions by clicking the relevant button. The following are some example expressions:

<i>Name</i>	<i>XPath Expression</i>	<i>Prefix</i>	<i>URI</i>
The First WSSE Security element	//wsse:Security[1]	wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
Text Nodes in SOAP Body	/soap:Envelope/soap:Body/text()	soap	http://schemas.xmlsoap.org/soap/envelope/

Fail if no nodes returned from XPath:

If this option is selected, and the XPath expression returns no nodes, the filter returns false. If this option is *not* selected, and the XPath returns no nodes, the filter returns true, and no nodes are removed.

Save deleted nodes to be re-inserted to new location:

You can use this option in cases where you want to move XML nodes from one location in the message to another. By selecting this option, the deleted nodes are stored in the `deleted.node.list` message attribute. You can then use the **Add XML Node** filter to insert the deleted nodes back into a different location in the message. For more details, see the [Add XML Node](#) filter.

Set Message

Overview

The **Set Message** filter replaces the body of the message. The replacement data can be plain text, HTML, XML, or any other text-based markup.

You can use properties representing the values of message attributes in the replacement text to insert message-specific data into the message body. For example, you can insert the authenticated user's ID into a `<Username>` element by using the `${authentication.subject.id}` property as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Username>${authentication.subject.id}</Username>
  </soap:Header>
  <soap:Body>
    <getQuote xmlns="oracle.com">
      <ticker>ORM.L</ticker>
    </getQuote>
  </soap:Body>
</soap:Envelope>
```

Assuming the `oracle` user authenticated successfully to the Enterprise Gateway, the message body is set to the following contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Username>oracle</Username>
  </soap:Header>
  <soap:Body>
    <getQuote xmlns="oracle.com">
      <ticker>ORM.L</ticker>
    </getQuote>
  </soap:Body>
</soap:Envelope>
```

You can also use the **Set Message** filter to customize SOAP faults that are returned to clients in the case of a failure or exception in the policy.

Configuration

Perform the following steps to configure the **Set Message** filter:

1. Enter a name for this filter in the **Name** field.
2. Specify the content type of the new message body in the **Content-type** field. For example, if the new message body is HTML markup, enter `text/html` in the **Content-Type** field.
3. Enter the new message body in the **Message Body** text area. You can use properties, as shown in the example above, to ensure that current message attribute values are inserted into the message body at the appropriate places. You can also load the message contents from a file by browsing to the location of the file using the **Browse** button.

XSLT Transformation

Overview

Extensible Stylesheet Language Transformations (XSLT) is a declarative, XML-based language used to transform XML documents into other XML documents. An XSL stylesheet is used to transform an XML document into another document type. The stylesheet defines how elements in the XML source document should appear in the resulting XML document.

The Enterprise Gateway can convert XML data to other data formats using XSL files. For example, an incoming XML message adhering to a specific XML schema can be converted to an XML message adhering to a different schema before it is sent to the destination Web Service.

This type of conversion is especially valuable in the Web Services arena, where a Web Service might receive SOAP requests from various types of clients, such as browsers, applications, and mobile phones. Each client might send up a different type of SOAP request to the Web Service. Using stylesheets, the Enterprise Gateway can then convert each type of request to the same format. The requests can then be processed in the same fashion.

Enter an appropriate name for this filter in the **Name** field, and configure the following tabs.

Stylesheet Location

Select an XSL stylesheet from the **Stylesheet Location** drop-down list, which is populated with the contents of the Stylesheet Library. You can import a new stylesheet into the library by clicking the **View/Import** button, and the **Add** button on the **Stylesheet Library** dialog.

You can modify existing stylesheets in the **XSLT Contents** text area, and then update them in the Enterprise Gateway configuration by clicking the **Update** button.

Stylesheet Parameters

You can pass parameters to an XSL stylesheet using specified values in `<xsl:param>` elements. These values are then used in the templates defined throughout the stylesheet.

Using the **XSLT Transformation** filter, you can pass the values of message attributes to the configured stylesheet. For example, you can take the value of the `authentication.subject.id` message attribute, pass it to the configured XSL stylesheet, and then output this value to the result produced by the conversion.

To use this feature, select the **Use Message Attributes as Stylesheet Parameters** checkbox, and specify the message attribute to pass to the stylesheet by clicking the **Add** button.

The following example from an XSL stylesheet that uses parameters shows how to configure this:

```
<xsl:param name="authentication.subject.id"/>
<xsl:param name="authentication.issuer.id"/>
```

To pass the corresponding message attribute values to the stylesheet, you must add the `authentication.subject.id` and `authentication.issuer.id` message attributes to the **Message Attributes to use** table.

Important Note:

The name of the specified parameter must be a valid Enterprise Gateway message attribute name, and there *must* be an

equivalent parameter name in the stylesheet.

Advanced

Complete the following fields on this tab:

Provider Class Name:

Enter the fully qualified name of the XSLT provider class of the XSLT library that you want to use. This class *must* be added to the Enterprise Gateway's classpath. If this field is blank, the default provider is used.

The simplest way to add a provider class to the Enterprise Gateway's classpath is to drop the required JAR file into the `PRODUCT_HOME/ext/lib` directory, where `PRODUCT_HOME` refers to the root of your Enterprise Gateway installation.

Result will be XML:

You can convert an incoming XML message to other data formats. Select this option if the result of the XSLT conversion is always XML. If not, the content-type of the result document depends on the output method of the XSLT stylesheet. For example, if the stylesheet specifies an output method of HTML (`<xsl:output method="html">`), this field should be left blank so that the Enterprise Gateway can forward on the HTML output document to the target Web Service.

XML-Encryption Settings

Overview

The Enterprise Gateway can XML encrypt an XML message so that only certain specified recipients can decrypt the message. XML Encryption is a W3C standard that enables data to be encrypted and decrypted at the application layer of the OSI stack, thus ensuring complete end-to-end confidentiality of data.

The **XML-Encryption Settings** should be used in conjunction with the **XML-Encryption** filter, which performs the encryption. The **XML-Encryption Settings** generates the `encryption.properties` message attribute, which is required by the **XML-Encryption** filter.

XML Encryption Overview

XML Encryption facilitates the secure transmission of XML documents between two application endpoints. Whereas traditional transport-level encryption schemes, such as SSL and TLS, can only offer point-to-point security, XML Encryption guarantees complete end-to-end security. Encryption takes place at the application-layer, and so the encrypted data can be encapsulated in the message itself. The encrypted data can therefore remain encrypted as it travels along its path to the target Web Service.

Before explaining how to configure the Enterprise Gateway to encrypt XML messages, it is useful to examine an XML encrypted message. The following example shows a SOAP message containing information about Oracle:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <getCompanyInfo xmlns="http://www.oracle.com">
      <name>Company</name>
      <description>XML Security Company</description>
    </getCompanyInfo>
  </s:Body>
</s:Envelope>
```

After encrypting the SOAP Body, the message is as follows:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Security xmlns="http://schemas.xmlsoap.org/ws/2003/06/secext" s:actor="Enc">
      <!-- Encapsulates the recipient's key details -->
      <enc:EncryptedKey xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
        Id="00004190E5D1-7529AA14" MimeType="text/xml">
        <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5">
          <enc:KeySize>256</enc:KeySize>
        </enc:EncryptionMethod>
        <enc:CipherData>
          <!-- The session key encrypted with the recipient's public key -->
          <enc:CipherValue>
            AAAAAJ/lK ... mrTF8Egg==
          </enc:CipherValue>
        </enc:CipherData>
      <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
```



```

    <dsig:KeyName>sample</dsig:KeyName>
  <dsig:X509Data>
    <!-- The recipient's X.509 certificate -->
    <dsig:X509Certificate>
      MIEZzCCA0 ... fzmc/YR5gA
    </dsig:X509Certificate>
  </dsig:X509Data>
</dsig:KeyInfo>
<enc:CarriedKeyName>Session key</enc:CarriedKeyName>
<enc:ReferenceList>
  <enc:DataReference URI="#00004190E5D1-5F889C11"/>
</enc:ReferenceList>
</enc:EncryptedKey>
</Security>
</s:Header>
<enc:EncryptedData xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
  Id="00004190E5D1-5F889C11" MimeType="text/xml"
  Type="http://www.w3.org/2001/04/xmlenc#Element">
  <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc">
    <enc:KeySize>256</enc:KeySize>
  </enc:EncryptionMethod>
  <enc:CipherData>
    <!-- The SOAP Body encrypted with the session key -->
    <enc:CipherValue>
      E2ioF8ib2r ... KJAnrX0GQV
    </enc:CipherValue>
  </enc:CipherData>
  <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
    <dsig:KeyName>Session key</dsig:KeyName>
  </dsig:KeyInfo>
</enc:EncryptedData>
</s:Envelope>

```

The most important elements are as follows:

- **EncryptedKey:**
The `EncryptedKey` element encapsulates all information relevant to the encryption key.
- **EncryptionMethod:**
The `Algorithm` attribute specifies the algorithm used to encrypt the data. The message data (`EncryptedData`) is encrypted using the Advanced Encryption Standard (AES) *symmetric cipher*, but the session key (`EncryptedKey`) is encrypted with the RSA *asymmetric* algorithm.
- **CipherValue:**
The value of the encrypted data. The contents of the `CipherValue` element are always Base64 encoded.
- **DigestValue:**
Contains the Base64-encoded message-digest.
- **KeyInfo:**
Contains information about the recipient and his encryption key, such as the key name, X.509 certificate, and Common Name.
- **ReferenceList:** This element contains a list of references to encrypted elements in the message. The `ReferenceList` contains a `DataReference` element for each encrypted element, where the value of a `URI` attribute points to the `Id` of the encrypted element. In the previous example, you can see that the `DataReference` `URI` attribute contains the value `#00004190E5D1-5F889C11`, which corresponds with the `Id` of the `EncryptedData` element.
- **EncryptedData:**

The XML elements or content that has been encrypted. In this case, the SOAP `Body` element has been encrypted, and so the `EncryptedData` block has replaced the SOAP `Body` element.

Now that you have seen how encrypted data can be encapsulated in an XML message, it is important to discuss how the data is encrypted. When a message is encrypted, it is encrypted in such a manner that only the intended recipients of the message can decrypt it. By encrypting the message with the recipient public key, the sender can be guaranteed that only the intended recipient can decrypt the message using his private key, to which he has sole access. This is the basic principle behind *asymmetric cryptography*.

In practice, however, encrypting and decrypting data with a public-private key pair is a notoriously CPU-intensive and time consuming affair. Because of this, asymmetric cryptography is seldom used to encrypt large amounts of data. The following steps show a more typical encryption process:

1. The sender generates a one-time *symmetric* (or session) key which is used to encrypt the data. Symmetric key encryption is much faster than asymmetric encryption, and is far more efficient with large amounts of data.
2. The sender encrypts the data with the symmetric key. This same key can then be used to decrypt the data. It is therefore crucial that only the intended recipient can access the symmetric key and consequently decrypt the data.
3. To make sure that nobody else can decrypt the data, the symmetric key is encrypted with the recipient's *public key*.
4. The data (encrypted with the symmetric key), and session key (encrypted with the recipient's public key), are then sent together to the intended recipient.
5. When the recipient receives the message, he decrypts the encrypted session key using his *private key*. Because the recipient is the only one with access to the private key, he is the only one who can decrypt the encrypted session key.
6. Armed with the decrypted session key, the recipient can decrypt the encrypted data into its original plaintext form.

Now that you understand the structure and mechanics of XML Encryption, you can configure the Enterprise Gateway to encrypt egress XML messages. The following section describes how to configure the tabs on the **XML Encryption Settings** screen.

Encryption Key

The settings on this tab determine the key to use to encrypt the message, and how this key is referred to in the encrypted data. It is important to note that a symmetric key is used to encrypt the data. This symmetric key is then encrypted (asymmetrically) with the recipient's public key. In this way, only the recipient can decrypt the symmetric encryption key with its private key. When the recipient has access to the unencrypted encryption key, it can decrypt the data. The following configuration options are available:

Generate Encryption Key:

Select this option to generate a symmetric key to encrypt the data with.

Encryption Key in Message Attribute:

If you have already used a symmetric key in a previous filter (for example, a **Sign Message** filter), you can reuse that key to encrypt data by selecting this option and specifying the `symmetric.key` message attribute.

Include Encryption Key in Message:

Select this option if you want to include the encryption key in the message. The encryption key is encrypted for the recipient so that only the recipient can access the encryption key. You may choose not to include the symmetric key in the message if the Enterprise Gateway and recipient have agreed on the symmetric encryption key using some other means.

Specify Method of Associating the Encryption Key with the Encrypted Data:

This section enables you to configure the method by which the encrypted data references the key used to encrypt it. The following options are available:

- **Point to Encryption Key with Security Token Reference:**

This option creates a `<SecurityTokenReference>` in the `<EncryptedData>` that points to an `<EncryptedKey>`.

- **Embed Symmetric Key Inside Encrypted Data:**
Place the `<xenc:EncryptedKey>` inside the `<xenc:EncryptedData>` element.
- **Specify Encryption Key via Carried Keyname:**
Place the encrypted key's carried keyname inside the `<dsig:KeyInfo>/<dsig:KeyName>` of the `<xenc:EncryptedData>`.
- **Specify Encryption Key via Retrieval Method:**
Refer to a symmetric key via a retrieval method reference from the `<xenc:EncryptedData>`.
- **Symmetric Key Refers to Encrypted Data:**
The symmetric key refers to `<xenc:EncryptedData>` using a reference list.

Use Derived Key:

Select this option if you want to derive a key from the symmetric key configured above to encrypt the data. The `<enc:EncryptedData>` has a `<wsse:SecurityTokenReference>` to the `<wssc:DerivedKeyToken>`. The `<wssc:DerivedKeyToken>` refers to the `<enc:EncryptedKey>`. Both `<wssc:DerivedKeyToken>` and `<enc:EncryptedKey>` are placed inside a `<wsse:Security>` element.

Key Info

This tab configures the content of the `<KeyInfo>` section of the generated `<EncryptedData>` block. Configure the following fields on this tab:

Do Not Include KeyInfo Section:

This option enables you to omit all information about the certificate that contains the public key that was used to encrypt the data from the `<EncryptedData>` block. In other words, the `<KeyInfo>` element is omitted from the `<EncryptedData>` block. This is useful where a downstream Web Service uses an alternative method to decide what key to use to decrypt the message. In such cases, adding certificate information to the message may be regarded as an unnecessary overhead.

Include Certificate:

This is the default option, which places the certificate that contains the encryption key inside the `<EncryptedData>`. The following example, shows an example of a `<KeyInfo>` that has been produced using this option:

```
<enc:EncryptedData xmlns:enc="http://www.w3.org/2001/04/xmenc#">
  <dsig:KeyInfo>
    <dsig:X509Data>
      <dsig:X509SubjectName>CN=Sample...</dsig:X509SubjectName>
      <dsig:X509Certificate>
        MII EZDCCA0yg
        ....
        RNp9aKD1fEQgJ
      </dsig:X509Certificate>
    </dsig:X509Data>
  </dsig:KeyInfo>
</enc:EncryptedData>
```

Expand Public Key:

The details of the public key used to encrypt the data are inserted into a `<KeyValue>` block. The `<KeyValue>` block is only inserted when this option is selected.

```

<enc:EncryptedData xmlns:enc="http://www.w3.org/2001/04/xmlenc#">
  ...
  <dsig:KeyInfo>
    <dsig:X509Data>
      <dsig:X509SubjectName>CN=Sample...</dsig:X509SubjectName>
      <dsig:X509Certificate>
        MIIIE ..... EQgJ
      </dsig:X509Certificate>
    </dsig:X509Data>
    <dsig:KeyValue>
      <dsig:RSAKeyValue>
        <dsig:Modulus>
          AMfb2tT53GmMiD
          ...
          NmrNht7iy18=
        </dsig:Modulus>
        <dsig:Exponent>AQAB</dsig:Exponent>
      </dsig:RSAKeyValue>
    </dsig:KeyValue>
  </dsig:KeyInfo>
</enc:EncryptedData>

```

Include Distinguished Name:

If this checkbox is selected, the Distinguished Name of the certificate that contains the public key used to encrypt the data is inserted in an <X509SubjectName> element as shown in the following example:

```

<enc:EncryptedData xmlns:enc="http://www.w3.org/2001/04/xmlenc#">
  ...
  <dsig:KeyInfo>
    <dsig:X509Data>
      <dsig:X509SubjectName>CN=Sample,C=IE...</dsig:X509SubjectName>
      <dsig:X509Certificate>
        MIIEZDCCA0yg
        ...
        RNP9aKD1fEQgJ
      </dsig:X509Certificate>
    </dsig:X509Data>
  </dsig:KeyInfo>
</enc:EncryptedData>

```

Include Key Name:

This option enables you insert a key identifier, or <KeyName>, to allow the recipient to identify the key to use to decrypt the data. Enter an appropriate value for the <KeyName> in the **Value** field. Typical values include Distinguished Names (DName) from X.509 certificates, key IDs, or email addresses. Specify whether the specified value is a **Text value** of a **Distinguished name attribute** by selecting the appropriate radio button.

```

<enc:EncryptedData xmlns:enc="http://www.w3.org/2001/04/xmlenc#">

```

```

...
<dsig:KeyInfo>
  <dsig:KeyName>test@oracle.com</dsig:KeyName>
</dsig:KeyInfo>
</enc:EncryptedData>

```

Put Certificate in an Attachment:

The Enterprise Gateway supports SOAP messages with attachments. By selecting this option, you can save the certificate containing the encryption key to the file specified in the input field. This file can then be sent along with the SOAP message as a SOAP attachment.

From previous examples, it is clear that the user's certificate is usually placed inside a `<KeyInfo>` element. However, in this example, the certificate is contained in an attachment, and not in the `<EncryptedData>`. Clearly, you need a way to reference the certificate from the `<EncryptedData>` block, so that the recipient can determine what key it should use to decrypt the data. This is the role of the `<SecurityTokenReference>` block.

The `<SecurityTokenReference>` block provides a generic mechanism for applications to retrieve security tokens in cases where these tokens are not contained in the SOAP message. The name of the security token is specified in the `URI` attribute of the `<Reference>` element.

```

<enc:EncryptedData xmlns:enc="http://www.w3.org/2001/04/xmlenc#">
  ...
  <dsig:KeyInfo>
    <wsse:SecurityTokenReference xmlns:wsse="http://schemas.xmlsoap.org/ws/...">
      <wsse:Reference URI="c:\myCertificate.txt"/>
    </wsse:SecurityTokenReference>
  </dsig:KeyInfo>
</enc:EncryptedData>

```

When the message is sent, the certificate attachment is given a `Content-Id` corresponding to the `URI` attribute of the `<Reference>` element. The following example shows the wire format of the complete multipart MIME SOAP message. It should help illustrate how the `<Reference>` element refers to the `Content-ID` of the attachment:

```

POST /adoWebSvc.asmx HTTP/1.0
Content-Length: 3790
User-Agent: Enterprise Gateway
Accept-Language: en
Content-Type: multipart/related; type="text/xml";
              boundary="-----Multipart-SOAP-boundary"

-----Multipart-SOAP-boundary
Content-Id: soap-envelope
Content-Type: text/xml; charset="utf-8";
SOAPAction=getQuote

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  ...

```

```

<enc:EncryptedData xmlns:enc="http://www.w3.org/2001/04/xmlenc#">
  ...
  <dsig:KeyInfo>
    <ws:SecurityTokenReference xmlns:ws="http://schemas.xmlsoap.org/ws/...">
      <ws:Reference URI="c:\myCertificate.txt"/>
    </ws:SecurityTokenReference>
  </dsig:KeyInfo>
</enc:EncryptedData>
  ...
</s:Envelope>

-----=Multipart-SOAP-boundary
Content-Id: c:\myCertificate.txt
Content-Type: text/plain; charset="US-ASCII"

MIIEZDCCA0ygAwIBAgIBAzANBgkqhki
....
7uFveG0eL0zBwZ5qwLRNp9aKD1fEQgJ
-----=Multipart-SOAP-boundary-

```

Security Token Reference:

A `<wsse:SecurityTokenReference>` element can be used to point to the security token used to encrypt the data. If you wish to use a `<wsse:SecurityTokenReference>`, enable this option, and select a Security Token Reference type from **Reference Type** drop-down list.

The `<wsse:SecurityTokenReference>`, (in the `<dsig:KeyInfo>`), may contain a `<wsse:Embedded>` security token. Alternatively, the `<wsse:SecurityTokenReference>`, (in the `<dsig:KeyInfo>`), may refer to a certificate using a `<dsig:X509Data>`. Select the appropriate button, **Embed** or **Refer**, depending on whether you want to use an embedded security token or a referred one.

If you have configured the `SecurityContextToken (sct)` mechanism from the **Security Token Reference** drop-down list, you can select to use an **Attached SCT** or an **Unattached SCT**. The default option is to use an **Attached SCT**, which should be used in cases where the SCT refers to a security token inside the `<wsse:Security>` header. If the SCT is located outside the `<wsse:Security>` header, you should select the **Unattached SCT** option.

You can make sure to include a `<BinarySecurityToken>` (BST) that contains the certificate (that contains the encryption key) in the message by selecting the **Include BinarySecurityToken** option. The BST is inserted into the WS-Security header regardless of the type of Security Token Reference selected from the dropdown.

Select the **Include TokenType** checkbox if you want to add the `TokenType` attribute to the `SecurityTokenReference` element

Important Note:

When using the Kerberos Token Profile standard, and the Enterprise Gateway is acting as the initiator of a secure transaction, it can use Kerberos session keys to encrypt a message. The `KeyInfo` must be configured to use a Security Token Reference with a `ValueType` of `GSS_Kerberosv5_AP_REQ`. In this case, the Kerberos token is contained in a `<BinarySecurityToken>` in the message.

If the Enterprise Gateway is acting as the recipient of a secure transaction, it can also use the Kerberos session keys to encrypt the message returned to the client. However, in this case, the `KeyInfo` must be configured to use a Security Token Reference with `ValueType` of `Kerberosv5_APREQSHA1`. When this is selected, the Kerberos token is not contained in the message. The Security Token Reference contains a SHA1 digest of the original Kerberos token received from the client, which identifies the session keys to the client.

When using the WS-Trust for SPENGO standard, the Kerberos session keys are not used directly to encrypt messages because a security context with an associated symmetric key is negotiated. This symmetric key is shared by both client

and service and can be used to encrypt messages on both sides.

Recipients

XML Messages can be encrypted for multiple recipients. In such cases, the symmetric encryption key is encrypted with the public key of each intended recipient and added to the message. Each recipient can then decrypt the encryption key with their private key and use it to decrypt the message.

The following SOAP message has been encrypted for 2 recipients (*oracle_1* and *oracle_2*). The encryption key has been encrypted twice: once for *oracle_1* using its public key, and a second time for *oracle_2* using its public key. It is important to note that the data itself is only encrypted once, while the encryption key must be encrypted for each recipient. For illustration purposes, only those elements relevant to the above discussion have been included in the following XML encrypted message.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Security xmlns="http://schemas.xmlsoap.org/ws/2003/06/secext"
      s:actor="Enc Keys">
      <enc:EncryptedKey xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
        Id="0000418BBB61-A692675C" MimeType="text/xml">
        ...
        <enc:CipherData>
          <!-- Enc key encrypted with oracle_1's public key and base64-encoded -->
          <enc:CipherValue>AAAAAExx1A ... vuAhCgMQ==</enc:CipherValue>
        </enc:CipherData>
        <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
          <dsig:KeyName>oracle_1</dsig:KeyName>
        </dsig:KeyInfo>
        <enc:CarriedKeyName>Session key</enc:CarriedKeyName>
        <enc:ReferenceList>
        <enc:DataReference URI="#0000418BBB61-D4495D9B"/>
        </enc:ReferenceList>
        </enc:EncryptedKey>
        <enc:EncryptedKey xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
          Id="#0000418BBB61-D4495D9B" MimeType="text/xml">
          ...
          <enc:CipherData>
            <!-- Enc key encrypted with oracle_2's public key and base64-encoded -->
            <enc:CipherValue>AAAAABZH+U ... MrMEEM/Ps=</enc:CipherValue>
          </enc:CipherData>
          <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
            <dsig:KeyName>oracle_2</dsig:KeyName>
          </dsig:KeyInfo>
          <enc:CarriedKeyName>Session key</enc:CarriedKeyName>
          <enc:ReferenceList>
          <enc:DataReference URI="#0000418BBB61-D4495D9B"/>
          </enc:ReferenceList>
          </enc:EncryptedKey>
        </Security>
      </s:Header>
      <enc:EncryptedData xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
        Id="0000418BBB61-D4495D9B" MimeType="text/xml"
        Type="http://www.w3.org/2001/04/xmlenc#Element">
        <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc">
          <enc:KeySize>256</enc:KeySize>
        </enc:EncryptionMethod>
        <enc:CipherData>
          <!-- SOAP Body encrypted with symmetric enc key and base64-encoded -->
          <enc:CipherValue>WD0TmuMk9 ... GzYFeq8SM=</enc:CipherValue>
        </enc:CipherData>
```

```

<dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
  <dsig:KeyName>Session key</dsig:KeyName>
</dsig:KeyInfo>
</enc:EncryptedData>
</s:Envelope>

```

There are two `<EncryptedKey>` elements, one for each recipient. The `<CipherValue>` element contains the symmetric encryption key encrypted with the recipient's public key. The encrypted symmetric key must be Base64-encoded so that it can be represented as the textual contents of an XML element.

The `<EncryptedData>` element contains the encrypted data, along with information about the encryption process, including the encryption algorithm used, the size of the encryption key, and the type of data that was encrypted (for example, whether an element or the contents of an element was encrypted).

Click the **Add** button to add a new recipient for which the data will be encrypted. Configure the following fields on the **XML Encryption Recipient** dialog:

Recipient Name:

Enter a name for the recipient in this field. The name entered can then be selected on the main **Recipients** tab of the filter.

Actor:

The `<EncryptedKey>` for this recipient is inserted into the specified SOAP actor/role.

Use Key in Message Attribute:

Specify the message attribute that contains the recipient's public key that is used to encrypt the data. By default, the `certificate` attribute is used. Typically, this attribute is populated by the **Find Certificate** filter, which retrieves a certificate from any one of a number of locations, including the Trusted Certificate Store, an LDAP directory, HTTP header, or from the message itself.

If you want to encrypt the message for multiple recipients, you must configure multiple **Find Certificate** filters (or some other filter that can retrieve certificates). Each **Find Certificate** filter retrieves a certificate for a single recipient and store it in a unique message attribute.

For example, a **Find Certificate** filter called **Find Certificate for Recipient1** filter could locate Recipient1's certificate from the Certificate Store and store it in a `certificate_recip1` message attribute. You would then configure a second **Find Certificate** filter called **Find Certificate for Recipient2**, which could retrieve Recipient2's certificate from the Certificate Store and store it in a `certificate_recip2` message attribute.

On the **Recipients** tab of the **XML Encryption Settings** filter, you would then configure two recipients. For the first recipient (Recipient1), you would enter `certificate_recip1` as the location of the encryption key, while for the second recipient (Recipient2), you would specify `certificate_recip2` as the location of the encryption key.

It is important to note that if the Enterprise Gateway fails to encrypt the message for any of the recipients configured on the **Recipients** tab, the filter will fail.

What to Encrypt

This tab is used to identify parts of the message that must be encrypted. Each encrypted part will be replaced by an `<EncryptedData>` block, which contains all information required to decrypt the block.

You can use *any* combination of **Node Locations**, **XPaths**, and the nodes contained in a **Message Attribute** to specify the nodes that are required to be encrypted. Please refer to the [Locate XML Nodes](#) filter for more information on how to use these node selectors.

Important Note:

Note the difference between encrypting the element and encrypting the element content. When encrypting the element, the entire element is replaced by the `<EncryptedData>` block. This is not recommended, for example, if you wish to encrypt the SOAP Body because if this element is removed from the SOAP message, the message may no longer be considered a valid SOAP message.

Element encryption is more suitable when encrypting security blocks, (for example, WS-Security Username tokens and SAML assertions) that may appear in a WS-Security header of a SOAP message. In such cases, replacing the element content (for example, a `<UsernameToken>` element) with an `<EncryptedData>` block will not affect the semantics of the WS-Security header.

If you wish to encrypt the SOAP Body, you should use element content encryption, where the children of the element are replaced by the `<EncryptedData>` block. In this way, the message can still be validated against the SOAP schema.

When using **Node Locations** to identify nodes that are to be encrypted, you can configure whether to encrypt the element or the element contents on the **Locate XML Nodes** dialog. To encrypt the element, select the **Encrypt Node** radio button. Alternatively, to encrypt the element contents, select the **Encrypt Node Content** radio button.

If you are using XPath expressions to specify the nodes that are to be signed, be careful not to use an expression that returns a node and all its contents. The **Encrypt Node** and **Encrypt Node Content** options are also available when configuring XPath expressions on the **Enter XPath Expression** dialog.

Advanced

The **Advanced** tab on the main **XML-Encryption Settings** screen enables you to configure some of the more complicated settings regarding XML-Encryption. The following settings are available:

Algorithm Suite Tab:

The following fields can be configured on this tab:

Algorithm Suite:

WS-Security Policy defines a number of *algorithm suites* that group together a number of cryptographic algorithms. For example, a given algorithm suite uses specific algorithms for asymmetric encryption, symmetric encryption, asymmetric key wrap, and so on. Therefore, by specifying an algorithm suite, you are effectively selecting a whole suite of cryptographic algorithms to use.

If you want to use a particular WS-Security Policy algorithm suite, you can select it here. The **Encryption Algorithm** and **Key Wrap Algorithm** fields are automatically populated with the corresponding algorithms for that suite.

Encryption Algorithm:

The encryption algorithm selected is used to encrypt the data. The following algorithms are available:

- AES-256
- AES-192
- AES-128
- Triple DES

Key Wrap Algorithm:

The key wrap algorithm selected here is used to wrap (encrypt) the symmetric encryption key with the recipient's public key. The following key wrap algorithms are available:

- KwRsa15
- KwRsaOaep

Settings Tab:

The following advanced settings are available on this tab:

Generate a Reference List in WS-Security Block:

When this option is selected, a `<xenc:ReferenceList>` that holds a reference to all encrypted data elements is generated. The `<xenc:ReferenceList>` element is inserted into the WS-Security block indicated by the specified actor.

Insert Reference List into EncryptedKey:

When this option is selected, a `<xenc:ReferenceList>` that holds a reference to all encrypted data elements is generated. The `<xenc:ReferenceList>` element is inserted into the `<xenc:EncryptedKey>` element.

Layout Type:

Select the WS-Security Policy layout type that you want the generated tokens to adhere to. This includes elements such as the `<EncryptedData>`, `<EncryptedKey>`, `<ReferenceList>`, `<BinarySecurityToken>`, and `<DerivedKeyToken>` tokens, among others.

Fail if no Nodes to Encrypt:

Select this option if you want the filter to fail if any of the nodes specified on the **What to Encrypt** tab are found in the message.

Insert Timestamp:

This option enables you to insert a WS-Security Timestamp as an encryption property.

Indent:

This option enables you to format the inserted `<EncryptedData>` and `<EncryptedKey>` blocks by indenting the elements.

Insert CarriedKeyName for EncryptedKey:

Select this option to insert a `<CarriedKeyName>` element into the generated `<EncryptedKey>` block.

Auto-generation using the XML Encryption Settings Wizard

Because the **XML Encryption Settings** filter must always be used in conjunction with the **XML Encryption** and **Find Certificate** filters, the Policy Studio provides a wizard that can generate these three filters at the same time. Right-click a policy node on the **Policies** tab in the Policy Studio, and select **XML Encryption Settings**.

For more information on how to configure the **XML Encryption Settings Wizard** see the [XML Encryption Settings Wizard](#) topic.

XML Encryption Wizard

Overview

There are several filters involved in encrypting a message using XML Encryption. These filters are as follows:

<i>Filter</i>	<i>Role</i>
Find Certificate	Specify the certificate that contains the public key to use in the encryption. The data will be encrypted such that it can only be decrypted with the corresponding private key.
XML Encryption Settings	Specify the recipient of the encrypted data, what data to encrypt, what algorithms to use, and other such options that will affect the way the data is encrypted.
XML Encryption	Performs the actual encryption using the certificate selected in the Find Certificate filter and the options set in the XML Encryption Settings filter.

While these filters can be configured independently of each other, it makes sense to configure them all at the same time since they will have to play a role in the circuit that will XML-Encrypt messages. This can be done using the **XML Encryption Wizard**. The wizard is available by right-clicking on the name of the policy in the tree view of the Policy Studio and selecting the **XML Encryption Settings** menu option. The next section describes how to configure the settings on this dialog.

Configuration

The first step in configuring the **XML Encryption Wizard** is to select the certificate that contains the public key to use to encrypt the data. Once the data has been encrypted with this public key it will only be able to be decrypted using the corresponding private key. Select the relevant certificate from the list of **Certificates in the Trusted Certificate Store**.

When the wizard is completed, the information configured on this screen will result in the auto-generation of a **Find Certificate** filter. This filter will be automatically configured to use the selected certificate from the Trusted Certificate Store. Take a look at the [Find Certificate](#) help page for more information on this filter.

After clicking the **Next** button on the first screen of the wizard, the configuration options for the **XML Encryption Settings** filter are displayed. For more information on configuring this filter, please refer to the [XML Encryption Settings](#) help page.

Having completed all the steps in the wizard a policy will be created that comprises a **Find Certificate**, **XML Encryption Settings**, and **XML Encryption** filter. It is possible to insert other filters into this policy as required, however the order of the encryption filters must be maintained as follows:

1. Find Certificate
2. XML Encryption Settings
3. XML Encryption

XML-Encryption

Overview

The **XML-Encryption** filter is responsible for encrypting parts of XML messages based on the settings configured in the **XML-Encryption Settings** filter.

The **XML-Encryption Settings** filter generates the `encryption.properties` message attribute based on configuration settings. The **XML-Encryption** filter uses these properties to perform the encryption of the data.

Configuration

Enter a suitable name for the filter in the **Name** field.

Auto-generation using the XML Encryption Settings Wizard

Because the **XML Encryption** filter must always be used in conjunction with the **XML Encryption Settings** and **Find Certificate** filters, the Policy Studio provides a wizard that can generate these three filters at the same time. To use this wizard, right-click a policy node on the **Policies** tab in the Policy Studio, and select the **XML Encryption Settings** menu option.

For more information on how to configure the **XML Encryption Settings Wizard** see the [XML Encryption Settings Wizard](#) topic.

XML-Decryption Settings

Overview

The Enterprise Gateway can decrypt an XML encrypted message on behalf of its intended recipients. XML Encryption is a W3C standard that enables data to be encrypted and decrypted at the application layer of the OSI stack, thus ensuring complete end-to-end confidentiality of data.

You should use the **XML-Decryption Settings** in conjunction with the **XML-Decryption** filter, which performs the decryption. The **XML-Decryption Settings** generates the `decryption.properties` message attribute, which is required by the **XML-Decryption** filter.

Important Note

The output of a successfully executed decryption filter is the original unencrypted message. Depending on whether the **Remove EncryptedKey used in decryption** has been enabled, all information relating to the encryption key can be removed from the message. For more details, see [Options](#) section.

XML Encryption Overview

XML Encryption facilitates the secure transmission of XML documents between two application endpoints. Whereas traditional transport-level encryption schemes, such as SSL and TLS, can only offer point-to-point security, XML Encryption guarantees complete end-to-end security. Encryption takes place at the application-layer and so the encrypted data can be encapsulated in the message itself. The encrypted data can therefore remain encrypted as it travels along its path to the target Web Service. Furthermore, the data is encrypted such that only its intended recipients can decrypt it.

To understand how the Enterprise Gateway decrypts XML encrypted messages, you should first examine the format of an XML Encryption block. The following example shows a SOAP message containing information about Oracle:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <getCompanyInfo xmlns="www.oracle.com">
      <name>Company</name>
      <description>XML Security Company</description>
    </getCompanyInfo>
  </s:Body>
</s:Envelope>
```

After encrypting the SOAP Body, the message is as follows:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Security xmlns="http://schemas.xmlsoap.org/ws/2003/06/secext" s:actor="Enc">
      <!-- Encapsulates the recipient's key details -->
      <enc:EncryptedKey xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
        Id="00004190E5D1-7529AA14" MimeType="text/xml">
        <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5">
          <enc:KeySize>256</enc:KeySize>
        </enc:EncryptionMethod>
        <enc:CipherData>
```

```

    <!-- The session key encrypted with the recipient's public key -->
    <enc:CipherValue>
        AAAAAJ/lK ... mrTF8Egg==
    </enc:CipherValue>
</enc:CipherData>
<dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
    <dsig:KeyName>sample</dsig:KeyName>
    <dsig:X509Data>
        <!-- The recipient's X.509 certificate -->
        <dsig:X509Certificate>
            MIEZzCCA0 ... fzmC/YR5gA
        </dsig:X509Certificate>
    </dsig:X509Data>
</dsig:KeyInfo>
<enc:CarriedKeyName>Session key</enc:CarriedKeyName>
<enc:ReferenceList>
    <enc:DataReference URI="#00004190E5D1-5F889C11" />
</enc:ReferenceList>
</enc:EncryptedKey>
</Security>
</s:Header>
<enc:EncryptedData xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
    Id="00004190E5D1-5F889C11" MimeType="text/xml"
    Type="http://www.w3.org/2001/04/xmlenc#Element">
    <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc">
    <enc:KeySize>256</enc:KeySize>
    </enc:EncryptionMethod>
    <enc:CipherData>
        <!-- The SOAP Body encrypted with the session key -->
        <enc:CipherValue>
            E2ioF8ib2r ... KJAnrX0GQV
        </enc:CipherValue>
    </enc:CipherData>
    <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
        <dsig:KeyName>Session key</dsig:KeyName>
    </dsig:KeyInfo>
</enc:EncryptedData>
<s:Envelope>

```

The most important elements are as follows:

- **EncryptedKey:** The `EncryptedKey` element encapsulates all information relevant to the encryption key.
- **EncryptionMethod:** The `Algorithm` attribute specifies the algorithm that is used to encrypt the data. The message data (`EncryptedData`) is encrypted using the Advanced Encryption Standard (AES) *symmetric cipher*, but the session key (`EncryptedKey`) is encrypted with the RSA *asymmetric* algorithm.
- **CipherValue:** The value of the encrypted data. The contents of the `CipherValue` element are always Base64 encoded.
- **KeyInfo:** Contains information about the recipient and his encryption key, such as the key name, X.509 certificate, and Common Name.
- **ReferenceList:** This element contains a list of references to encrypted elements in the message. The `ReferenceList` contains a `DataReference` element for each encrypted element, where the value of a `URI` attribute points to the `Id` of the encrypted element. In the previous example, you can see that the `DataReference` `URI` attribute contains the value `#00004190E5D1-5F889C11`, which corresponds with the `Id` of the `EncryptedData` element.
- **EncryptedData:** The XML element(s) or content that has been encrypted. In this case, the SOAP Body element has been encrypted, and so the `EncryptedData` block has replaced the SOAP Body element.

Now that you have seen how encrypted data can be encapsulated in an XML message, it is important to discuss how this data gets encrypted in the first place. When you understand how data is encrypted, the fields that must be configured to decrypt this data become easier to understand.

When a message is encrypted, only the intended recipient(s) of the message can decrypt it. By encrypting the message with the recipient's public key, the sender can be guaranteed that only the intended recipient can decrypt the message using his private key, to which he has sole access. This is the basic principle behind *asymmetric cryptography*.

In practice, however, encrypting and decrypting data with a public-private key pair is notoriously CPU-intensive and time consuming. Because of this, asymmetric cryptography is seldom used to encrypt large amounts of data. The following steps exemplify a more typical encryption process:

1. The sender generates a one-time *symmetric* (or session) key which is used to encrypt the data. Symmetric key encryption is much faster than asymmetric encryption and is far more efficient with large amounts of data.
2. The sender encrypts the data with the symmetric key. This same key can then be used to decrypt the data. It is therefore crucial that only the intended recipient can access the symmetric key and consequently decrypt the data.
3. To ensure that nobody else can decrypt the data, the symmetric key is encrypted with the recipient's *public key*.
4. The data (encrypted with the symmetric key) and session key (encrypted with the recipient's public key) are then sent together to the intended recipient.
5. When the recipient receives the message he, decrypts the encrypted session key using his *private key*. Because the recipient is the only one with access to the private key, he is the only one who can decrypt the encrypted session key.
6. Armed with the decrypted session key, the recipient can decrypt the encrypted data into its original plaintext form.

Now that you understand how XML Encryption works, it is now time to learn how to configure the Enterprise Gateway to decrypt XML encrypted messages. The following sections describe how to configure the **XML Decryption Settings** filter to decrypt encrypted XML data.

Node(s) to Decrypt

An XML message may contain several `EncryptedData` blocks. This section specifies which encryption blocks are to be decrypted. There are two available options:

- [Decrypt All Encrypted Nodes](#)
- [Use XPath to Select Encrypted Nodes](#)

Decrypt All Encrypted Nodes:

The Enterprise Gateway attempts to decrypt *all* `EncryptedData` blocks contained in the message.

Use XPath to Select Encrypted Nodes:

This option enables the administrator to explicitly choose the `EncryptedData` block that the Enterprise Gateway should decrypt.

The following skeleton SOAP message contains two `EncryptedData` blocks:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    ...
  <s:Header>
  <s:Body>
    <!-- 1st EncryptedData block -->
    <e:EncryptedData xmlns:e="http://www.w3.org/2001/04/xmlenc#"

```

```

        Encoding="iso-8859-1" Id="ENC_1" MimeType="text/xml"
        Type="http://www.w3.org/2001/04/xmlenc#Element">
    ...
</e:EncryptedData>
<!-- 2nd EncryptedData block -->
<e:EncryptedData xmlns:e="http://www.w3.org/2001/04/xmlenc#"
    Encoding="iso-8859-1" Id="ENC_2" MimeType="text/xml"
    Type="http://www.w3.org/2001/04/xmlenc#Element">
    ...
</e:EncryptedData>
</s:Body>
</s:Envelope>

```

The `EncryptedData` blocks are selected using XPath. You can use the following XPath expressions to select the respective `EncryptedData` blocks:

<i>EncryptedData Block</i>	<i>XPath Expression</i>
1st	<code>//enc:EncryptedData[@Id='ENC_1']</code>
2nd	<code>//enc:EncryptedData[@Id='ENC_2']</code>

Click the **Add**, **Edit**, or **Delete** buttons to add, edit, or remove an XPath expression.

Decryption Key

This section specifies the key to use to decrypt the encrypted nodes. As discussed in the [Overview](#), data encrypted with a public key can only be decrypted with the corresponding private key. In this section, you can specify the private (decryption) key from the `<KeyInfo>` element of the XML Encryption block, or the certificate stored in the Oracle message attribute can be used to lookup the private key of the intended recipient of the encrypted data in the Certificate Store.

Find via KeyInfo in Message:

Select this option if you wish to determine the decryption key to use from the `KeyInfo` section of the `EncryptedKey` block. The `KeyInfo` section contains a reference to the public key used to encrypt the data. You can use this `KeyInfo` section reference to find the relevant private key (from the Oracle Certificate Store) to use to decrypt the data.

Find via Certificate in Attribute:

Select this option if you do not wish to use the `KeyInfo` section in the message. You can enter a message attribute in this field that contains a certificate, whose corresponding private key is stored in the Oracle Certificate Store.

Typically, a **Find Certificate** filter is used in a policy to locate an appropriate certificate and store it in the `certificate` message attribute. When the certificate has been stored in this attribute, the **XML Decryption Settings** filter can use this certificate to lookup the Certificate Store for a corresponding private key for the public key stored in the certificate. To do this, select the `certificate` attribute from the drop-down list.

Options

The following configuration options are available in this section:

Fail if no encrypted data found:

If this option is selected, the filter fails if no <EncryptedData> elements are found within the message.

Remove the EncryptedKey used in decryption:

Select this option to remove information relating to the decryption key from the message. When this option is selected, the <EncryptedKey> block is removed from the message.

It is important to note that in cases where the <EncryptedKey> block has been included in the <EncryptedData> block, it is removed regardless of whether this setting has been selected.

Default Derived Key Label:

If the Enterprise Gateway consumes a <DerivedKeyToken>, the default value entered is used to recreate the derived key that is used to decrypt the encrypted data.

Algorithm Suite Required:

Select the WS-Security Policy *Algorithm Suite* that must have been used when encrypting the message. This check ensures that the appropriate algorithms were used to encrypt the message.

Auto-generation using the XML Decryption Wizard

Because the **XML Decryption Settings** filter must always be paired with an **XML Decryption** filter, it makes sense to have a wizard that can generate both of these filters at the same time. To use the wizard, right-click the name of the policy in the tree view of the Policy Studio, and select the **XML Decryption Settings** menu option.

Configure the fields on the **XML Decryption Settings** dialog as explained in the previous sections. When finished, an **XML Decryption Settings** filter is created along with an **XML Decryption** filter.

XML-Decryption

Overview

The **XML Decryption** filter is responsible for decrypting data in XML messages based on the settings configured in the **XML-Decryption Settings** filter.

The **XML-Decryption Settings** filter generates the `decryption.properties` message attribute based on configuration settings. The **XML-Decryption** filter uses these properties to perform the decryption of the data.

Configuration

Enter an appropriate name for the filter in the **Name** field.

Auto-generation using the XML Decryption Wizard

Because the **XML Decryption** filter must always be paired with an **XML Decryption Settings** filter, the Policy Studio provides a wizard that can generate both of these filters at the same time. To use the wizard, right-click a policy node on the **Policies** tab in the Policy Studio, and select **XML Decryption Settings**.

Configure the fields on the **XML Decryption Settings** dialog as explained in the [XML Decryption Settings](#) topic. When finished, an **XML Decryption Settings** filter is created along with an **XML Decryption** filter.

SMIME Encryption

Overview

You can use the **SMIME Encryption** filter to generate an encrypted Secure/Multipurpose Internet Mail Extensions (SMIME) message. This filter enables you to configure the certificates of the recipients of the encrypted message. You can also configure advanced options such as ciphers and Base64 encoding.

General Configuration

Complete the following general field:

Name:

Enter an appropriate name for the filter.

Recipients

The **Recipients** tab enables you to configure the certificates of the recipients of the encrypted SMIME message. Select one of the following options:

Use the following certificates:

This is the default option. Select the certificates of the recipients of the encrypted message. The public keys associated with these certificates are used to encrypt the data so that it can only be decrypted using the associated private keys.

Certificate in attribute:

Alternatively, enter the message attribute that contains the certificate of the recipients of the encrypted message. Defaults to the `certificate` message attribute.

Advanced

The **Advanced** tab includes the following settings:

Cipher:

Enter the cipher that you want to use to encrypt the message data. Defaults to the `DES-EDE3-CBC` cipher.

Content-Type:

Enter the `Content-Type` of the message data. Defaults to `application/pkcs7-mime`.

Base64 encode:

Select whether to Base64 encode the message data. This option is not selected by default.

SMIME Decryption

Overview

The **SMIME Decryption** filter can be used to decrypt an encrypted SMIME message.

Configuration

Complete the following fields to configure this filter:

Name:

Enter a name for the filter in the **Name** field.

Use Certificate to Decrypt:

Check the box next to the certificate that you want to use to decrypt the encrypted PKCS#7 message with. The private key associated with this certificate will be used to actually decrypt the message.

SOAP Fault

Overview

In cases where a typical SOAP transaction fails, a *SOAP Fault* can be used to convey error information to the SOAP client. The following message shows the format of a SOAP Fault:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>Receiver</env:Value>
        <env:Subcode>
          <env:Value>policy failed</env:Value>
        </env:Subcode>
      </env:Code>
      <env:Detail xmlns:oraclefault="http://www.oracle.com/soapfaults"
        oraclefault:type="exception" type="exception"/>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

By default, the Enterprise Gateway returns a very basic SOAP Fault to the client when a message filter has failed. The **SOAP Fault** processor can be added to a policy to return more complicated error information to the client.

For security reasons, it is good practice to return as little information as possible to the client. However, for diagnostic reasons, it is useful to return as much information to the client as possible. Using the **SOAP Fault** processor, administrators have the flexibility to configure just how much information to return to clients, depending on their individual requirements.

SOAP Fault Format

The following configuration options are available in this section:

SOAP Version:

It is possible to send either a SOAP Fault 1.1 or 1.2 response to the client. Select the appropriate version using the radio buttons provided.

Fault Namespace:

Select the default namespace to use in SOAP Faults, or enter a new one if necessary.

Indent SOAP Fault:

If this option is checked, an XSL stylesheet will be run over the SOAP Fault in order to indent nested XML elements. The indented SOAP Fault will be returned to the client.

SOAP Fault Contents

The following configuration options are available in this section:

Show Detailed Explanation of Fault:

If this option is checked, a detailed explanation of the SOAP Fault will be returned within the fault message. This makes it possible to suppress the reason for the exception in a tightly locked down system, i.e. the reason will simply appear as "message blocked" in the SOAP Fault.

Show Filter Execution Path

When this option is checked, the Enterprise Gateway will return a SOAP Fault containing the list of filters run on the message before the error occurred. For each filter listed in the SOAP Fault, the status (i.e. "pass" or "fail") is given. The following message shows a *filter execution path* returned in a SOAP Fault:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
  </env:Header>
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>Receiver</env:Value>
        <env:Subcode>
          <env:Value>policy failed</env:Value>
        </env:Subcode>
      </env:Code>
      <env:Detail xmlns:oraclefault="http://www.oracle.com/soapfaults"
        oraclefault:type="exception" type="exception">
        <oraclefault:path>
          <oraclefault:visit node="HTTP Parser" status="Pass"></oraclefault:visit>
          <oraclefault:visit node="/services" status="Fail"></oraclefault:visit>
          <oraclefault:visit node="/status" status="Fail"></oraclefault:visit>
        </oraclefault:path>
        </env:Detail>
      </env:Fault>
    </env:Body>
  </env:Envelope>
```

Show Stack Trace

If this option is checked, the Enterprise Gateway will return the Java stack trace for the error to the client. This option should only be enabled under instructions from the Oracle Support Team.

Show Current Message Attributes

By checking this option, the message attributes present at the time the SOAP Fault was generated will be returned to the client. Each message attribute will form the content of a `<fault:attribute>` element, as illustrated in the following example:

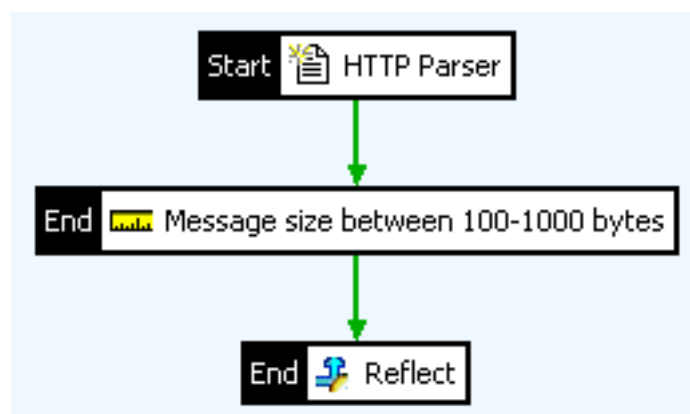
```
<fault:attributes>
  <fault:attribute name="circuit.failure.reason" value="null">
  <fault:attribute name="circuit.lastProcessor" value="HTTP Digest">
  <fault:attribute name="http.request.clientaddr" value="/127.0.0.1:4147">
  <fault:attribute name="http.response.status" value="401">
  <fault:attribute name="http.request.uri" value="/authn">
  <fault:attribute name="http.request.verb" value="POST">
  <fault:attribute name="http.response.info" value="Authentication Required">
  <fault:attribute name="circuit.name" value="Digest AuthN">
</fault:attributes>
```

Customized SOAP Faults

It is possible to create customized SOAP Faults using the **Set Message** filter. The **Set Message** filter can change the contents of the message body to any arbitrary content. When an exception occurs in a policy, it is possible to use this filter to customize the body of the SOAP fault. The following example demonstrates how to generate customized SOAP faults and return them to the client.

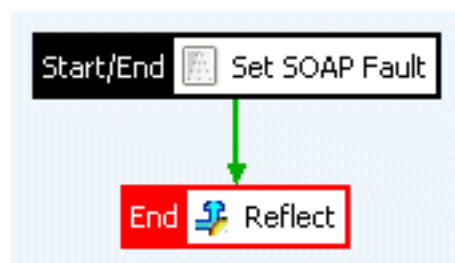
Step 1: Create the Top-level Policy:

We will first create a very simple policy called "Main Circuit". This policy will ensure the size of incoming messages is between 100 and 1000 bytes. Messages within this range will be echoed back to the client.



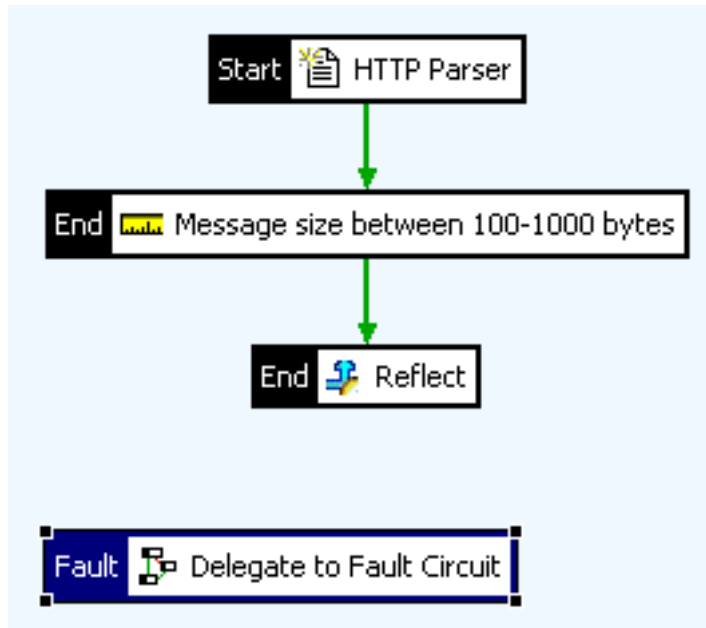
Step 2: Create the Fault Policy

Next, create a second policy called "Fault Circuit". This policy will use the **Set Message** filter to customize the body of the SOAP fault. When configuring this filter, enter the contents of the customized SOAP fault that you want to return to clients in the text area provided.



Step 3: Create a shortcut to the Fault Policy

Add a **Policy Shortcut** filter to the "Main Circuit" and configure it to refer to the "Fault Circuit". Do **not** connect this filter to the policy. Instead, right-click on the filter and select the **Set as Fault Handler** menu option. The "Main Circuit" now appears as follows:



So how does it work? Let's assume a 2000-byte message is received by the Enterprise Gateway and is passed to the "Main Circuit" for processing. The message is parsed by the **HTTP Parser** filter and then the size of the message is checked by the **Message Size** filter. Since the message is greater than the size constraints set by this filter, and since there is no failure path configured for this filter, an exception will be thrown.

When an exception is thrown in a policy, it is handled by the designated *Fault Handler*, if one is present. In the "Main Circuit", we have set a **Policy Shortcut** filter to be the fault handler. This filter delegates to the "Fault Circuit", meaning that when an exception occurs, "Main Circuit" will invoke (or delegate to) the "Fault Circuit".

The "Fault Circuit" consists of 2 filters, which play the following roles:

1. **Set Message:**
This filter is used to set the body of the message to the contents of the customized SOAP fault.
2. **Reflect:**
Once the SOAP fault has been set to the message body, it will be returned to the client using the **Reflect** filter.

XML Signature Generation

Overview

The Enterprise Gateway can sign both SOAP and non-SOAP XML messages. Attachments to the message can also be signed. The resultant XML Signature is inserted into the message for consumption by a downstream Web Service. At the Web Service, the signature can be used to authenticate the message sender and/or verify the integrity of the message.

Enter a name for the filter in the **Name** field. The **Signature** and **Advanced** top-level tabs contain the following tabs, which can be used to configure various aspects of the generated XML Signature.

Signing Key

It is possible to use either a symmetric or an asymmetric key to sign the message content. Select the appropriate radio button and configure the fields on the corresponding tab.

Asymmetric Key

With an asymmetric signature, the signatory's private key (from a public-private key pair) is used to sign the message. The corresponding public key is then used to verify the signature. The following fields are available for configuration on this tab:

Key in Store:

To use a signing key from the Certificate Store, check the **Key in Store** radio button and then click the **Signing Key** button. Select a certificate that has the required signing key associated with it. The signing key can also be stored on a HSM (Hardware Security Module). Please refer to the [Certificate Store](#) help page for more information on storing signing keys.

The *Distinguished Name* of the selected certificate will appear in the `X509SubjectName` element of the XML Signature as follows:

```
<dsig:X509SubjectName>  
  CN=Sample,OU=R&D,O=Company Ltd.,L=Dublin 4,ST=Dublin,C=IE  
</dsig:X509SubjectName>
```

Key in Message Attribute:

Alternatively, the signing key may have already been used by another filter and stored in a message attribute. In order to reuse this key, you can select the **Key in Message Attribute** radio button and enter the name of the attribute in the field provided.

Symmetric Key

With a symmetric signature, the same key is used to sign and verify the message. Typically the client generates the symmetric key and uses it to sign the message. The key must then be transmitted to the recipient so that they can verify the signature. It would be unsafe to transmit an unprotected key along with the message so it is usually encrypted (or wrapped) with the recipient's public key. The key can then be decrypted with the recipient's private key and can then be used to verify the signature. The following configuration options are available on this screen:

Generate Symmetric Key:

Select this option if you want this filter to generate a symmetric key.

Symmetric Key in Message Attribute:

Alternatively, if you have already generated a symmetric key using a different filter (e.g. an Encryption filter) and stored it in a message attribute, you can select this radio button and enter the name of the message attribute in the field provided.

Include Encrypted Symmetric Key in Message:

As described earlier, the symmetric key is typically encrypted for the recipient and included in the message. However, it is possible that the initiator and recipient of the transaction have agreed on a symmetric key using some out-of-bounds mechanism. In this case, it is not necessary to include the key in the message. However, the default option is to include the encrypted symmetric key in the message. The <KeyInfo> section of the Signature will point to the <EncryptedKey>.

Encrypt with Key in Store:

Select this option to encrypt the symmetric key with a public key from the Certificate Store. Click the **Signing Key** button and then select the certificate that contains the public key of the recipient. By encrypting the symmetric key with this public key, you are ensuring that only the recipient that has access to the corresponding private key will be able to decrypt the encrypted symmetric key.

Encrypt with Key in Message Attribute:

You can also use a key stored in a message attribute to encrypt (or wrap) the symmetric key. Select this radio button and enter the name of the message attribute that contains the public key you want to use to encrypt the symmetric key with.

Use Derived Key:

A <wssc:DerivedKeyToken> token can be used to derive a symmetric key from the original symmetric key held in and <enc:EncryptedKey>. The derived symmetric key is then used to actually sign the message, as opposed to the original symmetric key. It must be derived again during the verification process using the parameters in the <wssc:DerivedKeyToken>. One of these parameters is the symmetric key held in <enc:EncryptedKey>. The following example shows the use of a derived key:

```
<enc:EncryptedKey Id="Id-0000010b8b0415dc-0000000000000000">
  <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
  <dsig:KeyInfo>
    ...
  </dsig:KeyInfo>
  <enc:CipherData>
</enc:EncryptedKey>

<wssc:DerivedKeyToken wsu:Id="Id-0000010bd2b8eca1-00000000000000017"
  Algorithm="http://schemas.xmlsoap.org/ws/2005/02/sc/dk/p_sha1">
  <wsse:SecurityTokenReference wsu:Id="Id-0000010bd2b8ed5d-00000000000000018">
    <wsse:Reference URI="#Id Id-0000010b8b0415dc-0000000000000000"
      ValueType=".../oasis-wss-soap-message-security-1.1#EncryptedKey"/>
  </wsse:SecurityTokenReference>
  <wssc:Generation>0</wssc:Generation>
  <wssc:Length>32</wssc:Length>
  <wssc:Label>WS-SecureConverstaionWS-SecureConverstaion</wssc:Label>
  <wssc:Nonce>h9TTWKRYlCOz87+mcl/7Pg==</wssc:Nonce>
</wssc:DerivedKeyToken>

<dsig:Signature Id="Id-0000010b8b0415dc-00000000000000004">
  <dsig:SignedInfo>
    <dsig:CanonicalizationMethod
      Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <dsig:SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
    <dsig:Reference>...</dsig:Reference>
  </dsig:SignedInfo>
  <dsig:SignatureValue>...dsig:SignatureValue>
  <dsig:KeyInfo>
    <wsse:SecurityTokenReference wsu:Id="Id-0000010b8b0415dc-0000000000000006">
      <wsse:Reference
```

```

        URI="# Id-0000010bd2b8eca1-00000000000000017"
        ValueType="http://schemas.xmlsoap.org/ws/2005/02/sc/dk"/>
    </wsse:SecurityTokenReference>
</dsig:KeyInfo>
</dsig:Signature>

```

Symmetric Key Length:

This option allows the user to specify the length of the key to use when performing symmetric key signatures. It is important to realize that the longer the key, the stronger the encryption.

Key Info

This tab configures how the <KeyInfo> block of the generated XML Signature will appear. Configure the following fields on this tab:

Do Not Include KeyInfo Section:

This option allows you to omit all information about the signatory's certificate from the signature. In other words, the *KeyInfo* element is omitted from the signature. This is useful where a downstream Web Service uses an alternative method of authenticating the signatory, and wishes to use the signature for the sole purpose of verifying the integrity of the message. In such cases, adding certificate information to the message may be regarded as an unnecessary overhead.

Include Certificate:

This is the default option which places the signatory's certificate inside the XML Signature itself. The following example, shows an example of an XML Signature which has been created using this option:

```

<dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" id="Sample">
    ...
    <dsig:KeyInfo>
        <dsig:X509Data>
            <dsig:X509SubjectName>CN=Sample...</dsig:X509SubjectName>
            <dsig:X509Certificate>
                MII EZDCCA0yg
                ....
                RNP9aKD1fEQgJ
            </dsig:X509Certificate>
        </dsig:X509Data>
    </dsig:KeyInfo>
</dsig:Signature>

```

Expand Public Key:

The details of the signatory's public key are inserted into a *KeyValue* block. The *KeyValue* block is only inserted when this option is checked.

```

<dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" id="Sample">
    ...

```

```

<dsig:KeyInfo>
  <dsig:X509Data>
    <dsig:X509SubjectName>CN=Sample...</dsig:X509SubjectName>
    <dsig:X509Certificate>
      MIIIE ..... EQgJ
    </dsig:X509Certificate>
  </dsig:X509Data>
  <dsig:KeyValue>
    <dsig:RSAKeyValue>
      <dsig:Modulus>
        AMfb2tT53GmMiD
        ...
        NmrNht7iy18=
      </dsig:Modulus>
      <dsig:Exponent>AQAB</dsig:Exponent>
    </dsig:RSAKeyValue>
  </dsig:KeyValue>
</dsig:KeyInfo>
</dsig:Signature>

```

Include Distinguished Name:

If this checkbox is checked, the Distinguished Name of the signatory's X.509 certificate will be inserted in an `<X509SubjectName>` element as shown in the following example:

```

<dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" id="Sample">
  ...
  <dsig:KeyInfo>
    <dsig:X509Data>
      <dsig:X509SubjectName>CN=Sample,C=IE...</dsig:X509SubjectName>
      <dsig:X509Certificate>
        MII EZDCCA0yg
        ...
        RNp9aKD1fEQgJ
      </dsig:X509Certificate>
    </dsig:X509Data>
  </dsig:KeyInfo>
</dsig:Signature>

```

Include Key Name:

This option allows you insert a key identifier, or `KeyName`, to allow the recipient to identify the signatory. Enter an appropriate value for the `KeyName` in the **Value** field. Typical values include Distinguished Names (DName) from X.509 certificates, key IDs, or email addresses. Specify whether the specified value is a **Text value** of a **Distinguished name attribute** by checking the appropriate radio button.

```

<dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" id="Sample">
  ...

```

```

<dsig:KeyInfo>
  <dsig:KeyName>test@oracle.com</dsig:KeyName>
</dsig:KeyInfo>
</dsig:Signature>

```

Put Certificate in an Attachment:

The Enterprise Gateway supports SOAP messages with attachments. By selecting this option, you can save the signatory's certificate to the file specified in the input field. This file can then be sent along with the SOAP message as a SOAP attachment.

From previous examples, it is clear that the user's certificate is usually placed inside a `KeyInfo` element. However, in this example, the certificate is actually contained within an attachment, and not within the XML Signature itself. Clearly, we need a way to reference the certificate from the XML Signature, so that validating applications can process the signature correctly. This is the role of the `SecurityTokenReference` block.

The `SecurityTokenReference` block provides a generic way for applications to retrieve security tokens in cases where these tokens are not contained within the SOAP message. The name of the security token is specified in the `URI` attribute of the `Reference` element.

```

<dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" id="Sample">
  ...
  <dsig:KeyInfo>
    <wsse:SecurityTokenReference xmlns:wsse="http://schemas.xmlsoap.org/ws/...">
      <wsse:Reference URI="c:\myCertificate.txt"/>
    </wsse:SecurityTokenReference>
  </dsig:KeyInfo>
</dsig:Signature>

```

When the message is actually sent, the certificate attachment will be given a "Content-Id" corresponding to the `URI` attribute of the `Reference` element. The following example shows what the complete multipart MIME SOAP message looks like as it is sent over the wire. It should help illustrate how the `Reference` element actually refers to the "Content-ID" of the attachment:

```

POST /adoWebSvc.asmx HTTP/1.0
Content-Length: 3790
User-Agent: Enterprise Gateway
Accept-Language: en
Content-Type: multipart/related; type="text/xml";
              boundary="-----Multipart-SOAP-boundary"

-----Multipart-SOAP-boundary
Content-Id: soap-envelope
Content-Type: text/xml; charset="utf-8";
SOAPAction=getQuote

```

```

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  ...
  <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" id="Sample">
    ...
    <dsig:KeyInfo>
      <ws:SecurityTokenReference xmlns:ws="http://schemas.xmlsoap.org/ws/...">
        <ws:Reference URI="c:\myCertificate.txt"/>
      </ws:SecurityTokenReference>
    </dsig:KeyInfo>
  </dsig:Signature>
  ...
</s:Envelope>

-----Multipart-SOAP-boundary
Content-Id: c:\myCertificate.txt
Content-Type: text/plain; charset="US-ASCII"

MIEZDCCA0ygAwIBAgIBAzANBgkqhki
...
7uFveG0eL0zBwZ5qwLRNp9aKD1fEQgJ
-----Multipart-SOAP-boundary-

```

Security Token Reference:

A `<wsse:SecurityTokenReference>` element can be used to point to the security token used in the generation of the signature. Select this option if you wish to use this element. The type of the reference must be selected from the **Reference Type** dropdown.

The `<wsse:SecurityTokenReference>`, (within the `<dsig:KeyInfo>`), may contain a `<wsse:Embedded>` security token. Alternatively, the `<wsse:SecurityTokenReference>`, (within the `<dsig:KeyInfo>`), may refer to a certificate via a `<dsig:X509Data>`. Select the appropriate button, **Embed** or **Refer**, depending on whether you want to use an embedded security token or a referred one.

You can make sure to include a `<BinarySecurityToken>` (BST) that contains the certificate used to wrap the symmetric key in the message by selecting the **Include BinarySecurityToken** option. The BST will be inserted into the WS-Security header regardless of the type of Security Token Reference selected from the dropdown.

It is important to note that when using the "Kerberos Token Profile" standard and the Enterprise Gateway is acting as the initiator of a secure transaction, it can use Kerberos session keys to sign a message. The KeyInfo must be configured to use a Security Token Reference with a ValueType of "GSS_Kerberosv5_AP_REQ". In this case, the Kerberos token is contained within a `<BinarySecurityToken>` within the message.

If the Enterprise Gateway is acting as the recipient of a secure transaction, it can also use the Kerberos session keys to sign the message returned to the client. However, in this case, the KeyInfo must be configured to use a Security Token Reference with ValueType of "Kerberosv5_APREQSHA1". When this ValueType is selected, the Kerberos token is not contained within the message. The Security Token Reference contains a SHA1 digest of the original Kerberos token received from the client, which identifies the session keys to the client.

Note that when using the "WS-Trust for SPENGO" standard, the Kerberos session keys are not used directly to sign messages since a security context with an associated symmetric key is negotiated. This symmetric key is shared by both client and service and can be used to sign messages on both sides.

What to Sign

This tab is used to identify parts of the message that must be signed. Each signed part will be referenced from within the generated XML Signature.

It is possible to use *any* combination of **Node Locations**, **XPaths**, **XPath Predicates**, and the nodes contained in a **Message Attribute** to specify what must be signed. Please refer to the [Locate XML Nodes](#) filter for more information on how to use these node selectors.

It is important to consider the mechanisms available for referencing signed elements from within an XML Signature. With WSU IDs, an ID attribute is inserted into the root element of the nodeset that is to be signed. The XML Signature then references this ID to indicate to verifiers of the Signature the nodes that were signed. The use of WSU IDs is the default option since they are WS-I compliant.

Alternatively, a generic ID attribute (that is not bound to the WSU namespace) can be used to dereference the data. The ID attribute is inserted into the top-level element of the nodeset that is to be signed. The generated XML Signature can then reference this ID to indicate what nodes were signed.

When XPath transforms are used, an XPath expression that points to the root node of the nodeset that is signed will be inserted into the XML Signature. When attempting to verify the Signature, this XPath expression must be run on the message to retrieve the signed content.

Use WSU IDs:

Select this option to reference the signed data using a `wsu:Id` attribute. In this case, a `wsu:Id` attribute is inserted into the root node of the nodeset that is signed. This id is then referenced in the generated XML Signature as an indication of what nodes were signed. The following example shows the correlation:

```
<soap:Envelope xmlns:soap="...">
  <soap:Header>
    <wsse:Security xmlns:wsse="...">
      <dsig:Signature xmlns:dsig="..." Id="Id-00000112e2c98df8-00000000000000004">
        <dsig:SignedInfo>
          <dsig:CanonicalizationMethod
            Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <dsig:SignatureMethod
            Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
          <dsig:Reference URI="#Id-00000112e2c98df8-00000000000000003">
            <dsig:Transforms>
              <dsig:Transform
                Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
            </dsig:Transforms>
            <dsig:DigestMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
            <dsig:DigestValue>xChPoiWJjrrPZkbXN8FPB8S4U7w=</dsig:DigestValue>
          </dsig:Reference>
        </dsig:SignedInfo>
        <dsig:SignatureValue>KG4N .... /9dw==</dsig:SignatureValue>
        <dsig:KeyInfo Id="Id-00000112e2c98df8-00000000000000005">
          <dsig:X509Data>
            <dsig:X509Certificate>
              MIID ... ZiBQ==
            </dsig:X509Certificate>
          </dsig:X509Data>
        </dsig:KeyInfo>
      </dsig:Signature>
    </wsse:Security>
  </soap:Header>
  <soap:Body xmlns:wsu="..." wsu:Id="Id-00000112e2c98df8-00000000000000003">
    <vs:getProductInfo xmlns:vs="http://www.oracle.com">
      <vs:Name>Enterprise Gateway</vs:Name>
      <vs:Version>11.1.1.5.0</vs:Version>
    </vs:getProductInfo>
  </s:Body>
</s:Envelope>
```

In the above example, a `wsu:Id` attribute has been inserted into the `<soap:Body>` element. This `wsu:Id` attribute is then referenced by the `URI` attribute of the `<dsig:Reference>` element in the actual Signature.

When the Signature is being verified, the value of the `URI` attribute can be used to locate the nodes that have been signed.

Use IDs:

Select this option in order to use generic IDs (that are not bound to the WSU namespace) to dereference the signed data. Under this schema, the `URI` attribute of the `<Reference>` points at an ID attribute, which is inserted into the top-level node of the nodeset that is signed. Take a look at the following example, noting how the ID specified in the Signature matches the ID attribute that has been inserted into the `<Body>` element, indicating that the Signature applies to the entire contents of the SOAP Body.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
      Id="Id-0000011a101b167c-00000000000000013">
      <dsig:SignedInfo>
        <dsig:CanonicalizationMethod
          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
        <dsig:SignatureMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
        <dsig:Reference URI="#Id-0000011a101b167c-00000000000000012">
          <dsig:Transforms>
            <dsig:Transform
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          </dsig:Transforms>
          <dsig:DigestMethod
            Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
          <dsig:DigestValue>JCy0JoyhVZYzmrLrl92nxfr1+zQ=</dsig:DigestValue>
        </dsig:Reference>
      </dsig:SignedInfo>
      <dsig:SignatureValue>.....<dsig:SignatureValue>
      <dsig:KeyInfo Id="Id-0000011a101b167c-00000000000000014">
        <dsig:X509Data>
          <dsig:X509Certificate>.....</dsig:X509Certificate>
        </dsig:X509Data>
      </dsig:KeyInfo>
    </dsig:Signature>
  </soap:Header>
  <soap:Body Id="Id-0000011a101b167c-00000000000000012">
    <product version="11.1.1.5.0">
      <name>Enterprise Gateway</name>
      <company>Oracle</company>
      <description>SOA Security and Management</description>
    </product>
  </soap:Body>
</soap:Envelope>
```


Use SAML IDs for SAML Elements:

This option is only relevant if a SAML assertion is required to be signed. If this option is checked and the signature is to cover a SAML assertion, an `AssertionID` attribute will be inserted into a SAML version 1.1 assertion, or an `ID` attribute will be inserted into a SAML version 2.0 assertion. The value of this attribute will then be referenced from within a `<Reference>` block of the XML Signature.

Where to Place Signature**Append Signature to Root or SOAP Header:**

If the message is a SOAP message, the signature will be inserted into the `SOAP Header` element when this radio button is selected. The XML Signature will be inserted as an immediate child of the `SOAP Header` element. The following example shows a skeleton SOAP message which has been signed using this option:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <ws:Security xmlns:ws="http://schemas.xmlsoap.org/..." s:actor="test">
      <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/..." id="Sample">
        ...
      </dsig:Signature>
    </ws:Security>
  </s:Header>
  <s:Body>
    ...
  </s:Body>
</s:Envelope>
```

If, on the other hand, the message is just plain XML, the signature will be inserted as an immediate child of the root element of the XML message. The following example shows a non-SOAP XML message which has been signed using this option:

```
<PurchaseOrder>
  <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" id="Sample">
    ...
  </dsig:Signature>

  <Items>
    ...
  </Items>
</PurchaseOrder>
```

Place in WS-Security Element for SOAP Actor/Role:

By selecting this option, the XML Signature will be inserted into the WS-Security element identified by the specified SOAP *actor* or *role*. A SOAP actor/role is simply a way of distinguishing a particular WS-Security block from others which may be present in the message. Actors belong to the SOAP 1.1 specification, but were replaced in SOAP 1.2 by roles. Conceptually, however, they are identical.

Enter the name of the SOAP actor or role of the WS-Security block in the dropdown. The following SOAP message contains an XML Signature within a WS-Security block identified by the "test" actor:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <ws:Security xmlns:ws="http://schemas.xmlsoap.org/..." s:actor="test">
      <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/..." id="Sample">
        ...
      </dsig:Signature>
    </ws:Security>
  </s:Header>
  <s:Body>
    ...
  </s:Body>
</s:Envelope>
```

Use XPath Location:

This option is useful in cases where the signature must be inserted into a non-SOAP XML message. In such cases, it is possible to insert the signature into a location pointed to by an XPath expression. Select or add an XPath expression in the field provided, and then specify whether the Enterprise Gateway should insert the signature *before* the location to which the XPath expression points, or *append* it to this location.

Additional

The **Additional** tab allows you to select additional elements from the message that are to be signed. It is also possible to insert a WS-Security Timestamp into the XML Signature, if necessary.

Additional Elements to Sign:

The options here allow you to select other parts of the message that you may wish to sign.

- **Sign KeyInfo Element of Signature:**

The <KeyInfo> block of the XML Signature can be signed to prevent people cut-and-pasting a different <KeyInfo> block into the message, which may point to some other key material, for example.

- **Sign Timestamp:**

As stated earlier, timestamps are used to prevent replay attacks. However, to guarantee the end-to-end integrity of the timestamp, it is necessary to sign it. Note that this option is only enabled when you have elected to insert a Timestamp into the message using the relevant fields on the **Timestamp Options** panel below.

- **Sign Attachments:**

In addition to signing some or all of the contents of the SOAP message, it is also possible to sign attachments to the SOAP message. To sign all attachments, check the **Include Attachments** checkbox.

A signed attachment is referenced in an XML Signature using the *Content-Id* or *cid* of the attachment. The *URI* attribute of the *Reference* element corresponds to this Content-Id. The following example shows how an XML Signature refers to a sample attachment. It shows the wire format of the message and its attachment as they are sent to the destination Web Service. Multiple attachments will result in successive *Reference* elements.

```
POST /myAttachments HTTP/1.0
Content-Length: 1000
```

```

User-Agent: Enterprise Gateway
Accept-Language: en
Content-Type: multipart/related; type="text/xml";
            boundary="-----Multipart-SOAP-boundary"

-----Multipart-SOAP-boundary
Content-Id: soap-envelope
SOAPAction: none
Content-Type: text/xml; charset="utf-8"

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <dsig:Signature id="Sample" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
      <dsig:SignedInfo>
        <dsig:CanonicalizationMethod
          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n"/>
        <dsig:SignatureMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
        <dsig:Reference URI="cid:moredata.txt">...</dsig:Reference>
      </dsig:SignedInfo>
    </dsig:Signature>
  </s:Header>
  <s:Body>
    ...
  </s:Body>
</s:Envelope>

-----Multipart-SOAP-boundary
Content-Id: moredata.txt
Content-Type: text/plain; charset="UTF-8"

Some more data.
-----Multipart-SOAP-boundary--

```

Transform:

This dropdown is only available when you have selected the **Sign Attachments** box above. It determines the transform used to reference the signed attachments.

Timestamp Options:

It is possible to insert a timestamp into the message to indicate when exactly the signature was generated. Consumers of the signature can then validate the signature to ensure that it is not of date.

The following options are available:

1. **No Timestamp:**
No timestamp will be inserted into the signature.
2. **Embed in WSSE Security:**
The "wsu:Timestamp" will be inserted into a "wsse:Security" block. The Security block is identified by the SOAP actor/role specified on the **Signature** tab.
3. **Embed in Signature Property:**
The "wsu:Timestamp" will be placed inside a signature property element in the "dsig:Signature".

The **Expires In** fields allow the user to optionally specify the "wsu:Expires" for the "wsu:Timestamp". If all fields are left at "0", no "wsu:Expires" element will be placed inside the "wsu:Timestamp".

The following examples shows a "wsu:Timestamp" that has been inserted into a "wsse:Security" block:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <wsse:Security>
      <wsu:Timestamp wsu:Id="Id-0000011294a0311e-0000000000000003d">
        <wsu:Created>2007-05-16T11:22:45Z</wsu:Created>
        <wsu:Expires>2007-05-23T11:22:45Z</wsu:Expires>
      </wsu:Timestamp>
      <dsig:Signature ...>
        ...
      </dsig:Signature ...>
    </wsse:Security>
  </s:Header>
  <s:Body>
    ...
  </s:Body>
</s:Envelope>
```

Algorithm Suite

The fields on this tab determine the combination of cryptographic algorithms and ciphers that are used to sign the message parts.

Algorithm Suite:

WS-Security Policy defines a number of *algorithm suites* that group together a number of cryptographic algorithms. For example, a given algorithm suite will use specific algorithms for asymmetric signing, symmetric signing, asymmetric key wrap, and so on. Therefore, by specifying an algorithm suite, you are effectively selecting a whole suite of cryptographic algorithms to use.

If you want to use a particular WS-Security Policy algorithm suite, you can select it here. The **Signature Method**, **Key Wrap Algorithm**, and **Digest Method** fields will then be automatically populated with the corresponding algorithms for that suite.

Signature Method:

The **Signature Method** field allows you to configure the method used to generate the signature. Various strengths of the HMAC-SHA1 algorithms are available from the dropdown.

Key Wrap Algorithm:

Select the algorithm to use to wrap (i.e. encrypt) the symmetric signing key. This option need only be configured when you are using a symmetric key to sign the message.

Digest Algorithm:

Select the digest algorithm to you to produce a cryptographic hash of the signed data.

Options

Advanced Options:

This section allows you to configure various advanced options on the generated XML Signature. The following fields can be configured on this tab:

WS-Security Options:

WSSE 1.1 defines a <SignatureConfirmation> element that can be used as proof that a particular XML Signature was processed. A recipient and verifier of an XML Signature must generate a <SignatureConfirmation> element for each piece of data that was signed (i.e. for each <Reference> in the XML Signature). A <SignatureConfirmation> element contains the hash of the signed data and must be signed by the recipient before returning it in the response to the initiator (i.e. the original signatory of the data).

When the initiator receives the <SignatureConfirmation> elements in the response, it compares the hash with the hash of the data that it produced initially. If the hashes match, the initiator knows that the recipient has processed the same signature.

Select the **Initiator** option here if the Enterprise Gateway is the initiator as outlined in the scenario above. The Enterprise Gateway will keep a record of the signed data and will compare it to the contents of the <SignatureConfirmation> elements returned from the recipient in the response message.

Alternatively, if the Enterprise Gateway is acting as the recipient in this transaction, you can select the **Responder** radio button to instruct the Enterprise Gateway to generate the <SignatureConfirmation> elements and return them to the initiator. The signature confirmations will be added to the WS-Security header.

Layout Type:

Select the WS-Security Policy layout type that you want the XML Signature and any generated tokens to adhere to. This includes elements such as <Signature>, <BinarySecurityToken>, and <EncryptedKey>, which can all be generated as part of the signing process.

Fail if No Nodes to Sign:

Check this option if you want the filter to fail if it cannot find any nodes to sign as configured on the **What to Sign** tab.

Add Inclusive Namespaces for Exclusive Canonicalization:

It is possible to include information about the namespaces (and their associated prefixes) of signed elements in the signature itself. This ensures that namespaces that are in the same scope as the signed element, but not directly or "visibly" used by this element, are included in the signature. This ensures that the signature can be validated as a standalone entity outside of the context of the message from which it was extracted.

It is also worth pointing out that the WS-I specification only permits the use of exclusive canonicalization in an XML Signature. The <InclusiveNamespaces> element is an attempt to take advantage of some of the behavior of *inclusive* canonicalization, while maintaining the simplicity of *exclusive canonicalization*.

A *PrefixList* attribute is used to list the prefixes of in-scope, but not visibly used elements and attributes. The following example shows how the *PrefixList* attribute is used in practice:

```
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope'>
  <soap:Header>
    <wsse:Security xmlns:wsse='http://docs.oasis-open.org/...'
                  xmlns:wsu='http://docs.oasis-open.org/...'>
      <wsse:BinarySecurityToken wsu:Id='SomeCert'
                              ValueType='http://docs.oasis-open.org/...'>
        lui+Jy4WYKGJW5xM3aHnLxOpGVIpzSg4V486hHFe7sH
      </wsse:BinarySecurityToken>
      <ds:Signature xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
            Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#'>
            <cl4n:InclusiveNamespaces
              xmlns:cl4n='http://www.w3.org/2001/10/xml-exc-c14n#'
              PrefixList='wsse wsu soap' />
          </ds:CanonicalizationMethod>
          <ds:SignatureMethod
            Algorithm='http://www.w3.org/2000/09/xmldsig#rsa-sha1' />
          <ds:Reference URI=''>
```

```

<ds:Transforms>
  <dsig:XPath xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
              xmlns:m="http://example.org/ws">
    //soap:Body/m:SomeElement
  </dsig:XPath>
  <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
    <c14n:InclusiveNamespaces
        xmlns:c14n="http://www.w3.org/2001/10/xml-exc-c14n#"
        PrefixList="soap wsu test" />
  </ds:Transform>
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
<ds:DigestValue>VEPKwzfPG0xh2OUpoK0bcl58jtU=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>+diIuEyDpV7qxVoUOkb5rj61+Zs=</ds:SignatureValue>
<ds:KeyInfo>
  <wsse:SecurityTokenReference>
    <wsse:Reference URI="#SomeCert" />
  </wsse:SecurityTokenReference>
</ds:KeyInfo>
</ds:Signature>
</wsse:Security>
</soap:Header>
<soap:Body xmlns:wsu="http://docs.oasis-open.org/..."
            xmlns:test="http://www.test.com" wsu:Id='TheBody'>
  <m:SomeElement xmlns:m="http://example.org/ws" attr1="test:fdwfde" />
</soap:Body>
</soap:Envelope>

```

Indent:

Select this method to ensure that the generated signature is properly indented.

Create Enveloped Signature:

By selecting this option, an enveloped XML Signature is generated. The following skeleton signed SOAP message shows the enveloped signature

```

<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" id="Sample">
  <ds:SignedInfo>
    <ds:Reference URI="">
      <ds:Transforms>
        <ds:Transform
            Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
      </ds:Transforms>
    </ds:Reference>
  </ds:SignedInfo>
</ds:Signature>

```

This indicates to the application validating the signature that the signature itself should not be included in the signed

data. In other words, to validate the signature, the application must first strip out the signature. This is necessary in cases where the entire SOAP envelope has been signed, and the resulting signature has been inserted into the SOAP header. In this case, the signature is over a nodeset which has been altered (i.e. the Signature has been inserted), and so the signature will break.

Insert CarriedKeyName for EncryptedKey:

Check this option to include a <CarriedKeyName> element in the <EncryptedKey> block that is generated when using a symmetric signing key.

XML Signature Verification

Overview

In addition to validating XML Signatures for authentication purposes, the Enterprise Gateway can also use XML Signatures to prove message integrity. By signing an XML message, a client can be sure that any changes made to the message do not go unnoticed by the Enterprise Gateway. Therefore by validating the XML Signature on a message, the Enterprise Gateway can guarantee the *integrity* of the message.

Before configuring the **XML Signature Verification** filter, enter an appropriate name for this filter in the **Name** field.

Signature Verification

The following sections are available on the **Signature Verification** tab:

Signature Location:

Because there may be multiple signatures contained in the message, you must specify which signature the Enterprise Gateway uses to verify the integrity of the message. The signature can be extracted from one of the following:

- From the SOAP header
- Using WS-Security Actors
- Using XPath

Select the appropriate option from the drop-down list.

For details on all the configuration options available in this section, see the [Signature Location](#) topic.

Find Signing Key

The public key used to verify the signature can be taken from the following locations:

- **Via KeyInfo in Message:**
Typically, a `<KeyInfo>` block is used in an XML Signature to reference the key used to sign the message. For example, it is common for a `<KeyInfo>` block to reference a `<BinarySecurityToken>` that contains the certificate associated with the public key used to verify the signature.
- **Via Message Attribute:**
The certificate used to verify the signature can be extracted from a message attribute. For example, a previous filter (for example, a **Find Certificate** filter) may have already located a certificate and populated the `certificate` message attribute. If you wish to use this certificate to verify the signature, specify this attribute in the field provided.
- **Via Certificate in LDAP:**
Clients may not always want to include their public keys in their signatures. In such cases, the public key can be retrieved from a specified LDAP directory. To do this, select the **Via Certificate in LDAP** radio button, and select a previously configured LDAP directory from the drop-down list. You can add LDAP connections on the **External Connections** tab. Right-click the **LDAP Connection** tree node, and select **Add an LDAP Connection**.
- **Via Certificate in Store:**
Similarly, you can retrieve a certificate from the Certificate Store by selecting this option, and clicking the **Select** button. Select the checkbox next to the certificate that contains the public key that you want to use to verify the signature, and click **OK**.

What Must Be Signed

This section defines the content that must be signed for a SOAP message to pass the filter. This ensures that the client

has signed something meaningful (part of the SOAP message), instead of arbitrary data that would pass a blind signature validation. This further strengthens the integrity verification process.

The nodeset that must be signed can be identified by a combination of XPath expressions, node locations, and/or the contents of a message attribute. For more details on how to configure this section, see the [What Must Be Signed](#) topic.

Note:

If all attachments are required to be signed, select the **All attachments** checkbox to enforce this.

Advanced

The following advanced configuration options are available:

Signature Confirmation:

If this filter is configured as part of an Initiator circuit, where the Enterprise Gateway acts as the client in a Web Services transaction, select the **Initiator** option. This means that the filter keeps a record of the Signature that it has verified, and checks the <SignatureConfirmation> returned by the Recipient.

Alternatively, if the Enterprise Gateway acts as the Recipient in the transaction, select the **Recipient** option. In this case, the Enterprise Gateway returns the <SignatureConfirmation> elements in the response to the Initiator.

Default Derived Key Label:

If the Enterprise Gateway consumes a <DerivedKeyToken>, the default value entered is used to recreate the derived key.

Algorithm Suite:

Select the WS-Security Policy *Algorithm Suite* that must have been used when signing the message. This check ensures that the appropriate algorithms were used to sign the message.

Fail if No Signatures to Verify:

Select this option if you want to configure the filter to fail if no XML Signatures are present in the incoming message.

Verify Signature for Authentication Purposes:

You can use the **XML Signature Verification** filter to authenticate an end user. If the message can be successfully validated, it proves that only the private key associated with the public key used to verify the signature was used to sign the message. Because the private key is only accessible to its owner, a successful verification can be used to effectively authenticate the message signer.

Message Attribute Containing DOM:

You can configure this field to verify the response from a SAML PDP. When the Enterprise Gateway receives a response from the SAML PDP, it stores the signature on the response in a message attribute. You can select this attribute from the drop-down list to verify this signature.

Remove enclosing WS-Security element on successful verification:

Select this checkbox if you wish to remove the enclosing WS-Security block when the signature has been successfully verified. This setting is not selected by default.

SMIME Sign

Overview

You can use the **SMIME Sign** filter to digitally sign a multipart message as it passes through the Enterprise Gateway core pipeline. The recipient of the message can then verify the integrity of the SMIME message by validating the Public Key Cryptography Standards (PKCS) #7 signature.

Configuration

Complete the following fields to configure this filter:

Name:

Enter an appropriate name for the filter.

Sign Using Key:

Select the checkbox next to the certificate that contains the public key associated with the private signing key that you wish to use to sign the message.

Create Detached Signature in Attachment:

Specifies whether to create a detached digital signature in the message attachment. This is selected by default. For example, this is useful when the software reading the message does not understand the PKCS#7 binary structure, because it can still display the signed content, but without verifying the signature.

If this is unselected, the message content is embedded with the PKCS#7 binary signature. This means that user agents that do not understand PKCS#7 can not display the signed content. Intermediate systems between the sender and final recipient may modify the text content slightly (for example, line wrapping, whitespace, or text encoding). This may cause the message to fail signature validation due to changes in the signed text that are not malicious, nor necessarily affecting the meaning of the text.

SMIME Verify

Overview

You can use the **SMIME Verify** filter to check the integrity of a Secure/Multipurpose Internet Mail Extensions (SMIME) message. This filter enables you to verify the Public Key Cryptography Standards (PKCS) #7 signature over the message.

You can select the certificates that contain the public keys that you wish to use to verify the signature. Alternatively, you can specify a message attribute that contains the certificate with the public key that you wish to use.

Configuration

Complete the following fields to configure this filter:

Name:

Enter an appropriate name for the filter.

Use the following certificates:

Select the certificates that contain the public keys that you wish to use to verify the signature. This is the default option.

Certificate in attribute:

Alternatively, enter the message attribute that specifies the certificate that contains the public key that you wish to use to verify the signature. Defaults to the `certificate` message attribute.

Remove Outer Envelope if Verification is Successful:

Select this option if you want to remove the PKCS#7 signature and all its associated data from the message if it verifies successfully.

Logging Configuration

Overview

One of the most important features of a server-based product is its ability to maintain highly detailed and configurable logging. It is crucial that a record of each and every transaction is kept, and that these records can easily be queried by an administrator to carry out detailed transaction analysis. In recognition of this requirement, the Enterprise Gateway provides detailed logging to a number of possible locations.

You can configure the Enterprise Gateway so that it logs information about all requests. Such information includes the request itself, the time of the request, where the request was routed to, and the response that was returned to the client. The logging information can be written to the console, log file, local/remote syslog, and/or a database, depending on what is configured in the **Configure Logging** dialog.

The Enterprise Gateway can also digitally sign the logging information it sends to the log files and the database. This means that the logging information can not be altered after it has been signed, thus enabling an irreversible audit trail to be created.

Configuring Log Output

To edit the default logging settings that ship with the Enterprise Gateway, in the Policy Studio main menu, select **Settings** -> **Settings** -> **Default Logging Settings**. Alternatively, in the toolbar, click the drop-down option on the **Settings** button, and select **Default Logging Settings**.

In addition, at the Process level, you can override the default log settings by right-clicking the Process in the Policy Studio tree, and selecting **Logging** -> **Custom**.

Whether editing or customizing the default logging settings, you can use the **Configure Logging** tabbed dialog to configure the Enterprise Gateway to log to the following locations.

Log to Text File

To configure the Enterprise Gateway to log in text format to a file, click the **Text File** tab, and select the **Enable logging to file** checkbox. You can configure the following fields:

- **File Name:**
Enter the name of the text-based file that the Enterprise Gateway logs to. By default, the log file is called `oracle`.
- **File Extension:**
Enter the file extension of the log file in this field. By default, this has a `.log` extension.
- **Directory:**
Enter the directory of the log file in this field. By default, all log files are stored in the `/logs` directory of your Enterprise Gateway installation.
- **File Size:**
Enter the maximum size that the log file grows to. When the file reaches the specified limit, a new log file is created. By default, the maximum file size is 1000 kilobytes.
- **Roll Log Daily:**
Specify whether to roll over the log file at the start of each day. This is enabled by default.
- **Number of Files:**
Specify the number of log files that are stored. The default number is 20.
- **Format:**
You can specify the format of the logging output using the values entered here. You can use properties to output logging information that is specific to the request. The default logging format is as follows, with descriptions of the available logging properties below:

```
${level} ${timestamp} ${id} ${text} ${filterType} ${filterName}
```

- **level:**
The log level (fatal, fail, success).
- **timestamp:**
The time that the message was processed in user-readable form.
- **id:**
The unique transaction ID assigned to the message.
- **text:**
The text of the log message that was configured in the filter itself. In the case of the **Log Message Payload** filter, the `${payload}` property contains the message that was sent by the client.
- **filterName:**
The name of the filter that generated the log message.
- **filterType:**
The type of the filter that logged the message.
- **ip:**
The IP address of the client that sent the request.
- **Signing Key:**
To sign the log file, select a **Signing Key** from the Certificates Store that is used in the signing process. By signing the log files, you can verify their integrity at a later stage.

Log to XML File

To configure the Enterprise Gateway to log to an XML file, click the **XML File** tab, and select the **Enable logging to XML file** checkbox.

The log entries are written as the values of XML elements in this file. You can view historical XML log files (not the current file) as HTML for convenience by opening the XML file in your default browser. The `/logs/xsl/MessageLog.xsl` stylesheet is used to render the XML log entries in a more user-friendly HTML format.

You can configure the following fields on the **XML File** tab:

- **File Name:**
Enter the name of the text-based file that the Enterprise Gateway logs to. By default, the log file is called `oracle`.
- **File Extension:**
Enter the file extension of the log file in this field. By default, the log file is given the `.log` extension.
- **Directory:**
Enter the directory of the log file in this field. By default, all log files are stored in the `/logs` directory of your Enterprise Gateway installation.
- **File Size:**
Enter the maximum size that the log file grows to. When the file reaches the specified limit, a new log file is created. By default, the maximum file size is 1000 kilobytes.
- **Roll Log Daily:**
Specify whether to roll over the log file at the start of each day. This is enabled by default.
- **Number of Files:**
Specify the number of log files that are persisted. The default number is 20.
- **Signing Key:**
To sign the log file, select a **Signing Key** from the Certificates Store that will be used in the signing process. By

signing the log files, you can verify their integrity at a later stage.

Log to Database

Using this option, you can configure the Enterprise Gateway to log messages to an Oracle, SQL Server, or MySQL relational database. Before configuring the Enterprise Gateway to log to a database, you must first create the tables that the Enterprise Gateway writes to. The SQL commands required to set up these tables for each of these databases can be found under the `INSTALL_DIR/system/conf/sql`. This folder contains a directory for each of the Oracle, SQL Server, and MySQL databases, each of which contains the appropriate SQL scripts.

The SQL commands to generate the database table can be found in the `audit_trail.sql` file. Select this file from the appropriate directory (depending on your database), and use the tool of your choice to run the SQL commands contained in the file. For example, to create the logging tables in a MySQL database, you can simply copy and paste the SQL commands into the MySQL command prompt.

When you have set up the logging database tables, you can configure the Enterprise Gateway to log to the database. To do this, click the **Database** tab on the **Configure Logging** dialog, and select the **Enable logging to database** checkbox. You can configure the following fields on the **Database** tab:

- **Connection:**
Select an existing database from the **Connection** drop-down list. To add a database connection, click the **External Connections** button on the left, right-click the **Database Connections** tree node, and select **Add a Database Connection**. For more details, see the [Database Connection](#) topic.
- **Signing Key:**
You can sign log messages stored in the database to ensure that they are not tampered with. Click the **Signing Key** button to open the list of certificates in the Certificate Store. You can then select the key to use to sign log messages.

Log to Local Syslog

To configure the Enterprise Gateway to send logging information to the local UNIX syslog, click the **Local Syslog** tab, and select the **Enable logging to local UNIX Syslog** checkbox. You can configure the following fields:

- **Facility:**
Select the local syslog facility that the Enterprise Gateway should log to. The default is `LOCAL0`.
- **Format:**
You can specify the format of the log message using the values (including properties) entered in this field. For details on the properties that are available, see [Log to Text File](#).

Log to Remote Syslog

To configure the Enterprise Gateway to send logging information to a remote syslog, click the **Remote Syslog** tab, and select the **Enable logging to Remote Syslog** checkbox. You can configure the following fields:

- **Syslog Server**
Select a previously configured **Syslog Server** from the drop-down list.
- **Format:**
You can specify the format of the log message using the values (including properties) entered in this field. For details on the properties that are available, see [Log to Text File](#).

Log to System Console

To configure the Enterprise Gateway to send logging information to the system console, click the **System Console** tab, and select the **Enable logging to system console** checkbox. For details on how to use the **Format** field to configure the format of the log message, see [Log to Text File](#).

Log Level and Message

Overview

By default, logging is configured for a Web Service with logging level of failure. You can also configure each filter in a policy to log its own message depending on whether it succeeds, fails, and/or throws an exception. Log messages can be stored in several locations, including a database, a file, or the system console. For more details on configuring logging destinations, see the [Logging Configuration](#) topic.

Logging levels apply to the following cases:

- A filter succeeds if it returns a true result after carrying out its processing. For example, if an LDAP directory returns an `authorized` result to an authorization filter, the filter succeeds.
- A filter fails if it returns a false result after performing its processing. For example, an authorization filter returns false if an LDAP directory returns a `not authorized` result to the filter.
- A filter aborts when it can not make the decision it is configured to make. For example, if an LDAP-based authorization filter can not connect to the LDAP directory, it aborts because it can neither authorize nor refuse access. This is regarded as a *fatal* error.

Configuration

You can access the **Log Level and Message** configuration screen by clicking the **Next** button on the main screen of all filters. This screen includes the following fields:

Log Level:

Configure one of the following options:

Use Service Level Settings	This option is selected by default. Logging is configured for the Web Service with logging level of Failure .
Override Logging Level for this Filter	Alternatively, select this option to configure log messages for this filter when it succeeds, fails, and/or aborts. Select Success , Failure , and/or Fatal to configure this filter to log at the respective levels.

Log Messages:

Default log message values are provided at each level for all filters. When you select the checkbox for a particular level, the default log message for that level is used. You can specify an alternative log message by entering the message in the text field provided.

All filters *require* and *generate* message attributes, while some *consume* attributes. In some cases, it may be useful to log the value of these attributes. For example, instead of an authentication filter logging a generic `Authentication Failed` message, you can use the value of the `authentication.subject.id` attribute to log the ID of the user that could not be authenticated.

Use the following format to enter a message attribute as a property in a log message:

`${name_of_attribute}`

At runtime, the Enterprise Gateway expands these properties to the value of the message attribute. For example, to make sure the ID of a non-authenticated user is logged in the message, enter something like the following in the text field for the **Failure** case:

```
The user '${authentication.subject.id}' could not be authenticated.
```

Then if a user with ID `oracle` can not be authenticated by the Enterprise Gateway (a failure case), the following message is logged:

```
The user 'oracle' could not be authenticated.
```

Audit Logging Behavior:

This setting is relevant only in cases where you have configured the Enterprise Gateway to log audit trail messages to a database. For more details, see the instructions in the [Logging Configuration](#) topic.

You can select the **Abort circuit processing on database log error** checkbox if you have configured the Enterprise Gateway to write log messages to a database, but that database is not available at runtime. If you have selected this checkbox, and the database is not available, the filter aborts, which in turn causes the circuit to abort. In this case, the Fault Handler for the circuit is invoked.

Filter Category:

The category selected here identifies the category of filters to which this filter belongs. The default selection should be appropriate in most cases.

Log Message Payload

Overview

The **Log Message Payload** filter is used to log the message payload at any point in the policy. The message payload includes the HTTP headers and MIME/DIME attachments.

By placing the **Log Message Payload** filter at various key locations in the policy, a complete audit trail of the message can be achieved. For example, by placing the filter after each filter in the policy, the complete history of the message can be logged. This is especially useful in cases where the message has been altered by the Enterprise Gateway (for example, by signing or encrypting the message, inserting security tokens, or by converting the message to another grammar using XSLT).

Log messages can be stored in several locations, including a database, a file, or the system console.

Configuration

Enter an appropriate name for the **Log Message Payload** filter in the **Name** field. It is good practice to use descriptive names for these filters. For example, **Log message before signing message** and **Log message after signing** would be useful names to give to two **Log Message Payload** filters that are placed before and after a **Sign Message** filter.

By default, the **Log Message Payload** filter writes entries to the log file in the following format:

```
${timestamp} ${id} ${filterName} ${payload}
```

However, you can alter the format of the logging output using the values entered in the **Format** field. You can use properties to output logging information that is specific to the request. You can use the following properties:

- **level:**
The log level (i.e. fatal, fail, success).
- **id:**
The unique transaction ID assigned to the message.
- **ip:**
The IP address of the client that sent the request.
- **timestamp:**
The time that the message was processed in user-readable form.
- **filterName:**
The name of the filter that generated the log message.
- **filterType:**
The type of the filter that logged the message.
- **text:**
The text of the log message that was configured in the filter itself.
- **payload:**
The complete contents of the HTTP request, including HTTP headers, body, and attachments.

Log Access Filter

Overview

The **Log Access Filter** is used by the Enterprise Gateway to log records of all messages that pass through the filter.

The Enterprise Gateway writes the access log to an `access.log` file in the `log` directory. This file rolls over at the start of the day so that the name of the log file incorporates the date that the log file was created (for example, `access_30Apr2010.log`).

The format of the log entries is Common Log Format, which has the following format:

```
host ident authuser date request status bytes
```

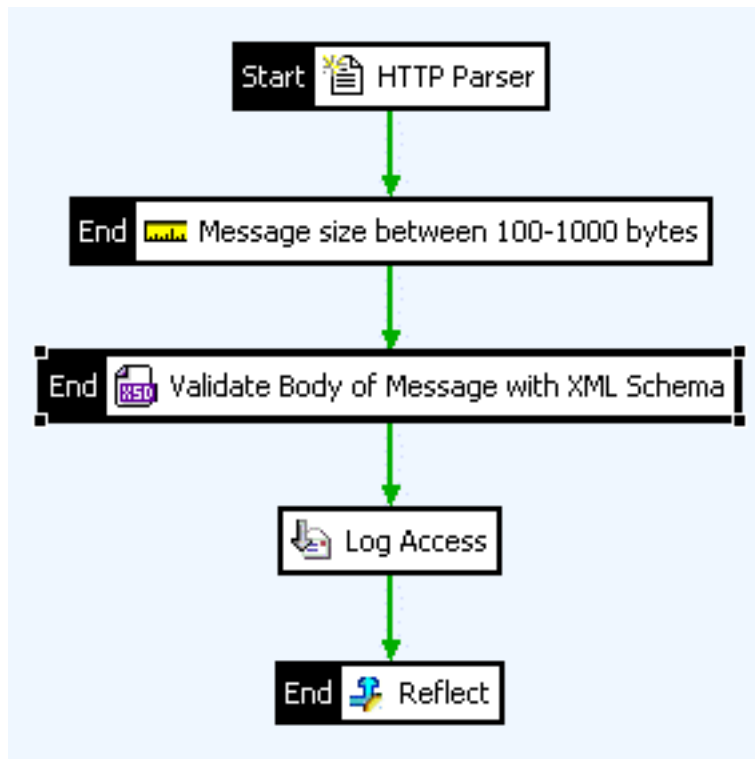
The following list explains each item:

- **host:** The remote hostname.
- **ident:** The remote logname of the user.
- **authuser:** The username by which the user has authenticated himself (for example, the Distinguished Name of a certificate).
- **date:** The date and time of the request.
- **request:** The request line exactly as it originated at the client.
- **status:** The HTTP status code returned to the client.
- **bytes:** The content-length of the document returned to the client.

The following extract from the `access.log` file illustrates the format:

```
m1.oracle.com - Good [30/Mar/2009:22:09:05 00] "http://services/qotd" 200 587
m3.oracle.com - Good [30/Mar/2009:22:10:34 00] "http://services/qotd" 200 671
m1.oracle.com - Good [30/Mar/2009:22:10:53 00] "http://services/qotd" 200 571
.....
.....
.....
```

Because the **Log Access** filter reports the number of bytes returned to the client (the `bytes` parameter explained above), it should be positioned towards the end of a policy. A typical policy involving a **Log Access** filter might appear as follows:



Configuration

The **Log Access Filter** requires only a **Name** field to be configured.

Service Level Agreement (SLA) Filter

Overview

A Service Level Agreement (SLA) is an agreement put in place between a Web Services host and a client of that Web Service in order to guarantee a certain minimum quality of service. It is common to see SLAs in place to ensure that a minimum number of messages result in a communications failure and that responses are received within an acceptable timeframe. In cases where the conditions of the SLA are breached, it is crucial that an alert can be sent to the appropriate party.

The Enterprise Gateway satisfies these requirements by allowing SLAs to be configured at the policy level. It is possible to configure SLAs to monitor the following types of problems:

- Response times
- HTTP status codes returned from the Web Service
- Communication failures

It is important to note that the SLA monitoring performed by the Enterprise Gateway is *statistical*. Because of this, a single message (or even a small number of messages) is not considered a sufficient sample to cause an alert to be triggered. The monitoring engine actually uses an *exponential decay* algorithm to determine whether an SLA is "failing" or not. This algorithm is best explained with an example.

Assume the *poll rate* is set to 3 seconds (i.e. 3000ms), the *data age* is set to 6 seconds (i.e. 6000ms), and you have a Web Service with an average processing time of 100ms. A single client sending a stream of requests through the Enterprise Gateway will be able to generate about 10 requests per second, given the Web Services's 100ms response time.

At every 3 seconds poll period you will have data from a previous 30 samples to consider the average response times of. However, rather than simply using the response time of the *last* 3 seconds worth of data, historical data is "smoothed" into the current estimate of the failing percentage. The new data is combined with the existing data such that it will take approximately the data age time for a sample to disappear from the average.

Therefore the closer the data age is to the sampling rate, the less significant historical data becomes, and the more significant the "last" sample becomes.

In order to generate an alert, you must also have enough significant samples at each poll period to consider the data to be statistically valid. For example, if a single request arrives over a period of 1 hour it may not be fair to say that "less than 20%" of all received requests have failed the response time requirements. For this reason, statistical analysis provides a more realistic SLA monitoring mechanism than a solution based purely on absolute metrics.

Response Time Requirements

You can monitor the response times of Web Services protected by the **SLA Filter**. This filter provides different ways of measuring response times:

<i>Response Time Measurement</i>	<i>Description</i>
receive-request-start	The time that the Enterprise Gateway receives the first byte of the request from the client.
receive-request-end	The time that the Enterprise Gateway receives the last byte of the request from the client.
send-request-start	The time that the Enterprise Gateway sends the first byte of the request to the Web Service.
send-request-end	The time that the Enterprise Gateway sends the last byte of

<i>Response Time Measurement</i>	<i>Description</i>
	the request to the Web Service
receive-response-start	The time that the Enterprise Gateway receives the first byte of the response from the Web Service
receive-response-end	The time that the Enterprise Gateway receives the last byte of the response from the Web Service.
send-response-start	The time that the Enterprise Gateway sends the first byte of the response to the client.
send-response-end	The time that the Enterprise Gateway sends the last byte of the response to the client.

The Enterprise Gateway will measure each of the 8 time values. They will be available for processing after the policy has completed for a single request. These 8 options are available for the following reasons:

- The Enterprise Gateway may start to send the first byte to the Web Service before the last byte is received from the client, i.e. send-request-start < receive-request-end. This will occur if the invoked policy does not require the full message to be read into memory.
- The Enterprise Gateway may start to send the response to the client before the complete response has been received from the Web Service, i.e. send-response-start < receive-response-end. This will occur when invoked policy does not require the full message to be read into memory.
- It is possible that the Web Service may start to send the response before it has received the complete request. However, the Enterprise Gateway will not start to read the response until it has sent the complete request. This means that the following is always true:- send-request-end < receive-response-start.
- The time value for send-response-end will depend upon the client application. This value will be larger if the client is slow to read the response.

To add a Response Time Requirement for an SLA, click the **Add** button.

To configure the start time and end time for the response time measurement, click the **Add** button. On the **Settings** tab, specify the percentage of response times that must be below a specified time interval (in milliseconds) in the fields provided. The purpose of these options is to allow for situations where a very small number of unusually slow requests may cause an SLA to trigger unnecessarily. By using percentages, such requests will not distort the statistics collected by the Enterprise Gateway.

Click on the **Message Text** tab to configure the messages that will appear in the alert message when the SLA is breached and also when the SLA is cleared, i.e. when the breached conditions are no longer in breach of the SLA.

Finally, click on the **Advanced** tab to configure timing information. Select a **Start Timing Point** from the 8 times listed in the table above. The Enterprise Gateway will *start* measuring the response time from this time. Then select an **End Timing Point** from the 8 times listed in the table above. The Enterprise Gateway will *stop* measuring the response time from this time.

HTTP Status Requirements

HTTP status codes may be received from a Web Service. The Enterprise Gateway can be configured to monitor these and generate alerts based on the number of occurrences of certain types of status code response. HTTP status codes are three digit codes that may be grouped into standard status "classes", with the first digit indicating the status class. The status classes are as follows:

HTTP Status Code Class	Description
1xx	These status codes indicate a provisional response.
2xx	These status codes indicate that the client's request was successfully received, understood, and accepted.
3xx	These status codes indicate that further action needs to be taken by the user agent in order to fulfill the request.
4xx	These status codes are intended for cases in which the client seems to have erred. For example 401, means that authentication has failed.
5xx	These status codes are intended for cases where the server has encountered an unexpected condition that prevented it from fulfilling the request. For example, 500 is used to transmit SOAP faults.

The Enterprise Gateway may monitor a class (i.e. range) of status codes, or they may monitor specific status codes. For example, it is possible to configure the following HTTP status code requirements:

- At least 97% of the requests must yield HTTP status codes between 200 and 299
- At most 2% of requests may yield HTTP status codes between 400 and 499
- At most 0% of requests may yield HTTP status code 500

Click the **Add** button in the **HTTP Status Code Requirements** section.

Select an existing status code or class of status codes from the **HTTP Status Code** dropdown. To add a new code or range of codes, click the **Add** button.

Enter a name for the new code or range of codes in the **Name** field of the **Configure HTTP Status Code** dialog. Enter the *first* HTTP status code in the range of status codes that you want to monitor in the **Start Status** field. Then enter the *last* HTTP status code in the range of status codes that you want to monitor in the **End Status** field.

If you just want to monitor one specific status code, enter the same code in the **Start Status** and **End Status** fields.

Click **OK** when you are satisfied with the selected range of status codes to return to the previous dialog. The remaining 2 fields allow the administrator to specify the minimum or maximum percentage of received HTTP status codes that fall into the configured range before an alert is triggered.

Again, the use of percentages here is to allow for situations where a very small number of requests return the status codes within the "forbidden" range. By using percentages, such requests will not distort the statistics collected by the Enterprise Gateway.

Click on the **Message Text** tab to configure the messages that will appear in the alert message when the SLA is breached and also when the SLA is cleared, i.e. when the breached conditions are no longer in breach of the SLA.

Communications Failure Requirements

The Enterprise Gateway is deemed to have experienced a *communications failure* when it fails to connect to the Web Service, fails to send the request, or fails to receive the response.

The requirements for communications failures may be expressed as follows:

- No more than 4% of requests may result in communications failures.

Enter the percentage of allowable communications failures in the field provided. An alert will be configured if the percentage of communicates failures rises above this level.

Click on the **Message Text** tab to configure the messages that will appear in the alert message when the SLA is breached and also when the SLA is cleared, i.e. when the breached conditions are no longer in breach of the SLA.

Select Alerting System

If an alert is triggered, it must be sent to an alerting destination. The Enterprise Gateway can send alerts to the following destinations:

- Windows Event Log
- Email Recipient
- SNMP Network Management System
- Unix Syslog
- CheckPoint's FireWall-1

The **Alert System** table at the bottom of the **SLA Filter** screen displays all available alerting destinations that have been configured through the alerting interface .

The administrator should select one or more alerting systems. An alert will be sent to each selected system in the event of a violation of the performance requirements. Alert clearances will be generated when the violation no longer exists.

Set Service Name

Overview

The **Set Service Name** filter configures service-level monitoring details. For example, you can use the fields on this filter screen to configure whether the Enterprise Gateway stores service usage and service usage per client details. You can also set the name of the service displayed when generating reports with the Service Monitor web-based reporting and monitoring tools.

Configuration

Name:

Enter an appropriate name for the filter to be displayed in a circuit.

Service Name:

Enter an appropriate name for this service to be displayed on the Service Monitor interface when generating reports for this service.

Monitoring Options:

The fields in this group enable you to configure whether this Web Service store usage metrics data to a database. This information can be used by Service Monitor to produce reports showing how and who is calling this Web Service. The following fields are available:

- **Monitor service usage:**
Select this option if you want to store message metrics for this Web Service.
- **Monitor service usage per client:**
Select this option if you want to generate reports monitoring which authenticated clients are calling which Web Services.
- **Monitor client usage:**
If you want to generate reports on authenticated clients, but are not interested in which Web Services they are calling, select this option and unselect the **Monitoring service usage per client** checkbox.
- **Which attribute is used to identify the client?:**
Enter the message attribute to use to identify authenticated clients. The default is `authentication.subject.id`, which stores the identifier of the authenticated user (for example, the username or user's X.509 Distinguished Name).

Oracle Access Manager Authorization

Overview

This filter enables you to authorize an authenticated user for a particular resource against Oracle Access Manager (OAM). The user must first have been authenticated to OAM using the [HTTP Basic](#) or [HTTP Digest](#) filter. After successful authentication, OAM issues a Single Sign On (SSO) token, which can then be used instead of the user name and password.

Configuration

Configure the following fields to authorize a user for a particular resource against Oracle Access Manager:

Name:

Enter a descriptive name for this filter.

Attribute Containing SSO Token:

Enter the name of the message attribute that contains the user's SSO token. This attribute will have been populated when authenticating to Oracle Access Manager using the [HTTP Basic](#) or [HTTP Digest](#) filter. By default, the SSO token is stored in the `oracle.sso.token` message attribute.

Resource Type:

Enter the type of the resource for which you are requesting access. For example, when seeking access to a Web-based URL, specify `http`.

Resource Name:

Enter the name of the resource for which the user is requesting access. By default, this field is set to `/hostname${http.request.uri}`, which contains the original path requested by the client.

Operation:

In most access management products, it is common to authorize users for a limited set of actions on the requested resource. For example, users with management roles may be able to write (HTTP POST) to a certain Web Service, but users with more junior roles might only have read access (HTTP GET) to the same service.

You can use this field to specify the operation that you want to grant the user access to on the specified resource. By default, this field is set to the `http.request.verb` message attribute, which contains the HTTP verb used by the client to send the message to the Enterprise Gateway (for example, POST).

Logout from Oracle Access Manager SSO Session

Overview

This filter enables you to log out a session from Oracle Access Manager by invalidating the SSO token that is associated with this session.

Configuration

Configure the following fields to explicitly log out (invalidate) an SSO token from Oracle Access Manager:

Name:

Enter a descriptive name for this filter.

Attribute Containing SSO Token ID:

Enter the name of the message attribute that contains the SSO token that you want to validate. This attribute will have been populated when authenticating to Oracle Access Manager using the [HTTP Basic](#) or [HTTP Digest](#) filter. By default, the SSO token is stored in the `oracle.sso.token` message attribute.

Oblix Installation Directory:

Enter the location of your Oblix installation directory. For more details on Oblix, see your Oracle Access Manager documentation.

Oracle Access Manager SSO Token Validation

Overview

This filter enables you to check an Oracle Access Manager Single Sign On (SSO) token to ensure that it is still valid. The SSO token is issued by Oracle Access Manager (OAM) after the Enterprise Gateway authenticates to it on behalf of an end-user using the [HTTP Basic](#) or [HTTP Digest](#) filter. After successfully authenticating to OAM, the SSO token is stored in the `oracle.sso.token` message attribute.

Oracle Access Manager SSO enables a client to send up its user name and password once, and then receive an SSO token (for example, in a cookie or in the XML payload). The client can then send up the SSO token instead of the user name and password.

Configuration

Configure the following fields to validate an SSO token issued by Oracle Access Manager:

Name:

Enter a descriptive name for the filter.

Attribute Containing SSO Token ID:

Enter the name of the message attribute that contains the SSO token that you want to validate. This attribute will have been populated when authenticating to Oracle Access Manager using the [HTTP Basic](#) or [HTTP Digest](#) filters. By default, the SSO token is stored in the `oracle.sso.token` message attribute.

Oblix Installation Directory:

Enter the location of your Oblix installation. For more details on Oblix, see your Oracle Access Manager documentation.

Oracle Entitlements Server Authorization

Overview

This filter enables you to authorize an authenticated user for a particular resource against Oracle Entitlements Server (OES). The user must first have been authenticated to OES (for example, using the [HTTP Basic](#) or [HTTP Digest](#) filter).

This filter enables you to configure the Enterprise Gateway to delegate authorization to Oracle Entitlements Server. You can configure the Enterprise Gateway to authorize an authenticated user for a particular resource against the Oracle Entitlements Server. Credentials used for authentication can be extracted from the HTTP Basic header, WS-Security username token, or the message payload. After successful authentication, the Enterprise Gateway can authorize the user to access a resource using the Oracle Entitlements Server.

General

Configure the following general field:

Name:

Enter an appropriate descriptive name for this filter.

Settings

Configure the following fields on the **Settings** tab:

Resource:

Enter the URL for the target resource (for example, Web Service). Alternatively, if this policy is reused for multiple services, enter a URL using message attribute properties, which are expanded at runtime to the value of the specified attribute. For example:

```
${http.destination.protocol}://${http.destination.host}:${http.destination.port}${http.request.u
```

Resource Naming Authority:

Enter `gatewayResource` to match the Naming Authority Definition loaded in the Oracle Entitlements Server settings. For more details, see [Oracle Security Service Module Settings](#).

Action:

Enter the HTTP verb (for example, POST, GET, DELETE, and so on). Alternatively, if this policy is reused for multiple services, enter a message attribute property, which is expanded at runtime to the value of the specified attribute (for example, `${http.request.verb}`).

Action Naming Authority:

Enter `gatewayAction` to match the Naming Authority Definition loaded in the Oracle Entitlements Server settings. For more details, see [Oracle Security Service Module Settings](#).

How access request is processed:

Select one of the following options:

ONCE	Specifies that the authorization query is only asked once for a resource and action.
POST	Specifies that the authorization query is asked after a resource is acquired, but before it has been processed or

	presented.
PRIOR	Specifies that the authorization query is asked before a resource is acquired.

Application Context

Configure the following field on the **Application Context** tab:

Application's Current Context:

Click **Add** to specify optional Application Contexts as name-value pairs. Enter a **Name** and **Value** in the **Properties** dialog. Repeat to specify multiple properties.

Get Roles from Oracle Entitlements Server

Overview

This filter enables you to get the set of roles that are assigned to an identity for a specific resource (for example, Web Service) and a specific action (for example, HTTP POST) from Oracle Entitlements Server (OES).

General

Configure the following general field:

Name:

Enter an appropriate descriptive name for this filter.

Settings

Configure the following fields on the **Settings** tab:

Resource:

Enter the URL of the target resource (for example, Web Service). Alternatively, if this policy is reused for multiple services, enter a URL using message attribute properties, which are expanded at runtime to the value of the specified attribute. For example:

```
${http.destination.protocol}://${http.destination.host}:${http.destination.port}${http.request.u
```

Resource Naming Authority:

Enter `gatewayResource` to match the Naming Authority Definition loaded in the Oracle Entitlements Server settings. For more details, see [Oracle Security Service Module Settings](#).

Action:

Enter the HTTP verb (for example, POST, GET, DELETE, and so on). Alternatively, if this policy is reused for multiple services, enter a message attribute property, which is expanded at runtime to the value of the specified attribute (for example, `${http.request.verb}`).

Action Naming Authority:

Enter `gatewayAction` to match the Naming Authority Definition loaded in the Oracle Entitlements Server settings. For more details, see [Oracle Security Service Module Settings](#).

Application Context

Configure the following field on the **Application Context** tab:

Application's Current Context:

Click **Add** to specify optional Application Contexts as name-value pairs. Enter a **Name** and **Value** in the **Properties** dialog. Repeat to specify multiple properties.

Relative Path Resolver

Overview

The **Relative Path** filter enables you to identify an incoming XML message based on the relative path on which the message is received.

The following example shows how to find the relative path of an incoming message. Consider the following SOAP message:

```
POST /services/helloService HTTP/1.1
Host: localhost:8095
Content-Length: 196
SOAPAction: HelloService
Accept-Language: en-US
UserAgent: Enterprise Gateway
Content-Type: text/XML; utf-8

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <getHello xmlns="http://www.oracle.com/" />
  </soap:Body>
</soap:Envelope>
```

The relative path for this message is as follows:

Relative Path
/services/helloService

Configuration

To configure the **Relative Path** filter, complete the following:

1. Enter an appropriate name for the filter in the **Name** field.
2. Enter a regular expression to match the value of the relative path on which messages are received in the **Relative Path** field. For example, enter `^/services/helloService$` to exactly match a path with a value of `/services/helloService`. Incoming messages received on a matching relative path value are passed on to the next filter on the success path in the policy.

SOAPAction Resolver

Overview

The **SOAPAction** filter enables you to identify an incoming XML message based on the SOAPAction HTTP header in the message. The **SOAPAction** filter applies to SOAP 1.1 and SOAP 1.2.

The following example illustrates how to locate the SOAPAction header in an incoming message. Consider the following SOAP message:

```
POST /services/helloService HTTP/1.1
Host: localhost:8095
Content-Length: 196
SOAPAction: HelloService
Accept-Language: en-US
UserAgent: Enterprise Gateway
Content-Type: text/XML; utf-8

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <getHello xmlns="http://www.oracle.com/" />
  </soap:Body>
</soap:Envelope>
```

The SOAPAction for this message is as follows:

SOAPAction
HelloService

Configuration

To configure the **SOAPAction** filter, complete the following:

1. Enter an appropriate name for the filter in the **Name** field.
2. Enter a regular expression to match the value of the SOAPAction HTTP header in the **SOAPAction** field. For example, enter `^getQuote$` to exactly match a SOAPAction header with a value of `getQuote`. Incoming messages with a matching SOAPAction value are passed on to the next filter on the success path in the policy.

SOAP Operation Resolver

Overview

The **SOAP Operation** filter enables you to identify an incoming XML message based on the SOAP Operation in the message.

The following example shows how to find the SOAP Operation of an incoming message. Consider the following SOAP message:

```
POST /services/timeservice HTTP/1.0
Host: localhost:8095
Content-Length: 374
SOAPAction: TimeService
Accept-Language: en-US
UserAgent: Enterprise Gateway
Content-Type: text/XML; utf-8

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns1:getTime xmlns:ns1="Some-URI">
      <ns1:city>Dublin</ns1:city>
    </ns1:getTime>
  </soap:Body>
</soap:Envelope>
```

The SOAP Operation for this message and its namespace are as follows:

SOAP Operation	getTime
SOAP Operation Namespace	urn:timeservice

The SOAP Operation is the first child element of the SOAP <Body> element.

Configuration

To configure the **SOAP Operation** filter, complete the following:

1. Enter an appropriate name for the filter in the **Name** field.
2. Enter the name of the SOAP Operation in the **Operation** field. Incoming messages with an operation name matching the value entered here are passed on to the next Success filter in the policy.
3. Enter the namespace to which the SOAP Operation belongs in the **Namespace** field.

Getting Started with Routing Configuration

Overview

This topic describes how to configure the Enterprise Gateway to send messages to external Web Services. The Enterprise Gateway offers a number of different filters that can be used to route messages. Depending on how the Enterprise Gateway is perceived by the client, different combinations of routing filters can be used.

For example, the Enterprise Gateway can act both as a proxy and as an endpoint (in-line) server for a client, depending on how the client is configured. In each case, the request received by the Enterprise Gateway appears slightly different, and the Enterprise Gateway can take advantage of this when routing the message onwards. Furthermore, the Enterprise Gateway can provide *service virtualization* by shielding the underlying hierarchy of back-end Web Services from clients.

This topic explains how clients can use the Enterprise Gateway both as a proxy and as an endpoint server. It then shows how *service virtualization* works. When these basic concepts are explained, this topic helps you to identify the combination of routing filters that is best suited to your deployment scenario.

Proxy or Endpoint Server

The Enterprise Gateway can be used by clients as both a proxy sever and an endpoint server. When a client uses the Enterprise Gateway as a proxy server, it sends up the complete URL of the destination Web Service in the HTTP request line. The Enterprise Gateway can use the URL to determine the host and port to route the message to. The following HTTP request shows an example of a request received by the Enterprise Gateway when acting as a proxy for a client:

```
POST http://localhost:8080/services/getHello HTTP/1.1
```

Alternatively, when the Enterprise Gateway is acting as an endpoint (in-line) server, the client sends the request directly to the Enterprise Gateway. In this case, the request line appears as follows:

```
POST /services/getHello HTTP/1.1
```

In this case, only the path on the server is specified, and no scheme, host, or port number is included in the HTTP request line. Because this information is not provided by the client, the Enterprise Gateway must be explicitly configured to route the message on to the specific destination.

Service Virtualization

It is sometimes desirable to shield the underlying structure of the directory hierarchy in which Web Services reside from external clients. You can do this by providing a mapping between the path that the client accesses and the actual path at which the Web Service resides.

For example, suppose you have two Web Services accessible at the `/helloService/getHello` and `/financeService/getQuote` URIs. You may wish to hide that these services are deployed under different paths, perhaps exposing them under a common `/services` base URI (for example, `/services/getHello` and `/services/getQuote`). The client is therefore unaware of the underlying hierarchy (for example, directory structure) of the two Web Services. This is termed *service virtualization*.

Choosing the Correct Routing Filters

This section first identifies how your clients perceive the Enterprise Gateway, and then determines whether you wish to virtualize your back-end Web Services. Depending on these requirements, you can use different combinations of routing filters, as described in the use cases in the subsequent sections.

Consider the following to determine which combination of routing filters is most appropriate for your scenario:

Proxy or Endpoint?

- If the client is using the Enterprise Gateway as a proxy server, see cases 1 or 2 below, depending on whether *service virtualization* is required.
- Alternatively, if the client using the Enterprise Gateway as the endpoint of the connection (as an in-line server), see cases 3 or 4 below.

Service Virtualization?

- If you want to shield the hierarchy of protected Web Services by exposing a *virtual* view of these services, see cases 2, 4, and 5 below.
- If *service virtualization* is not important, see cases 1 and 3 below.

These permutations are summarized in the following table:

Proxy or Endpoint?	Service Virtualization?	Example
Proxy	No	Case 1: Proxy without Virtualization
Proxy	Yes	Case 2: Proxy with Virtualization
Endpoint	No	Case 3: Endpoint without Virtualization
Endpoint	Yes	Case 4: Endpoint with Virtualization
Proxy or Endpoint	Yes	Case 5: Simple Redirect

Case 1: Proxy without Service Virtualization

In this case, the Enterprise Gateway is configured as an HTTP proxy for the client, and maintains the original path used by the client in the HTTP request. For example, if the Enterprise Gateway is listening at `http://localhost:8080/`, and the Web Service is running at `http://localhost:5050/services/getQuote`, the request line of the client HTTP request appears as follows:

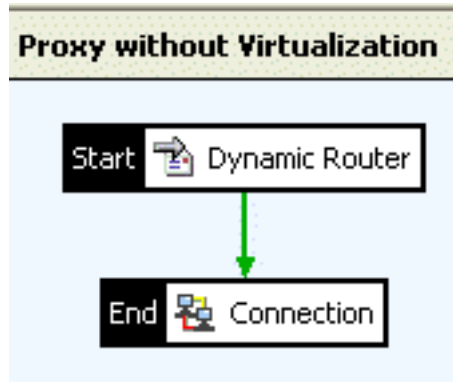
```
POST http://localhost:5050/services/getQuote HTTP/1.1
```

Because the client is configured to use the Enterprise Gateway instance running on `localhost` at port `8080` as its HTTP proxy, the client automatically sends all messages to the proxy. However, it includes the full URL of the ultimate destination of the message in the request line of the HTTP request.

When the Enterprise Gateway receives the request, it extracts this URL from the request line and uses it to determine

the destination of the message. In the above example, the Enterprise Gateway routes the message on to `http://localhost:5050/services/getQuote`.

You can configure the following policy to route the message to the URL specified in the request line of the client request:



The following table explains the role of each filter in the policy. For more information on a specific filter, click the appropriate link in the **Details** column.

<i>Filter</i>	<i>Role in Policy</i>	<i>Details</i>
Dynamic Router	Extracts the URL of the destination Web Service from the request line of the incoming HTTP request. The Dynamic Router is normally used when the Enterprise Gateway is perceived as a proxy by the client.	Dynamic Router
Connection	Establishes the connection to the destination Web Service, and sends the message over this connection. This connection can be mutually authenticated if necessary.	Connection

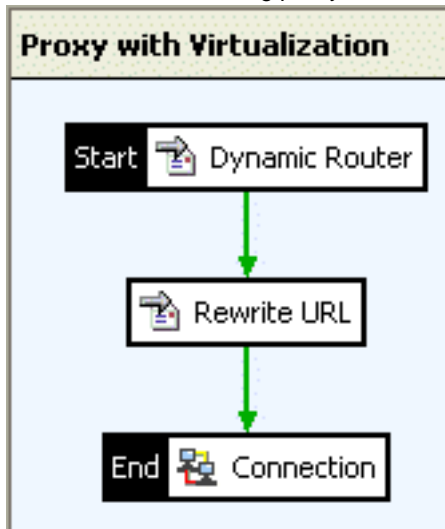
Case 2: Proxy with Service Virtualization

In this case, the Enterprise Gateway is also perceived as an HTTP proxy by the client. However, the Enterprise Gateway exposes a *virtualized* view of the services that it protects. This is termed *service virtualization*.

To achieve this, the Enterprise Gateway must provide a mapping between the path used by the client and the path under which the service is deployed. Assuming the Enterprise Gateway is running at `http://localhost:8080/services`, and the Web Service is deployed at `http://localhost:5050/financialServices/quotes/getQuote`, the following example shows what the client may send up in the HTTP request line:

```
POST http://localhost:5050/services/getQuote HTTP/1.1
```

To achieve this, the Enterprise Gateway must provide a mapping between what the client requests (`/services/getQuote`), and the address of the Web Service (`/financialServices/quotes/getQuote`). The **Rewrite URL** filter in the following policy fulfills this role:



The following table explains the roles of the each filter in the policy:

<i>Filter</i>	<i>Role in Policy</i>	<i>Details</i>
Dynamic Router	Extracts the URL of the destination Web Service from the request line of the incoming HTTP request. The Dynamic Router is normally used when the Enterprise Gateway is perceived as a proxy by the client.	Dynamic Router
Rewrite URL	Specifies the mapping between the path requested by the client and the path under which the Web Service is deployed, therefore providing service virtualization.	Rewrite URL
Connection	Establishes the connection to the destination Web Service, and sends the message over this connection. This connection can be mutually authenticated if necessary.	Connection

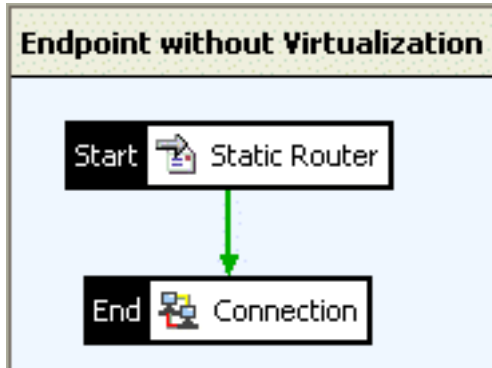
Case 3: Endpoint without Service Virtualization

In this scenario, the client sees the Enterprise Gateway as the endpoint to its connection, and the Enterprise Gateway must be configured to route messages on to a specific destination. For example, assuming that the Enterprise Gateway is running at `http://localhost:8080/services`, the request line of the client's HTTP request is received by the Enterprise Gateway as follows:

```
POST /services HTTP/1.1
```

The request line above shows that no information about the scheme, host, or port of the destination Web Service is specified. Therefore, this information must be configured in the Enterprise Gateway so that it knows where to route the message on to. The **Static Router** enables the user to enter connection details for the destination Web Service.

Assuming that the Web Service is running at `http://localhost:5050/stockquote/getPrice`, the host, port, and scheme respectively are: `localhost`, `5050`, and `http`. You must explicitly configure this information in the **Static Router**. The following policy illustrates this scenario:



The following table explains the role of each filter in the above policy:

Filter	Role in Policy	Details
Static Router	Enables the user to explicitly specify the host, port, and scheme at which the Web Service is listening. This filter must be used when the client sees the Enterprise Gateway as the endpoint to its connection (the Enterprise Gateway is not acting as a proxy for the client).	Static Router
Connection	Establishes the connection to the destination Web Service, and sends the message over this connection. This connection can be mutually authenticated if necessary.	Connection

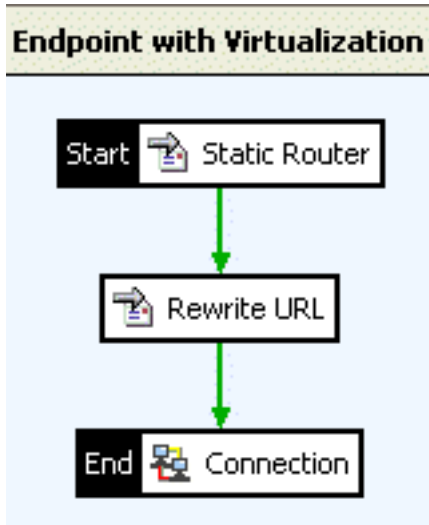
Case 4: Endpoint with Service Virtualization

In this case, the Enterprise Gateway acts as the endpoint to the client connection (and not as a proxy), and hides the deployment hierarchy of protected Web Services from clients. In other words, it performs *service virtualization*.

In this scenario, the client sends messages directly to the Enterprise Gateway. For example, assuming that the Enterprise Gateway is running at `http://localhost:8080/services`, and the Web Service is running at `http://localhost:5050/stockquote/getPrice`, the request line of the client HTTP request is received by the Enterprise Gateway as follows:

```
POST /services HTTP/1.1
```

You can then configure the **Static Router** filter to route the message on to port 8080 on localhost using the `http` scheme, while the **Rewrite URL** filter provides the mapping between the path requested by the client (`/services`) and the path under which the Web Service is deployed (`/stockquote/getPrice`). The following policy illustrates a sample policy that provides *service virtualization* when the Enterprise Gateway is used as an endpoint:



The following table explains the role of each filter in the policy:

Filter	Role in Policy	Details
Static Router	Enables you to explicitly specify the host, port, and scheme at which the Web Service is listening. This filter can be used when the client sees the Enterprise Gateway as the endpoint to its connection (not as a proxy for the client).	Static Router
Rewrite URL	Provides the mapping between the path requested by the client and the path under which the Web Service is deployed.	Rewrite URL
Connection	Establishes the connection to the destination Web Service, and sends the message over this connection. This connection can be mutually authenticated if necessary.	Connection

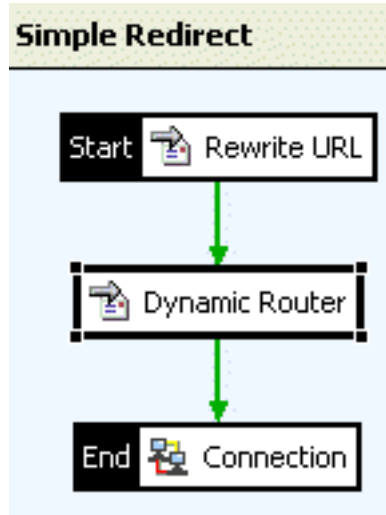
Important Note:

Alternatively, instead of using the **Static Router**, **Rewrite URL**, and **Connection** filters, you can use the **Connect to URL** filter, which is equivalent to using these three filters combined. You can configure the **Connect to URL** filter to send messages to a Web Service simply by specifying the destination URL. For more details, see the [Connect to URL](#) topic.

Case 5: Simple Redirect

In some cases, the Enterprise Gateway must route the incoming message to an entirely different URL. You can use the **Rewrite URL** filter for this purpose, in addition to rewriting the path on which the request is received (as described in cases 2 and 4 above). Note that the full URL of the destination Web Service should be specified in this case in the **Rewrite URL** filter.

The following policy illustrates the use of the **Redirect URL** filter to route messages to a fully qualified URL:



The following table describes the role of each filter in the policy:

<i>Filter</i>	<i>Role in Policy</i>	<i>Details</i>
Rewrite URL	Used to specify the fully qualified URL of the destination Web Service.	Rewrite URL
Dynamic Router	In this case, the Dynamic Router filter is used to parse the URL specified in the Rewrite URL filter into its constituent parts. The HTTP scheme, port, and host of the Web Service are extracted and set to the internal message object for use by the Connection filter.	Dynamic Router
Connection	Establishes the connection to the destination Web Service, and sends the message over this connection. This connection can be mutually authenticated if necessary.	Connection

Case 6: Routing on to an HTTP Proxy

This is a more advanced case where the Enterprise Gateway is configured to route on through an HTTP proxy to the

back-end Web Service, which sits behind the proxy. When the Enterprise Gateway is configured to route through a proxy, it connects directly to the proxy, and sends a request including the full URL of the target Web Service in the HTTP request URI. When the HTTP proxy receives this request, it uses the URL in the request line to determine where to route the message on to. The following example shows the request line of a request made through a proxy:

```
POST http://localhost:8080/services/getQuote HTTP/1.1
```

The following filters are required to configure the Enterprise Gateway to route through an HTTP proxy:

Filter	Role in Policy	Details
Static Router	You must explicitly specify the host, port, and scheme of the HTTP proxy in this filter.	Static Router
Rewrite URL	Enter the full URL of the Web Service in this filter (for example, <code>http://HOST:8080/myServices</code>). Because you are routing through a proxy, the full URL specified here is sent in the request line of the HTTP request.	Rewrite URL
Connection	In this case, the Connection filter connects to the HTTP proxy, which in turn routes the message on to the destination server named in the request URI. The Send via Proxy option must be enabled in the Connection filter to facilitate this.	Connection

Important Note:

It is important to note the differences between how the filters are configured to route on through a proxy and the scenario described in [Case 4: Endpoint with Service Virtualization](#) where no proxy is involved:

- **Static Router:**
When the Enterprise Gateway routes on to an HTTP proxy, the **Static Router** filter is configured with the details of the HTTP proxy. Otherwise, the **Static Router** filter is given the details of the Web Service endpoint directly.
- **Rewrite URL:**
The full URL of the Web Service endpoint must be specified in this filter when the Enterprise Gateway routes through a proxy. The full URL is then included in the request line of the HTTP request to the proxy. In cases where no proxy is involved, the **Rewrite URL** filter is only necessary when the back-end Web Services are virtualized. In this case, the Enterprise Gateway must send the request to a different URI than that requested by the client.
- **Connection:**
When routing through a proxy, the **Send via Proxy** option must be enabled in the **Connection** filter. This is not necessary when no proxy sits between the Enterprise Gateway and the back-end Web Service.

Summary

The following are the key concepts to consider when configuring the Enterprise Gateway to connect to external Web Services:

- The **Connection** or **Connect to URL** filter *must* always be used because it establishes the connection to the Web Service.
- *Service virtualization* can be achieved using the **Rewrite URL** or **Connect to URL** filter.
- If the client is configured to use the Enterprise Gateway as a proxy, the Enterprise Gateway can use the **Dynamic Router** filter to extract the URL from the request line of the HTTP request. It can then route the message on to this URL.
- If the client sees the Enterprise Gateway as the endpoint of the connection (not as a proxy), the **Static Router** filter can be used to explicitly configure the host, port, and scheme of the destination Web Service. Alternatively, you can use the **Connect to URL** to specify a URL.

Routing Wizard

Finally, the Enterprise Gateway provides a **Routing Wizard** to enable you to quickly configure and auto-generate the filters necessary to route messages on to a specific destination. For more details, see the [Routing Wizard](#) topic.

Routing Wizard

Overview

For convenience, the Enterprise Gateway provides a **Routing Wizard** to enable administrators to quickly configure the filters necessary to route messages on to a specific destination. The wizard auto-generates the following filters:

<i>Filter</i>	<i>Role</i>	<i>Details</i>
Rewrite URL	This filter determines the request URI of the HTTP request that is ultimately made by the Connection filter below. You should enter a complete URL (for example, <code>http://host:8080/services</code>), from which the host and port is extracted and used to configure the Static Router below.	Rewrite URL
Static Router	The Static Router filter specifies the host of the destination server together with the port to connect to on that host.	Static Router
Connection	The Connection filter establishes the connection to the URL, host, and port specified in the Rewrite URL and Static Router filters. If an SSL connection is required, you can select a certificate from the Trusted Certificate Store to use to authenticate to the destination server. You can also select what certificates are considered trusted by the Enterprise Gateway so that the destination server's certificate can be trusted.	Connection

To use the **Routing Wizard** for a particular policy, right-click the policy in the **Policies** tab in the Policy Studio, and select **Routing Wizard**. Configuring the **URL** field and/or the **Proxy Settings** tab in the **Routing Wizard** auto-generates a **Rewrite URL** filter and a **Static Router** filter.

Configuration

You can configure the following fields in the **Routing Wizard**:

URL:

Enter the *full* URL of the destination Web Service. The host, port, and scheme (HTTP or HTTPS) are extracted from the URL and used to configure a **Static Router** filter.

Proxy Host:

If you want to send the request using an HTTP proxy, configure the **Proxy Host** and **Proxy Port**. In this case, the **Static Router** filter is configured with the host and port entered in these fields. The **Connection** filter sends the complete URL specified in the **URL** field as the request URI to the proxy, as required by the HTTP specification. The proxy then knows where to route the message on to.

Note:

If the **Proxy Host** and **Proxy Port** fields are completed, the wizard automatically selects the **Send via proxy** field on the **Advanced** tab of the auto-generated **Connection** filter.

Proxy Port:

If you want to route messages using a proxy to the destination Web Service, enter the port on the **Proxy Host** specified above on which the proxy accepts requests.

SSL Port:

If the proxy is SSL-enabled, enter the SSL port in this field.

Trusted Certificates, Client SSL Authentication, and HTTP Authentication Tabs

The settings configured on the remaining tabs of the wizard correspond to the settings configured on the tabs displayed on the **Connection** filter. For more information on configuring the fields on these tabs, see the [Connection](#) topic.

Call Internal Service

Overview

The **Call Internal Service** filter is a special filter that passes messages to an internal Servlet Application that has been deployed at the Enterprise Gateway. The appropriate application is selected based on the relative path on which the request message was received.

The filter is used by Management Services that are configured to listen on the Management Interface on port 8090. For more information on how the **Call Internal Service** filter is used by these services, see the section on [Management Services](#) in the [Configuring HTTP Services](#) topic.

Connection

Overview

The **Connection** filter makes the connection to the remote Web Service. It relies on connection details that are set by the other filters in the **Routing** category. Because the **Connection** filter connects out to other services, it negotiates the SSL handshake involved in setting up a mutually authenticated secure channel.

Depending on how the Enterprise Gateway is perceived by the client, different combinations of routing filters can be used. For an introduction to using the various filters in the **Routing** category, see the topic on [Getting Started with Routing Configuration](#).

General Configuration

Enter an appropriate name for the filter in the **Name** field. Click the tabs to configure the various aspects of the **Connection** filter.

Trusted Certificates

This section lists all CA certificates that have been imported into the Oracle **Certificate Store**. Select the certificates that the Enterprise Gateway considers to be trusted when it attempts to establish a connection to a remote server. The remote server's SSL certificate must be issued by one of the selected trusted certificates on this tab.

Client SSL Authentication

You can configure the Enterprise Gateway to authenticate itself to the remote Web Service. It does so using the certificate selected on this tab.

HTTP Authentication

The Enterprise Gateway can use both HTTP basic and HTTP digest authentication to authenticate to the remote server. In both cases, the **User Name** and **Password** of a user must be specified in the fields provided.

Kerberos Authentication

The settings on this tab enable you to authenticate to a Kerberos Service by sending a Kerberos service ticket in the HTTP request to that service.

Note:

You can also configure the Enterprise Gateway to authenticate to a Kerberos Service by including the relevant Kerberos tokens inside the XML message. For more details, see the [Kerberos Client Authentication](#) topic.

Kerberos Client:

The selected Kerberos Client has two roles. First, it must obtain a Kerberos TGT (Ticket Granting Ticket). Second, it uses this TGT to obtain a service ticket for the **Kerberos Service Principal** selected below.

You can configure Kerberos Clients globally on the **External Connections** tab in the Policy Studio. These can then be selected in the **Kerberos Client** drop-down list. For more details on configuring Kerberos Clients, see the [Kerberos Client](#) topic.

Kerberos Service Principal:

The Kerberos Client selected in the drop-down list must obtain a ticket from the Kerberos Ticket Granting Server (TGS) for the selected Kerberos Service Principal. The Service Principal can be used to uniquely identify the Service in the Kerberos realm. The TGS grants a ticket for the selected Principal, which the client can then send to the Kerberos Service. The Principal in the ticket must match the Kerberos Service's Principal for the client to be successfully authenticated.

You can also configure Kerberos Principals globally on the **External Connections** tab in the Policy Studio. For more details on configuring Kerberos Principals, see the [Kerberos Principals](#) topic.

Send token with first request:

In some cases, the client may not authenticate (send the `Authorization` HTTP header) to the Kerberos Service on its first request. The Kerberos Service should then respond with an HTTP 401 response containing a `www-Authenticate: Negotiate` HTTP header. This header value instructs the client to authenticate to the server by sending up the `Authorization` header. The client then sends up a second request, this time with the `Authorization` header, which contains the relevant Kerberos token. Select this option to force the **Connection** filter to *always* send the `Authorization` HTTP header that contains the Kerberos Service ticket on its first request to the Kerberos Service.

Send body only after establish context:

You can configure the Kerberos client to only send the message body after the context has been fully established (the client has mutually authenticated with the service).

Pass when service returns 200 even if context not established:

In some rare cases, a Kerberos Service may return a 200 OK response to a Kerberos Client's initial request even though the security context has not yet been fully established. This 200 OK response may not contain the `www-authenticate` HTTP header.

By selecting this option, you are instructing the **Connection** filter to send the request to the Kerberos Service despite that the context has not been established. It is the responsibility of the Kerberos Service to decide whether to process the request depending on the status of the security context.

Behavior

The **Behavior** tab enables you to configure **Retries** and **Failure Settings**. By default, both of these sections are collapsed. Click a section to expand it.

Retries:

To specify the retry settings for this filter, complete the following fields:

Perform Retries	Select whether the filter performs retries. By default, this setting is not selected, no retries are performed, and all Retries settings are disabled. This means that the filter only attempts to perform the connection once.
Retry On	Select the HTTP status ranges on which retries can be performed. If a host responds with an HTTP status code that matches one of the selected ranges, this filter performs a retry. Select one or more ranges in the table (for example, <code>Client Error 400-499</code>). For details on adding custom HTTP status ranges, see the next subsection.
Retry Count	Enter the maximum number of retries to attempt. The default is 5.
Retry Interval (ms)	Enter the amount of time to delay between retries in milliseconds. The default is 500 ms.

Adding HTTP Status Ranges

To add an HTTP status range to the default list displayed in the **Retry On** table, click the **Add** button. In the **Configure HTTP Status Code** dialog, complete the following fields:

Name	Enter a name for the HTTP status range.
Start status	Enter the first HTTP status code in the range of status

	codes that you wish to monitor.
End status	Enter the last HTTP status code in the range of status codes that you wish to monitor.

If you wish to monitor one specific status code only, enter the same code in the **Start status** and **End status** fields. Click **OK** to finish. You can manage existing HTTP status ranges using the **Edit** and **Delete** buttons.

Failure Settings:

To specify the failure settings for this filter, complete the following fields:

Consider SLA Breach as Failure	Select whether to attempt the connection if a configured SLA has been breached. This is not selected by default. If this option is selected, and an SLA breach is encountered, the filter returns <i>false</i> .
Save Transaction on Failure (for replay)	Select whether to store the incoming message in the specified directory and file if a failure occurs during processing. This is not selected by default.
File name	Enter the name of the file that the message content is saved to. You can specify this using a property, which is expanded to the specified value at runtime. Defaults to <code>\${id}.out</code> .
Directory	Enter the directory that the file is saved to. You can specify this using a property, which is expanded to the specified value at runtime. Defaults to <code>\${VINSTDIR}/message-archive</code> , where <code>VINSTDIR</code> is the root of your Enterprise Gateway installation.
Maximum number of files in directory	Enter the maximum number of files that can be saved in the directory. Defaults to 500.
Maximum file size	Enter the maximum file size in MB. Defaults to 1000.
Include HTTP Headers	Select whether to include HTTP headers in the file. HTTP headers are not included by default.
Call policy on Connection Failure	Select whether to execute a policy circuit in the event of a connection failure. This is not selected by default.
Connection Failure Policy	Click the browse button on the right, and select the policy to run in the event of a connection failure in the dialog.

Advanced

This tab enables you to configure certain advanced features of the **Connection** filter. The following configuration options are available:

Handles Redirects:

Specifies whether the Enterprise Gateway handles HTTP redirects, and connects to the redirect URL specified in the HTTP response. This setting is enabled by default.

Send via Proxy:

Select this option if the Enterprise Gateway must connect to the destination Web Service through an HTTP proxy. In this case, the Enterprise Gateway includes the full URL of the destination Web Service in the request line of the HTTP re-

quest. For example, if the destination Web Service resides at `http://localhost:8080/services`, the request line is as follows:

```
POST http://localhost:8080/services HTTP/1.1
```

If the Enterprise Gateway is not routing through a proxy, the request line is as follows:

```
POST /services HTTP/1.1
```

Proxy Server:

You can configure a specific proxy server to use for the connection in the **Proxy Server** field. Click the button next to this field, and select an existing proxy server in the dialog. You must have already configured the proxy server on the **External Connections** tab. For more details, see the [Proxy Servers](#) topic.

Transparent Proxy (present client's IP address to server)

Enables the use of the Enterprise Gateway as a *transparent proxy* on Linux systems with the `TPROXY` kernel option set. When selected, the IP address of the original client connection that caused the circuit to be invoked is used as the local address of the connection to the destination server. For more details, see [Configuring a Transparent Proxy](#).

Ciphers:

The **Ciphers** field enables the administrator to specify the ciphers that the server supports. The server sends this list of supported ciphers to the destination server, which then selects the highest strength common cipher as part of the SSL handshake. The selected cipher is then used to encrypt the data as it is sent over the secure channel.

HTTP Host Header:

An HTTP 1.1 client *must* send a `Host` header in all HTTP 1.1 requests. The `Host` header identifies the host name and port number of the requested resource as specified in the original URL given by the client.

When routing messages on to target Web Services, the Enterprise Gateway can forward on the `Host` as received from the client, or it can specify the address and port number of the destination Web Service in the `Host` header that it routes onwards.

Select **Use Host header specified by client** to force the Enterprise Gateway to always forward on the original `Host` header that it received from the client. Alternatively, to configure the Enterprise Gateway to include the address and port number of the destination Web Service in the `Host` header, select the **Generate new Host header** radio button.

Connect to URL

Overview

The **Connect to URL** filter is the simplest routing filter to use to connect to a target Web Service. To configure this filter to send messages to a Web Service, you need only enter the URL of the service in the **URL** field. If the Web Service is SSL enabled or requires mutual authentication, you can use the other tabs on the **Connect to URL** filter to configure this.

Depending on how the Enterprise Gateway is perceived by the client, different combinations of routing filters can be used. Using the **Connect to URL** filter is equivalent to using the following combination of routing filters:

- Static Router
- Rewrite URL
- Connection

The **Connect to URL** filter enables the Enterprise Gateway to act as the endpoint to the client connection (and not as a proxy), and to hide the deployment hierarchy of protected Web Services from clients. In other words, the Enterprise Gateway performs *service virtualization*. For an introduction to routing scenarios and the filters in the **Routing** category, see the [Getting Started with Routing Configuration](#) topic.

General Configuration

Configure the following general settings:

Name:

Enter an appropriate name for the filter.

URL:

Enter the complete URL of the target Web Service.

Trusted Certificates

When Enterprise Gateway connects to a server over SSL, it must decide whether to trust the server's SSL certificate. You can select a list of CA or server certificates from the **Trusted Certificates** tab that are considered trusted by Enterprise Gateway when connecting to the server specified in the **URL** field on this dialog.

The table displayed on the **Trusted Certificates** tab lists all certificates imported into the Enterprise Gateway Certificate Store. To *trust* a certificate for this particular connection, select the box next to the certificate in the table.

Client SSL Authentication

In cases where the destination server requires clients to authenticate to it using an SSL certificate, you must select a client certificate on the **Client SSL Authentication** tab. Select the checkbox next to the client certificate that you want to use to authenticate to the server specified in the **URL** field.

HTTP Authentication

If the destination server requires clients to authenticate to it using HTTP basic or digest authentication, use the fields on this tab.

None, HTTP Basic, or HTTP Digest:

Select the method that you want to use to authenticate to the server.

User Name:

Enter the user name you want to use to authenticate to the server.

Password:

Enter the password for this user.

Kerberos Authentication

The settings on this tab enable you to authenticate to a Kerberos Service by sending a Kerberos service ticket in the HTTP request to that service.

Note:

You can also configure the Enterprise Gateway to authenticate to a Kerberos Service by including the relevant Kerberos tokens inside the XML message. For more details, see the [Kerberos Client Authentication](#) topic.

Kerberos Client:

The selected Kerberos Client has two roles. First, it must obtain a Kerberos TGT (Ticket Granting Ticket). Second, it uses this TGT to obtain a service ticket for the **Kerberos Service Principal** selected below.

You can configure Kerberos Clients globally on the **External Connections** tab in the Policy Studio. These can then be selected in the **Kerberos Client** drop-down list. For more details on configuring Kerberos Clients, see the [Kerberos Client](#) topic.

Kerberos Service Principal:

The Kerberos Client selected in the drop-down list must obtain a ticket from the Kerberos Ticket Granting Server (TGS) for the selected Kerberos Service Principal. The Service Principal can be used to uniquely identify the Service in the Kerberos realm. The TGS grants a ticket for the selected Principal, which the client can then send to the Kerberos Service. The Principal in the ticket must match the Kerberos Service's Principal for the client to be successfully authenticated.

You can also configure Kerberos Principals globally on the **External Connections** tab in the Policy Studio. For more details on configuring Kerberos Principals, see the [Kerberos Principals](#) topic.

Send token with first request:

In some cases, the client may not authenticate (send the `Authorization` HTTP header) to the Kerberos Service on its first request. The Kerberos Service should then respond with an HTTP 401 response containing a `WWW-Authenticate: Negotiate` HTTP header. This header value instructs the client to authenticate to the server by sending up the `Authorization` header. The client then sends up a second request, this time with the `Authorization` header, which contains the relevant Kerberos token. Select this option to force the **Connect to URL** filter to *always* send the `Authorization` HTTP header that contains the Kerberos Service ticket on its first request to the Kerberos Service.

Send body only after establish context:

You can configure the Kerberos client to only send the message body after the context has been fully established (the client has mutually authenticated with the service).

Pass when service returns 200 even if context not established:

In some rare cases, a Kerberos Service may return a 200 OK response to a Kerberos Client's initial request even though the security context has not yet been fully established. This 200 OK response may not contain the `WWW-authenticate` HTTP header.

By selecting this option, you are instructing the **Connect to URL** filter to send the request to the Kerberos Service even if the context has not been established. It is the responsibility of the Kerberos Service to decide whether to process the request depending on the status of the security context.

Behavior

The **Behavior** tab enables you to configure **Retries** and **Failure Settings**. By default, both of these sections are collapsed. Click a section to expand it.

Retries:

To specify the retry settings for this filter, complete the following fields:

Perform Retries	Select whether the filter performs retries. By default, this setting is not selected, no retries are performed, and all Retries settings are disabled. This means that the filter only attempts to perform the connection once.
Retry On	Select the HTTP status ranges on which retries can be performed. If a host responds with an HTTP status code that matches one of the selected ranges, this filter performs a retry. Select one or more ranges in the table (for example, <code>Client Error 400-499</code>). For details on adding custom HTTP status ranges, see the next subsection.
Retry Count	Enter the maximum number of retries to attempt. The default is 5.
Retry Interval (ms)	Enter the amount of time to delay between retries in milliseconds. The default is 500 ms.

Adding HTTP Status Ranges

To add an HTTP status range to the default list displayed in the **Retry On** table, click the **Add** button. In the **Configure HTTP Status Code** dialog, complete the following fields:

Name	Enter a name for the HTTP status range.
Start status	Enter the first HTTP status code in the range of status codes that you wish to monitor.
End status	Enter the last HTTP status code in the range of status codes that you wish to monitor.

If you wish to monitor one specific status code only, enter the same code in the **Start status** and **End status** fields. Click **OK** to finish. You can manage existing HTTP status ranges using the **Edit** and **Delete** buttons.

Failure Settings:

To specify the failure settings for this filter, complete the following fields:

Consider SLA Breach as Failure	Select whether to attempt the connection if a configured SLA has been breached. This is not selected by default. If this option is selected, and an SLA breach is encountered, the filter returns <code>false</code> .
Save Transaction on Failure (for replay)	Select whether to store the incoming message in the specified directory and file if a failure occurs during processing. This is not selected by default.
File name	Enter the name of the file that the message content is saved to. You can specify this using a property, which is expanded to the specified value at runtime. Defaults to <code>\${id}.out</code> .
Directory	Enter the directory that the file is saved to. You can specify this using a property, which is expanded to the specified value at runtime. Defaults to <code>\${VINSTDIR}/message-archive</code> , where <code>VINSTDIR</code> is the root of your Enterprise Gateway installation.
Maximum number of files in directory	Enter the maximum number of files that can be saved in

	the directory. Defaults to 500.
Maximum file size	Enter the maximum file size in MB. Defaults to 1000.
Include HTTP Headers	Select whether to include HTTP headers in the file. HTTP headers are not included by default.
Call policy on Connection Failure	Select whether to execute a policy circuit in the event of a connection failure. This is not selected by default.
Connection Failure Policy	Click the browse button on the right, and select the policy to run in the event of a connection failure in the dialog.

Advanced

This tab enables you to configure certain advanced features of the **Connect to URL** filter. The following configuration options are available:

Handles Redirects:

Specifies whether the Enterprise Gateway handles HTTP redirects, and connects to the redirect URL specified in the HTTP response. This setting is enabled by default.

Send via Proxy:

Select this option if the Enterprise Gateway must connect to the destination Web Service through an HTTP proxy. In this case, the Enterprise Gateway includes the full URL of the destination Web Service in the request line of the HTTP request. For example, if the destination Web Service resides at `http://localhost:8080/services`, the request line is as follows:

```
POST http://localhost:8080/services HTTP/1.1
```

If the Enterprise Gateway was not routing through a proxy, the request line is as follows:

```
POST /services HTTP/1.1
```

Proxy Server:

You can configure a specific proxy server to use for the connection in the **Proxy Server** field. Click the button next to this field, and select an existing proxy server in the dialog. You must have already configured the proxy server on the **External Connections** tab. For more details, see the [Proxy Servers](#) topic.

Transparent Proxy (present client's IP address to server)

Enables the use of the Enterprise Gateway as a *transparent proxy* on Linux systems with the `TProxy` kernel option set. When selected, the IP address of the original client connection that caused the circuit to be invoked is used as the local address of the connection to the destination server. For more details, see [Configuring a Transparent Proxy](#).

Ciphers:

The **Ciphers** field enables the administrator to specify the ciphers that Enterprise Gateway supports. The Enterprise

Gateway sends this list of supported ciphers to the destination server, which then selects the highest strength common cipher as part of the SSL handshake. The selected cipher is then used to encrypt the data as it is sent over the secure channel.

HTTP Host Header:

An HTTP 1.1 client **must** send a `Host` header in all HTTP 1.1 requests. The `Host` header identifies the hostname and port number of the requested resource as specified in the original URL given by the client.

When routing messages on to target Web Services, the Enterprise Gateway can forward on the `Host` as received from the client, or it can specify the address and port number of the destination Web Service in the `Host` header that it routes onwards.

Select **Use Host header specified by client** to force the Enterprise Gateway to always forward on the original `Host` header that it received from the client. Alternatively, to configure the Enterprise Gateway to include the address and port number of the destination Web Service in the `Host` header, select the **Generate new Host header** radio button.

Dynamic Router

Overview

The Enterprise Gateway can act as a proxy for clients of the secured Web Service. When a client uses a proxy, it includes the fully qualified URL of the destination in the request line of the HTTP request. It sends this request to the configured proxy, which then forwards the request to the host specified in the URL. The relative path used in the original request is preserved by the proxy on the outbound connection.

The following is an example of an HTTP request line that was made through a proxy, where `WEB_SERVICE_HOST` is the name or IP address of the machine hosting the destination Web Service:

```
POST http://WEB_SERVICE_HOST:80/myService HTTP/1.0
```

When the Enterprise Gateway acts as a proxy for clients, it can receive requests like the one above. The **Dynamic Router** filter can route the request on to the URL specified in the request line (`http://WEB_SERVICE_HOST:80/myService`).

Depending on how the Enterprise Gateway is perceived by the client, different combinations of routing filters can be used. For an introduction to using the various filters in the **Routing** category, see the topic on [Getting Started with Routing Configuration](#).

Configuration

Enter an appropriate name for the router in the **Name** field on the **Dynamic Router** filter configuration screen.

HTTP Status Code

Overview

This filter sets the HTTP status code on response messages. This enables an administrator to ensure that a more meaningful response is sent to the client in the case of an error or anomaly occurring in a configured policy.

For example, if a **Relative Path** filter fails, it may be useful to return a 503 `Service Unavailable` response. Similarly, if a user does not present identity credentials when attempting to access a protected resource, you can configure the Enterprise Gateway to return a 401 `Unauthorized` response to the client.

HTTP status codes are returned in the *status-line* of an HTTP response. The following are some typical examples:

```
HTTP/1.1 200 OK
HTTP/1.1 400 Bad Request
HTTP/1.1 500 Internal Server Error
```

Configuration

Name:

Enter an appropriate name for this filter.

HTTP response code status:

Enter the status code returned to the client. For a complete list of status codes, see the [HTTP Specification](http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html) [http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html].

Insert WS-Addressing

Overview

The WS-Addressing specification defines a transport-independent standard for including addressing information in SOAP messages. The Enterprise Gateway can generate WS-Addressing information based on a configured endpoint in a policy, and then insert this information into SOAP messages.

Configuration

Complete the following fields to configure the Enterprise Gateway to insert WS-Addressing information into the SOAP message header.

Name:

Enter an appropriate name for the filter.

To:

The message is delivered to the specified destination.

From:

Informs the destination server where the message originated from.

Reply To:

Indicates to the destination server where it should send response messages to.

Fault To:

Indicates to the destination server where it should send fault messages to.

MessageID:

A unique identifier to distinguish this message from others at the destination server. It also provides a mechanism for correlating a specific request with its corresponding response message.

Action:

The specified action indicates what action the destination server should take on the message. Typically, the value of the WS-Addressing `Action` element corresponds to the `SOAPAction` on the request message. For this reason, this field defaults to the `soap.request.action` message attribute.

Relates To:

If responses are to be received asynchronously, the specified value provides a method to associate an incoming reply to its corresponding request.

Namespace:

The WS-Addressing namespace to use in the WS-Addressing block.

Messaging System Filter

Overview

A *messaging system* is a loosely coupled, peer-to-peer facility where clients can send messages to, and receive messages from any other client. In a messaging system, a client sends a message to a messaging agent. The recipient of the message can then connect to the same agent and read the message. However, the sender and recipient of the message do not need to be available at the same time to communicate (for example, unlike HTTP). The sender and recipient need only know the name and address of the messaging agent to talk to.

The Java Messaging System (JMS) is an implementation of such a messaging system. It provides an API for creating, sending, receiving, and reading messages. Java-based applications can use it to connect to other messaging system implementations. A *JMS provider* can deliver messages synchronously or asynchronously, which means that the client can fire and forget messages or wait for a response before resuming processing. Furthermore, the JMS API ensures different levels of reliability in terms of message delivery. For example, it can ensure that the message is delivered once and only once, or at least once.

The Enterprise Gateway uses the JMS API to connect to other messaging systems that expose a JMS interface, including Oracle WebLogic Server, IBM MQSeries, JBoss Messaging, TIBCO EMS, IBM WebSphere Server, and Progress SonicMQ.

Important Note:

You must add the JMS provider JAR files to the Enterprise Gateway classpath for this filter to function correctly. If the provider's implementation is platform-specific, copy the provider JAR files to the `INSTALL_DIR/ext/PLATFORM` folder, where `INSTALL_DIR` points to the root of your product installation, and `PLATFORM` is the platform on which the Enterprise Gateway is installed (`win32`, `Linux.i386`, or `SunOS.sun4u-32`). If the provider implementation is platform-independent, you can place the JAR files in `INSTALL_DIR/ext/lib`.

Request Settings

The **Request** tab specifies properties of the request to the messaging system. You can configure the following fields:

JMS Service:

Select an existing JMS service from the drop-down list. You can configure JMS services globally on the **External Connections** tab in Policy Studio. For more details on configuring JMS services so that they can be used in this Connection filter, see the [Messaging System](#) topic.

Destination:

Enter the name of the JMS queue or topic that you want to drop messages on to.

Delivery Mode:

The Enterprise Gateway supports persistent and non-persistent delivery modes:

- **Persistent:**
Instructs the JMS provider to ensure that a message is not lost in transit if the JMS provider fails. A message sent with this delivery mode is logged to persistent storage when it is sent.
- **Non-persistent:**
Does not require the JMS provider to store the message. With this mode, the message may be lost if the JMS provider fails.

Priority Level:

You can use message priority levels to instruct the JMS provider to deliver urgent messages first. The ten levels of priority range from 0 (lowest) to 9 (highest). If you do not specify a priority level, the default level is 4. A JMS provider tries to deliver higher priority messages before lower priority ones but does not have to deliver messages in exact order of priority.

Time to Live:

By default, a message never expires. However, if a message becomes obsolete after a certain period, you may want to set an expiration time (in milliseconds). If the specified time to live value is 0, the message never expires.

Message ID:

Enter an identifier to be used as the unique identifier for the message. By default, the unique identifier is the ID assigned to the message by the Enterprise Gateway (`${id}`). However, you can use a proprietary correlation system, perhaps using MIME message IDs instead of Enterprise Gateway message IDs.

Correlation ID:

Enter an identifier for the message that the Enterprise Gateway uses to correlate response messages with the corresponding request messages. Usually, if `$id` is specified in the **Message ID** field above, it is also used here to correlate request messages with their correct response messages.

Message Type:

This drop-down list enables you to specify the type of data to be serialized and sent in the JMS message to the JMS provider. The option selected depends on what part of the message you want to send to the consumer.

For example, if you want to send the message body, select the option to format the body according to the rules defined in the [SOAP over JMS](http://www.w3.org/TR/soapjms/) [http://www.w3.org/TR/soapjms/] recommendation. Alternatively, if you wanted to serialize a list of name-value pairs to the JMS message, choose the option to create a `MapMessage`.

The following list describes the various serialization options available:

- Use `content.body` attribute to create a message in the format specified in the SOAP over Java Messaging Service recommendation:
If this option is selected, messages are formatted according to the [SOAP over JMS](http://www.w3.org/TR/soapjms/) [http://www.w3.org/TR/soapjms/] recommendation. This is the default option because, in most cases, the message body is to be routed to the messaging system. If this option is selected, a `javax.jms.BytesMessage` is created and a JMS property containing the content type (`text/xml`) is set on the message.
- Create a `MapMessage` from the `java.lang.Map` in the attribute named below:
Select this option to create a `javax.jms.MapMessage` from the Enterprise Gateway message attribute named below that consists of name-value pairs.
- Create a `BytesMessage` from the attribute named below:
Select this option to create a `javax.jms.BytesMessage` from the Enterprise Gateway message attribute named below.
- Create an `ObjectMessage` from the `java.lang.Serializable` in the attribute named below:
Select this option to create a `javax.jms.ObjectMessage` from the Enterprise Gateway message attribute named below.
- Create a `TextMessage` from the attribute named below:
Select this option to create a `javax.jms.TextMessage` from the message attribute named below.
- Use the `javax.jms.Message` stored in the attribute named below:
If a `javax.jms.Message` has already been stored in a message attribute, select this option, and enter the name of the attribute in the field below.

Attribute Name:

Enter the name of the Enterprise Gateway message attribute that holds the data that is to be serialized to a JMS message and sent over the wire to the JMS provider. The type of the attribute named here must correspond to that selected in the **Message Type** drop-down field above.

Use Shared JMS Session:

By default, each running instance of a **Messaging System** filter creates its own session (using its own thread) with the JMS provider. You can select this option to force all running instances of this filter to share the same JMS session (using a common shared thread) to the JMS provider. Reusing a shared session across multiple filter instances in this manner may result in performance degradation as each connection to the provider using the session blocks until the response (if

any) is received.

Custom Message Properties:

You can set custom properties for messages in addition to those provided by the header fields. Custom properties may be required to provide compatibility with other messaging systems. You can use message attributes as property values. For example, you can create a property called `AuthNUser`, and set its value to `${authenticated.subject.id}`. Other applications can then filter on this property (for example, only consume messages where `AuthNUser` equals `admin`).

To add a new property, click the **Add** button, and enter a name and value in the fields provided on the **Properties** dialog.

Response Settings

The **Response** tab specifies whether the Enterprise Gateway uses asynchronous or synchronous communication when talking to the messaging system. If you want the Enterprise Gateway to use asynchronous communication, select the **No response** radio button. If synchronous communication is required, you can select to read the response from either a temporary queue or from a named queue or topic.

When you select to use synchronous communication, the Enterprise Gateway waits on a message from a queue/topic from the messaging system. The Enterprise Gateway sets the `JMSReplyTo` property on each message that it sends. The value of the `JMSReplyTo` property is the queue, temporary queue, topic, or temporary topic that was selected in the **Response Type** drop-down list. It is the responsibility of the application that consumes the message from the queue to send the message back to the destination specified in `JMSReplyTo`.

The Enterprise Gateway sets the `JMSCorrelationID` property to the value of the **Correlation ID** field on the **Response** tab to correlate requests messages to their corresponding response messages. If you select to use a temporary queue or temporary topic, this is created when the Enterprise Gateway starts up.

Response Type:

Select where the response message is to be placed using one of the following options:

- **No Response:**
Select this option if you do not expect or do not care about receiving a response from the JMS provider.
- **Response from Temporary Queue:**
Select this option to instruct the JMS provider to place the response message on a temporary queue. In this case, the temporary queue is created when the Enterprise Gateway starts up. Only the Enterprise Gateway can read from the temporary queue, but any application can write to it. The Enterprise Gateway uses the value of the `JMSReplyTo` header to indicate the location where reads responses from.
- **Take Response from Queue or Topic:**
If you want the JMS provider to place response messages on a named queue or topic, select this option, and enter the name of the queue or topic in the **Destination** field below.

Destination:

If a named queue or topic is to be used, enter its name in this field.

Selector:

The entered expression specifies the messages that the consumer is interested in receiving. By using a selector, the task of filtering the messages is performed by the JMS provider instead of by the consumer. The selector is a string that specifies an expression whose syntax is based on the SQL92 conditional expression syntax. The Process only receives messages whose headers and properties match the selector.

Time Out:

The Enterprise Gateway waits a certain time period for a response to be received before it times out. If the Enterprise Gateway times out waiting for a response, this filter fails. Enter the time out value in milliseconds.

Read WS-Addressing

Overview

The WS-Addressing specification defines a transport-independent standard for including addressing information in SOAP messages. The Enterprise Gateway can read WS-Addressing information contained in a SOAP message and subsequently use this information to route the message to its intended destination.

Configuration

Complete the following fields to configure the Enterprise Gateway to read WS-Addressing information contained in a SOAP message.

Name:

Enter an appropriate name for the filter.

Address location:

Specify the name of the element in the WS-Addressing block that contains the address of the destination server to which the Enterprise Gateway routes the message. For more information on configuring XPath expressions, see the [Configuring XPath Expressions](#) topic.

By default, XPath expressions are available to extract the destination server from the `From`, `To`, `ReplyTo`, and `FaultTo` elements. Click the **Add** button to add a new XPath expression to extract the address from a different location.

Remove enclosing WS-Addressing element:

If this option is selected, the WS-Addressing element returned by the XPath expression configured above is removed from the SOAP Header when it has been consumed.

Rewrite URL

Overview

You can use the **Rewrite URL** filter to specify the path on the remote machine to send the request to. This filter normally used in conjunction with a **Static Router** filter, whose role is to supply the host and port of the remote service. For more details, see the [Static Router](#) topic.

Depending on how the Enterprise Gateway is perceived by the client, different combinations of routing filters can be used. For an introduction to using the various filters in the **Routing** category, see the topic on [Getting Started with Routing Configuration](#).

Configuration

Configure the following fields on the **Rewrite URL** filter configuration screen:

Name:

Enter an appropriate name for the filter in the **Name** field.

URL:

Enter the relative path of the Web Service in the **URL** field. The Enterprise Gateway combine the specified path with the host and port number specified in the **Static Router** filter to build up the complete URL to route to.

Alternatively, you can perform simple URL rewrites by specifying a fully qualified URL into the **URL** field. You can then use a **Dynamic Router** to route the message to the specified URL.

Save to File

Overview

The **Save to file** filter enables you to write the current message contents to a file. For example, you can save the message contents to a file in a directory where it can be accessed by an external application. This can be used to quarantine messages to the file system for offline examination. This filter can also be useful when integrating legacy systems. Instead of making drastic changes to the legacy system by adding an HTTP engine, the Enterprise Gateway can save the message contents to the file system, and route them on over HTTP to another back-end system.

Configuration

To configure the **Save to file** filter, specify the following fields:

Name	Name of the filter to be displayed in a circuit. Defaults to Save to file .
File name	Enter the name of the file that the content is saved to. You can specify this using a property, which is expanded to the specified value at runtime. Defaults to <code>\${id}.out</code> .
Directory	Enter the directory that the file is saved to. You can specify this using a property, which is expanded to the specified value at runtime. Defaults to <code>\${VINSTDIR}/message-archive</code> , where <code>VINSTDIR</code> is the root of your Enterprise Gateway installation.
Maximum number of files in directory	Enter the maximum number of files that can be saved in the directory. Defaults to 500.
Maximum file size	Enter the maximum file size in MB. Defaults to 1000.
Include HTTP Headers	Select whether to include HTTP headers in the file. HTTP headers are not included by default.

SMTP Routing

Overview

You can use the **SMTP Routing** filter to relay messages to an email recipient using a previously configured SMTP server. For details on configuring SMTP servers, see the [External Connections](#) topic.

General Settings

Complete the following general settings:

Name:

Specify a descriptive name for this SMTP Server in this field.

SMTP Server Settings:

Select a previously configured global SMTP server from the drop-down list. You can configure an SMTP server on the **External Connections** tab by right-clicking the **SMTP Server** node, and selecting **Add an SMTP Server**. For more details, see the [External Connections](#) topic.

Message Settings

Complete the following fields in the **Message Settings** section of the screen:

To:

Enter the email address of the recipient of the messages.

From:

Enter the email address that you wish to appear in the **From** field of the email messages.

Subject:

Enter some text as the **Subject** of the email messages.

Static Router

Overview

The Enterprise Gateway uses the information configured in the **Static Router** filter to connect to a machine that is hosting a Web Service. You should use the **Static Router** filter in conjunction with a **Rewrite URL** filter to specify the path to send the message to on the remote machine. For more details, see the [Rewrite URL](#) topic.

Depending on how the Enterprise Gateway is perceived by the client, different combinations of routing filters can be used. For an introduction to using the various filters in the **Routing** category, see the topic on [Getting Started with Routing Configuration](#).

Configuration

You must configure the following fields must be configured on the **Static Router** configuration screen:

Name:

Enter a name for the filter.

Host:

Enter the host name or IP address of the remote machine that is hosting the destination Web Service.

Port:

Enter the port on which the remote service is listening.

HTTP:

Select this option if the Enterprise Gateway should send the message to the remote machine over plain HTTP.

HTTPS:

Select this option if the Enterprise Gateway should send the message to the remote machine over a secure channel using SSL. You can use a **Connection** filter to configure the Enterprise Gateway to mutually authenticate to the remote system.

TIBCO Rendezvous Routing

Overview

TIBCO Rendezvous® is a low latency messaging product for real-time high throughput data distribution applications. It facilitates the exchange of data between applications over the network.

A TIBCO Rendezvous *daemon* runs on each participating node on the network. All data sent to and read by each application passes through the daemon. Enterprise Gateway uses the TIBCO Rendezvous API to communicate with a TIBCO Rendezvous daemon running locally (by default) to send messages to other TIBCO Rendezvous programs.

You can configure the **TIBCO Rendezvous** filter to route messages (using a TIBCO Rendezvous daemon) to other TIBCO Rendezvous programs. This filter is found in the Routing category of filters.

Configuration

Configure the following fields to route messages to other TIBCO Rendezvous programs:

Name:

Enter an appropriate name for this filter in the field provided.

TIBCO Rendezvous Daemon to Use:

Select a previously configured **TIBCO Daemon** from the tree by selecting the checkbox next to the daemon. The Enterprise Gateway sends messages to the specified **TIBCO Rendezvous Subject** on this daemon. You can configure **TIBCO Daemons** globally on the **External Connections** tab in the Policy Studio. For more information, see the [TIBCO Daemon Configuration](#) topic.

TIBCO Rendezvous Subject:

The message is sent with the subject entered here meaning that all other TIBCO daemons on the network that have subscribed to this subject name will receive the message. The subject name comprises a series of elements, including properties (for example, *), separated by dot characters, for example:

- `news.sport.soccer`
- `news.sport.*`
- `FINANCE.ACCOUNT.SALES`

For more information on the subject name syntax, see the TIBCO Rendezvous documentation.

Field Name:

Click the **Add** button to add details about a particular field to add to the message. On the **Message Field Definition** dialog, enter the name of the field to send in the message in the **Field Name** field, and complete the remaining fields.

Type:

Select the data type of the value specified in either of the following fields:

Set value to the following constant value:

You can explicitly set this value by entering it here.

Set value to the object found in the following attribute:

If you would like to dynamically populate the field value using the contents of a message attribute, you can select this attribute from this drop-down list. At runtime, the contents of the message attribute are placed into the message that is sent to TIBCO Rendezvous.

TIBCO Enterprise Messaging System Routing Filter

Overview

TIBCO Enterprise Messaging System™ (EMS) provides a distributed message bus with native support for JMS (Java Messaging Service) and TIBCO Rendezvous, along with other protocols.

In general, TIBCO EMS clients *produce* messages and send them to the TIBCO EMS Server. Similarly, TIBCO EMS clients can connect to the TIBCO EMS Server and declare an interest in a particular queue or topic on that server. In doing so, it can *consume* messages that have been produced by another TIBCO EMS client.

The Enterprise Gateway can act as a message producer by sending messages to the TIBCO EMS Server and as a message consumer by listening on a queue or topic at the server. The TIBCO EMS Routing filter can be used as a message producer in this manner.

Connection

The list of all globally configured **TIBCO EMS Connections** are listed on this tab. You can configure TIBCO EMS Connections on the **External Connections** tab in the Policy Studio. For more information on configuring TIBCO EMS Connections, see the [TIBCO Enterprise Messaging System Connection](#) topic.

Select a previously configured TIBCO EMS Connection from the list. Messages are sent to this TIBCO EMS Connection and are dropped on the queue or topic specified on the **Request** tab.

Request

The **Request** tab is used to configure properties of the request to the messaging system. You can configure the following fields:

Destination Type:

You must specify whether the name specified in the **Queue or Topic Name** field below is a `Queue` or `Topic`.

Queue or Topic Name:

Enter the name of the queue or topic that you want to drop messages on to.

Delivery Mode:

The Enterprise Gateway supports the following delivery modes:

- **Persistent:**
Instructs the TIBCO EMS Server to ensure that a message is not lost in transit if the server fails. A message sent with this delivery mode is logged to persistent storage when it is sent.
- **Non-persistent:**
Does not require the TIBCO EMS Server to store the message. With this mode, the message may be lost if the server fails.
- **Reliable:**
When using reliable mode the TIBCO EMS Server never sends an acknowledgment or confirmation receipt back to the producer. This greatly decreases the volume of traffic on the network and can result in improved performance.

Priority Level:

You can use message priority levels to instruct the TIBCO EMS server to deliver urgent messages first. The ten levels of priority range from 0 (lowest) to 9 (highest). If you do not specify a priority level, the default level is 1. The TIBCO EMS Server tries to deliver higher priority messages before lower priority ones but does not have to deliver messages in exact order of priority.

Time to Live:

By default, a message never expires. However, if a message becomes obsolete after a certain period, you may want to set an expiration time (in milliseconds). If the specified time to live value is 0, the message never expires.

Message ID:

Enter an identifier to be used as the unique identifier for the message. By default, the unique identifier is the ID assigned to the message by the Enterprise Gateway (`${id}`). However, you can use a proprietary correlation system, perhaps using MIME message IDs instead of Oracle message IDs.

Correlation ID:

Enter an identifier for the message that the Enterprise Gateway uses to correlate response messages with the corresponding request messages. Usually, if `$id` is specified in the **Message ID** field above, it is also used here to correlate request messages with their correct response messages.

Message Type:

This enables you to specify the type of data to be serialized and sent in the message to the TIBCO EMS Server. The option selected depends on what part of the message you want to send to the consumer.

For example, if you wish to send the message body you should select the option to format the body according to the rules defined in the [SOAP over JMS](http://www.w3.org/TR/soapjms/) [http://www.w3.org/TR/soapjms/] recommendation. Alternatively, if you wish to serialize a list of name-value pairs to the message, choose the option to create a `MapMessage`.

The following list describes the various serialization options available:

- Use `content.body` attribute to create a message in the format specified in the SOAP over Java Messaging Service recommendation:
If this option is selected, messages are formatted according to the [SOAP over JMS](http://www.w3.org/TR/soapjms/) [http://www.w3.org/TR/soapjms/] recommendation. This is the default option.
- Create a `MapMessage` from the `java.lang.Map` in the attribute named below:
Select this option to create a `javax.jms.MapMessage` from the Oracle message attribute (named below) that consists of name-value pairs.
- Create a `ByteMessage` from the attribute named below:
Select this option to create a `javax.jms.ByteMessage` from the Oracle message attribute named below.
- Create an `ObjectMessage` from the `java.lang.Serializable` in the attribute named below:
Select this option to create a `javax.jms.ObjectMessage` from the Oracle message attribute named below.
- Create a `TextMessage` from the attribute named below:
A `javax.jms.TextMessage` can be created from the message attribute named below by selecting this option.

Attribute Name:

Enter the name of the Oracle message attribute that holds the data to be serialized to a JMS message and sent over the wire to the TIBCO EMS Server. The type of the attribute named here must correspond to that selected in the **Message Type** field above.

Custom Message Properties:

You can set custom properties for messages in addition to the standard JMS header fields. Custom properties may be useful to pass additional information to the TIBCO EMS Server. You can use message attributes as property values. For example, you can create a property called `AuthNUser`, and set its value to `${authenticated.subject.id}`. When this message is routed to the specified queue or topic, other consumers can then filter on this property (for example, only consume messages where `AuthNUser` equals `admin`).

Response

The **Response** tab is used to configure whether the Enterprise Gateway should use asynchronous or synchronous communication when talking to the messaging system. If the Enterprise Gateway is to use asynchronous communication, select **No Response** from the **Response** drop-down list. If synchronous communication is required, you must configure where to read the response from the TIBCO EMS Server.

When synchronous communication is selected, the Enterprise Gateway waits on a message from a queue/topic from the

TIBCO EMS Server. The Enterprise Gateway sets the `JMSReplyTo` property on each message that it sends. The value of the `JMSReplyTo` property is the queue, temporary queue, topic, or temporary topic that was selected in the **Response** drop-down list. It is the responsibility of the application that consumes the message from the queue to send the message back to the destination specified in the `JMSReplyTo` property.

The Enterprise Gateway sets the `JMSCorrelationID` property to the value of the **Correlation ID** field on the **Response** tab to correlate requests messages to their corresponding response messages. If the user has selected to use a temporary queue or temporary topic, this is created when the Enterprise Gateway starts up.

Response:

Select where to read the response message from. The following options are available:

- **No Response:**
Select this option if you do not expect or require a response from the TIBCO EMS Server.
- **Response in Queue Named Below:**
If you want the TIBCO EMS Server to place response messages into a named queue, select this option, and enter the name of the queue in the field below.
- **Response in Temporary Queue:**
You can also instruct the TIBCO EMS Server to place response messages on a temporary queue from which the Enterprise Gateway can pick them up.
- **Response in Topic Named Below:**
Select this option to tell the TIBCO EMS Server to place response messages in the topic named in the field below.
- **Response in Temporary Topic:**
If you want to read response messages from a temporary topic, select this option.

Note:

When a temporary destination is selected, this destination is created at start-up of the Enterprise Gateway. Only the Enterprise Gateway can read from the temporary destination, however, any application can write to it. The Enterprise Gateway uses the value of the `JMSReplyTo` header to indicate the location where it reads responses from.

Reply Topic/Queue Name:

If you have selected a named queue or topic (*not* a temporary queue or topic) from the **Response** field above, enter its name here.

Time Out:

The Enterprise Gateway waits a certain time period for a response to be received before times out. If the Enterprise Gateway does time out waiting for a response, this filter fails. Enter the time out value in milliseconds.

TIBCO Enterprise Messaging System Connection

Overview

TIBCO Enterprise Messaging System™ (EMS) provides a distributed message bus with support for JMS (Java Messaging Service) and TIBCO Rendezvous, along with other protocols.

In general, TIBCO EMS clients *produce* messages and send them to the TIBCO EMS Server. Similarly, TIBCO EMS clients can connect to the TIBCO EMS Server and declare an interest in a particular queue or topic on that server. In doing so, it can *consume* messages that have been produced by another TIBCO EMS client.

The Enterprise Gateway can act as a message producer by sending messages to the TIBCO EMS Server and as a message consumer by listening on a queue or topic at the server. Both configurations require a connection to the TIBCO EMS Server. For more information on consuming and producing messages to and from TIBCO EMS, see the following topics:

- [TIBCO Integration](#)
- [TIBCO EMS Consumer](#)
- [TIBCO EMS Routing Filter](#)

This topic describes how to configure a connection to an TIBCO EMS Server. For more detailed information on configuring TIBCO EMS Connections, see the TIBCO EMS documentation.

Configuration

The TIBCO EMS Connection is configured globally so that it can be referenced when configuring TIBCO EMS consumers and TIBCO EMS producers in the Enterprise Gateway. To configure a global connection to an TIBCO EMS Server, right-click the **TIBCO Enterprise Messaging Service Connections** node on the **External Connections** tab in the Policy Studio, and select **Add a TIBCO EMS Connection** from the context menu. The remainder of this topic describes how to configure the tabs and fields on the **TIBCO Enterprise Messaging System Connection** dialog.

Before configuring the following fields you must enter a name for this TIBCO EMS Connection in the **Name** field. This connection is then available when configuring a TIBCO EMS Consumer and when configuring a TIBCO EMS Routing filter.

General Tab:

The following fields are available on the **General Tab**:

Server URL:

Enter the full URL of the TIBCO EMS Server in this field, for example `tcp://hostname:7222` for non-SSL connections or `ssl://server:7243` for SSL-enabled TIBCO EMS Servers.

User Name:

Enter a username to use when the Enterprise Gateway connects to the TIBCO EMS Server.

Password:

Enter the password for this user.

SSL Tab:

The following tabs and fields are available on the **SSL Tab**:

Limit the use of SSL to improve performance:

If this option is selected, SSL is only used for establishing (mutual) authentication with the TIBCO EMS Server, which takes place during the initial SSL handshaking process. When the channel is set up, data sent over this channel is sent

in the clear and is not encrypted like in a typical SSL session.

Enable client verification of the host certificate or host name:

Select this option if you want to compare the Common Name (cn) X.509 attribute of the Distinguished Name in the TIBCO EMS Server's certificate. Typically, the SSL handshake requires that the common name in the host's certificate matches the name of the host machine. For example, to trust the certificate associated with the `www.abc.com` site, the certificate must have the common name attribute set to this name (`cn=www.abc.com`). If you wish to perform this check on the TIBCO EMS Server's certificate presented to the Enterprise Gateway during SSL setup, select this setting.

Expected Host Name:

In cases where the common name in the certificate is *not* the same as the host machine, you can override the default validation by specifying a host name that you expect instead of the host given in the common name of the server's certificate.

For example, a generic TIBCO EMS Server certificate is issued for testing purposes, and this certificate is created with a common name of `server` (`cn=server`). Now, assume that you want to create an SSL session with a TIBCO EMS Server running on a machine that is called `host`.

The default client verification of the host name setting checks to make sure that the host on which the TIBCO EMS Server is running is called `server` because this is what is in the common name of the certificate. However, the host name of this machine is `host`, and so this check fails.

In such cases, you must override the default host checking behavior by specifying the *expected* host name in this field. In this case, enter `host` in the **Expected Host Name** field.

Cipher suites to be used:

Specify the OpenSSL cipher suites that the Enterprise Gateway supports. The ciphers are negotiated during the SSL handshake with the TIBCO EMS Server so that the strongest and most secure ciphers that are common to both parties are used.

Trusted Certificates Tab:

You can select the CA (Certificate Authority) certificates that you consider trusted for setting up the connection to the TIBCO EMS Server on this tab.

The TIBCO EMS Server's certificate can be explicitly trusted by importing it into the Certificate Store and selecting it in the list. Alternatively, in a solution more typical for a Public Key Infrastructure, the CA certificate that issued the TIBCO EMS Server's certificate is imported into the Certificate Store and is selected in the list. In this case, a chain of trust is established because all certificates issued by the CA are implicitly trusted if the CA is considered trusted.

Client Identity Tab:

If you want to configure mutual authentication to the TIBCO EMS Server you must select a client certificate from the list that the Enterprise Gateway can use to authenticate to the TIBCO EMS Server. For the SSL channel to be established successfully, the TIBCO EMS Server must trust the client certificate selected here.

Important Note:

If the selected client certificate has been issued by a CA (it is not self-signed), the certificate of this CA *must* be imported into the Trusted Certificate Store. If a chain of certificates exists (for example, the client certificate was issued by an intermediary CA, which was issued by the root CA), all intermediary CA certificates must be imported into the Certificate Store.

Wait for Response Packets

Overview

Packet Sniffers are a type of Passive Service. Rather than opening up a TCP port and *actively* listening for requests, the Packet Sniffer *passively* reads data packets off the network interface. The Sniffer assembles these Packets into complete messages that can then be passed into an associated policy.

Because the Packet Sniffer operates passively (does not listen on a TCP port) and transparently to the client, it is most useful for monitoring and managing Web Services. For example, you can deploy the Sniffer on a machine running a Web Server acting as a container for Web Services. Assuming that the Web Server is listening on TCP port 80 for traffic, the Packet Sniffer can be configured to read all packets destined for port 80 (or any other port, if necessary). The packets can then be marshaled into complete HTTP/SOAP messages by the Sniffer and passed into a policy that, for example, logs the message to a database.

Packet Sniffer Configuration

Because Packet Sniffers are mainly used as passive monitoring agents, they are usually created in their own Service Group. For example, you can create a new group for this purpose by right-clicking the Process on the **Services** tab in the Policy Studio, selecting **Add Service Group**, and then entering `Packet Sniffer Group` on the **Add Service Group** dialog.

You can then add a Relative Path Service to this Group by right-clicking the `Packet Sniffer Group`, and selecting **Add Relative Path**. Enter a path in the field provided, and select the policy that you want to dispatch messages to when the Packet Sniffer detects a request for this path (after it assembles the packets). For example, if the Relative Path is configured as `/a`, and the Packet Sniffer assembles packets into a request for this path, the request is dispatched to the policy selected in the Relative Path Service.

Finally, you can add the Packet Sniffer by right-clicking the `Packet Sniffer Group` node, selecting **Packet Sniffer -> Add**. Complete the following fields on the **Packet Sniffer** dialog:

Device to Monitor:

Enter the name or identifier of the network interface that the Packet Sniffer monitors. The default entry is `any`, but it is this is only valid on Linux. On UNIX-based systems, network interfaces are usually identified using names like `eth0`, `eth1`, and so on. On Windows, these names are more complicated (for example, `\Device\NPF_{00B756E0-518A-4144 ... }`).

Filter:

You can configure the Packet Sniffer to only intercept certain types of packets. For example, it can ignore all UDP packets, only intercept packets destined for port 80 on the network interface, ignore packets from a certain IP address, listen for all packets on the network, and so on.

The Packet Sniffer uses the **libpcap** library filter language to achieve this. This language has a complicated but powerful syntax that enables you to *filter* what packets are intercepted, and what packets are ignored. As a general rule, the syntax consists of one or more expressions combined with conjunctions, such as `and`, `or`, and `not`. The following table lists a few examples of common filters and explains what they filter:

Filter Expression	Description
<code>port 80</code>	Captures only traffic for the HTTP Port (i.e. 80).
<code>host 192.168.0.1</code>	Captures traffic to and from IP address 192.168.0.1.
<code>tcp</code>	Captures only TCP traffic.
<code>host 192.168.0.1 and port 80</code>	Captures traffic to and from port 80 on IP address 192.168.0.1.

<code>tcp portrange 8080-8090</code>	Captures all TCP traffic destined for ports from 8080 through to 8090.
<code>tcp port 8080 and not src host 192.168.0.1</code>	Captures all TCP traffic destined for port 8080 but not from IP address 192.168.0.1.

The default filter of `tcp` captures all TCP packets arriving on the network interface. For more information on how to configure filter expressions like these, see the [tcpdump man page](http://www.tcpdump.org/tcpdump_man.html) [http://www.tcpdump.org/tcpdump_man.html].

Promiscuous Mode:

When listening in promiscuous mode, the Packet Sniffer captures all packets on the same Ethernet network, regardless of whether the packets are addressed to the network interface that the Sniffer is monitoring.

Sniffing Response Packets

The Enterprise Gateway can capture both incoming and outgoing packets when it is listening passively (not opening any ports) on the network interface. For example, a Web Service is deployed in a web server that listens on port 80. The Enterprise Gateway can be installed on the same machine as the web server. It is configured *not* to open any ports and to use a Packet Sniffer to capture all packets destined for TCP port 80.

When packets arrive on the network interface that are destined for this port, they are assembled by the Packet Sniffer into HTTP messages and passed into the configured policy. Typically, this policy logs the message to an audit trail, and so usually consists of just a **Log Message** filter.

Assuming that you also want to log response messages passively, as is typically required for a complete audit trail, you can use the **Wait for Response Packets** filter to correlate response packets with their corresponding requests. The **Wait for Response Packets** filter assembles the response messages into HTTP messages and can then log them again using the **Log Message Payload** filter. The following circuit logs both request and response messages captured transparently by the Packet Sniffer:

You can see from the circuit that the first logging filter logs the *request* message. By this stage, the Packet Sniffer has assembled the request packets into a complete HTTP request, and this is what is passed to the **Log Request Message** filter.

The **Assemble response packets** filter is a **Wait for Response Packets** filter that assembles response packets into complete HTTP response messages and passes them to the **Log Response Message** filter, which logs the complete response message. More information on the **Log Message Payload** filter is available in the [Log Message Payload](#) topic.

Proxy Servers

Overview

You can configure settings for individual proxy servers on the **External Connections** tab, which you can then specify at the filter level (in the **Connection** and **Connect To URL** filters). When configured, the filter connects to the HTTP proxy server, which in turn routes the message on to the destination server named in the request URI. For more details, see the [Connection](#) and [Connect To URL](#) topics.

These proxy server settings are different from the global proxy settings in the **Preferences** dialog in the Policy Studio, which apply only when downloading WSDL, XSD, and XSLT files from the Policy Studio. For more details, see the [Policy Studio Preferences](#) topic.

Configuration

To configure a proxy server on the **External Connections** tab, right-click the **Proxy Servers** node, and select **Add a Proxy Server**. You can configure the following settings in the dialog:

<i>Proxy Server Setting</i>	<i>Description</i>
Name	Unique name or alias for these proxy server settings.
Host	Host name or IP address of the proxy server.
Port	Port number on which to connect to the proxy server.
Username	Optional user name when connecting to the proxy server.
Password	Optional password when connecting to the proxy server.
Scheme	Specifies whether the proxy server uses the HTTP or HT-TPS transport. Defaults to HTTP.

DSS Signature Generation Service

Overview

The filter allows the Enterprise Gateway to generate XML Signatures "as a service" according to the OASIS DSS (Digital Signature Services) specification. The DSS specification describes how a client can send a message containing an XML Signature to a DSS Signature Web Service that can sign the (relevant parts of the) message and return the resulting XML Signature to the client.

The advantage of this approach is that the Signature generation code is abstracted away from the logic of the Web Service and does not have to be coded into the Web Service. Furthermore, in an SOA (Services Oriented Architecture), a centralized DSS server provides a single implementation point for all XML Signature related services, which can then be accessed by all Services running within the SOA. This represents a much more manageable solution than one in which the security layer is actually coded into each Web Service.

Configuration

Complete the following fields to configure the **DSS Signature Generation Service** filter.

Name:

Enter a descriptive name for the filter in this field.

Signing Key:

Click the **Signing Key** button to select a private key from the Certificate Store. This key will be used to perform the signing operation.

DSS Signature Verification Web Service

Overview

The filter allows the Enterprise Gateway to verify XML Signatures "as a service" according to the OASIS DSS (Digital Signature Services) specification. The DSS specification describes how a client can send a message containing an XML Signature to a DSS Signature verification Web Service that can verify the Signature and return the result of the verification to the client.

The advantage of this approach is that the Signature verification code is abstracted away from the logic of the Web Service and does not have to be coded into the Web Service. Furthermore, in an SOA (Services Oriented Architecture), a centralized DSS server provides a single implementation point for all XML Signature related services, which can then be accessed by all Services running within the SOA. This represents a much more manageable solution that one in which the security layer is actually coded into each Web Service.

Configuration

Complete the following fields to configure the **DSS Signature Verification Web Service** filter.

Name:

Enter a descriptive name for the filter in this field.

Find Signing Key:

The public key to be used to verify the signature can be retrieved from one of the following locations:

- **KeyInfo in Message:**
The verification certificate can be located via the <KeyInfo> block within the XML Signature. For example, the certificate could be contained within a <BinarySecurityToken> element in a WSSE Security header. The <KeyInfo> section of the XML Signature can then reference this BinarySecurityToken. The Enterprise Gateway can automatically resolve this reference in order to locate the certificate that contains the public key necessary to perform the signature verification.
- **Message Attribute:**
The certificate used to verify the signature can be extracted from the message attribute specified here. The certificate must have been placed into this attribute by a predecessor of the **DSS Signature Verification Web Service** filter.
- **Certificate in LDAP:**
The certificate used to verify the Signature can be retrieved from an LDAP directory. Select a previously configured LDAP directory from the dropdown or add a new one by clicking on the **Add/Edit** button. For more information on configuring LDAP directories, please refer to the [LDAP Configuration](#) help page.
- **Certificate in Store:**
Finally, the verification certificate can be selected from the Certificate Store. Click the **Select** button to view the certificate that have been added to the store. Select the verification certificate by checking the checkbox next to it in the table.

Encrypt and Decrypt Web Services

Overview

This filter allows the Enterprise Gateway to act as an XML *encrypting* Web Service, where clients can send up XML blocks to the Enterprise Gateway that are required to be encrypted. The Enterprise Gateway can then encrypt the XML data, replacing it with <EncryptedData> blocks in the message. The encrypted content is then returned to the client.

Similarly, the Enterprise Gateway can act as an XML *decrypting* Web Service, where clients can send up <EncryptedData> blocks to the Enterprise Gateway, which can then decrypt them and return the plain-text data back to the client.

By deploying the Enterprise Gateway as a centralized encryption/decryption service, clients distributed throughout an SOA (Services Oriented Architecture) can abstract out the security layer from their core business logic. This simplifies the logic of the client applications and makes the task of managing and configuring the security aspect a lot simpler since it is centralized.

Furthermore, the Enterprise Gateway's XML and cryptographic acceleration capabilities ensure that the process of encrypting and decrypting XML messages - a task that involves some very CPU-intensive operations - is performed at optimum speed.

Configuration

To configure both the **Encrypt Web Service** and **Decrypt Web Service** filters you simply need to enter a descriptive name for the filter in the **Name** field.

STS Web Service

Overview

This filter can be used to expose a Security Token Service, allowing clients to obtain security tokens for use within a SOA (Services Oriented Architecture) network.

Configuration

Complete the following field to configure the **STS Web Service** filter.

Name:

Enter a descriptive name for the filter in this field.

Consume WS-Trust Message

Overview

You can configure the Enterprise Gateway to consume various types of WS-Trust messages, including `RequestSecurityToken` (RST), `RequestSecurityTokenResponse` (RSTR), and `RequestSecurityTokenResponseCollection` (RSTRC) messages.

For more information on the various types of WS-Trust messages and their semantics and format, please see the WS-Trust specification.

Consume WS-Trust Message Types

The Enterprise Gateway can consume the following types of WS-Trust messages. Select the appropriate message type based on your requirements:

- **RST: RequestSecurityToken**
The RST message contains a request for a single token to be issued by the Security Token Service (STS).
- **RSTR: RequestSecurityTokenResponse**
The RSTR message is sent in response to an RST message from a token requestor. It contains the token issued by the STS.
- **RSTRC: RequestSecurityTokenResponseCollection**
The RSTRC message contains an RSTR (containing a single issued token) for each RST that was received in an RSTC message.

Message Consumption

The configuration options available in this section enable you to extract various parts of the WS-Trust message and store them in Oracle message attributes for use in subsequent filters.

Extract Token:

Extracts a `<RequestedSecurityToken>` from the WS-Trust message and stores it in a message attribute. Select the expected value of the `<TokenType>` element in the `<RequestSecurityToken>` block. The default URI is `http://schemas.xmlsoap.org/ws/2005/02/sc/sct`.

Extract BinaryExchange:

Extracts a `<BinaryExchange>` token from the message and stores it in a message attribute. You should select the `ValueType` of the token from the drop-down list.

Extract Entropy:

The client can provide its own key material (entropy) that the token issuer may use when generating the token. The issuer can use this entropy as the key itself, it can derive another key from this entropy, or it can choose to ignore the entropy provided by the client altogether in favor of generating its own entropy.

Extract RequestedProofToken:

Select this option if you want to extract a `<RequestedProofToken>` from the WS-Trust message and store it in a message attribute for later use. You must select the type of the token (`encryptedKey` or `computedKey`) from the drop-down list.

Extract CancelTarget:

You can select this option to extract a `<CancelTarget>` block from the WS-Trust message and store it in a message attribute.

Extract RequestedTokenCancelled:

You can select this option to extract a `<RequestedTokenCancelled>` block from the WS-Trust message and store it in a message attribute.

Match Context ID:

Select this option if you wish to correlate the response message from the STS with a specific request message. The `Context` attribute on the `RequestSecurityTokenResponse` message is compared to the value of the `ws.trust.context.id` message attribute, which contains the context ID of the current token request.

Extract Lifetime:

Select this option to remove the `<Lifetime>` elements from the WS-Trust token.

Extract Authenticator:

Select this option to extract the `<Authenticator>` from the WS-Trust token and store it in a message attribute.

Advanced

The following fields can be configured on the **Advanced** tab:

WS-Trust Namespace:

Enter the WS-Trust namespace that you expect all WS-Trust elements to be bound to in tokens that are consumed by this filter. The default namespace is `http://schemas.xmlsoap.org/ws/2005/02/trust`.

Cache Security Context Session Key:

Select the cache to store the security context session key. The session key (the value of the `security.context.session.key` attribute), is cached using the value of the `security.context.token.unattached.id` message attribute as the key into the cache.

Lifetime of ComputedKey:

The settings in this section enable you to add a timestamp to the extracted `computedKey` using the values specified in the `<Lifetime>` element. This section is enabled only after selecting the **Extract RequestedProofToken** checkbox above, selecting the `computedKey` option from the associated dropdown list, and finally by selecting the **Extract Lifetime** checkbox. Configure the following fields in this section:

- **Add Lifetime to ComputedKey:**
Adds the `<Lifetime>` details to the `security.context.session.key` message attribute. This enables you to check the validity of the key every time it is used against the details in the `<Lifetime>` element.
- **Format of Timestamp:**
Specify the format of the timestamp using the Java date and time pattern settings.
- **Timezone:**
Select the appropriate time zone from the drop-down list.
- **Drift:**
To allow for differences in the clock times on the machine on which the WS-Trust token was generated and the machine running the Enterprise Gateway, you can enter a drift time here. The drift time allows for differences in the clock times on these machines and is used when validating the timestamp on the `computedKey`.

Verify Authenticator Using:

You can verify the authenticator using either the **Generated** or **Consumed** message. In either case you should select the appropriate type of WS-Trust message from the available options.

Create WS-Trust Message

Overview

You can configure the Enterprise Gateway to create various types of WS-Trust messages. The Enterprise Gateway can act both as a WS-Trust client when generating a `RequestSecurityToken` (RST) message, but also as a WS-Trust service, or Security Token Service (STS), when generating `RequestSecurityTokenResponse` (RSTR) and `RequestSecurityTokenResponseCollection` (RSTRC) messages.

A token requestor generates an RST message and sends it to the STS, which generates the required token and returns it in an RSTR message. If several tokens are required, the requestor can send up multiple RST messages in a single `RequestSecurityTokenCollection` (RSTC) request. The STS generates an RSTR for each RST in the RSTC message and returns them all in batch mode in an RSTRC message.

For more information on the various types of WS-Trust messages and their semantics and format, please see the WS-Trust specification.

Create WS-Trust Message Type

The **Create WS-Trust Message** filter can create the following types of WS-Trust message. Select the appropriate message type based on your requirements:

- **RST: RequestSecurityToken**
The RST message contains a request for a single token to be issued by the STS.
- **RSTR: RequestSecurityTokenResponse**
The RSTR message is sent in response to an RST message from a token requestor. It contains the token issued by the STS.
- **RSTRC: RequestSecurityTokenResponseCollection**
The RSTRC message contains an RSTR (containing a single issued token) for each RST that was received in an RSTC message.

Message Creation

The settings on this tab specify characteristics of the WS-Trust message. The following fields are available:

Insert Token Type:

Select the type of token requested from the drop-down list. The type of token selected here is returned in the response from the STS. By default, the Security Token Context type is used, which is identified by the URI `http://schemas.xmlsoap.org/ws/2005/02/sc/sct`.

Binary Exchange:

You can use a `<BinaryExchange>` when negotiating a secure channel that involves the transfer of binary blobs as part of another security negotiation protocol (for example, SPNEGO). The contents of the blob are always Base64-encoded to ensure safe transmission.

Select the **Binary Exchange** option if you wish to use a negotiation-type protocol for the exchange of keys, such as SPNEGO. The URI selected in the **Value Type** field identifies the type of the negotiation in which the blob is used. The URI is placed in the `ValueType` attribute of the `<BinaryExchange>` element.

Entropy:

The client can provide its own key material (entropy) that the token issuer may use when generating the token. The issuer can use this entropy as the key itself, it can derive another key from this entropy, or it can choose to ignore the entropy provided by the client altogether in favor of generating its own entropy.

Select this option to generate some entropy, which is included in the `<wst:entropy>` element of the `<wst:RequestSecurityToken>` block.

Insert Key Size:

The client can request the key size (in number of bits) required in a `<RequestSecurityToken>` request. However, the WS-Trust token issuer does not have to use the requested key size. It is merely intended as an indication of the strength of security required. The default request key size is 256 bits.

Insert Lifetime:

Select this option to insert a `<Lifetime>` element into the WS-Trust message. Use the associated fields to specify when the message expires. The lifetime of the WS-Trust message is expressed in terms of `<Created>` and `<Expires>` elements.

Lifetime Format:

The specified date/time pattern string determines the format of the `<Created>` and `<Expires>` elements. The default format is `yyyy-MM-dd'T'HH:mm:ss.SSS'Z'`, which can be altered if necessary. For more details on how to use this format, see the Javadoc for the `java.text.SimpleDateFormat` Java class in the [Java Platform, Standard Edition 6 API Specification](http://java.sun.com/javase/6/docs/api/index.html) [http://java.sun.com/javase/6/docs/api/index.html].

Insert RequestedTokenCancelled:

Select this option to insert a `<RequestedTokenCancelled>` element into the generated WS-Trust message.

RST Creation

The following configuration fields specify the way in which a WS-Trust RST message is created:

Insert Request Type:

You can create two types of RST message. Select one of the following request types from the drop-down list:

- **Issue:** This type of RST message is used to request the STS to *issue* a token for the requestor.
- **Cancel:** This type of RST message is used to cancel a specific token.

Insert Key Type:

Select this option to insert the key type into the RST WS-Trust message.

Insert Computed Key Algorithm:

Select this option to insert the computed key algorithm into the message.

Insert Endpoint Reference:

Select this option and enter a suitable endpoint if you want to include an endpoint reference in the RST message.

RSTR Creation

The following configuration fields determine the way in which a WS-Trust RSTR message is created:

Insert RequestedProofToken:

Select this checkbox to insert a `<RequestedProofToken>` element into the generated WS-Trust message. The type of this token can be set to either `computedKey` or `encryptedKey` using the associated drop-down list.

Insert Authenticator:

Select this option to insert an authenticator into the RSTR message.

Advanced Settings

This section enables you to configure certain advanced aspects of the SOAP message that is sent to the WS-Trust Service.

WS-Trust Namespace:

Enter the WS-Trust namespace to bind all WS-Trust elements to in this field. The default namespace is `http://schemas.xmlsoap.org/ws/2005/02/trust`.

WS-Addressing Namespace:

Select the WS-Addressing namespace version to use in all created WS-Trust messages.

WS-Policy Namespace:

Select the appropriate WS-Policy namespace from the drop-down list. The selected version selected can affect the ordering of tokens that are inserted into the WS-Security header of the SOAP message.

SOAP Version:

Select the SOAP version to use when creating the WS-Trust message.

Overwrite SOAP Method:

Select this option if you want the WS-Trust token to overwrite the SOAP method in the request. In this case, the token appears as a direct child of the SOAP Body element. You should use this option if you wish to preserve the contents of the SOAP Header, if present.

Overwrite SOAP Envelope:

Select this option if you want the generated WS-Trust message to form the entire contents of the message. In other words, the generated WS-Trust message replaces the original SOAP request.

Content-Type:

Specify the HTTP content-type of the WS-Trust message. For example, for Microsoft Windows Communication Foundation (WCF), you should use `application/soap+xml`.

Generate Authenticator Using:

You can verify the authenticator using the **Generated** or **Consumed** message. In either case, you should select the appropriate type of WS-Trust message from the available options.

Writing a Custom Filter using the Oracle Enterprise Gateway SDK

Tutorial Overview

Oracle Enterprise Gateway exposes several powerful APIs as part of its Software Development Kit (SDK) to enable users to build their own bespoke message filters. Users can leverage Oracle Enterprise Gateway's pluggable and extensible architecture to enhance the message processing capabilities of the Oracle Enterprise Gateway core processing engine.

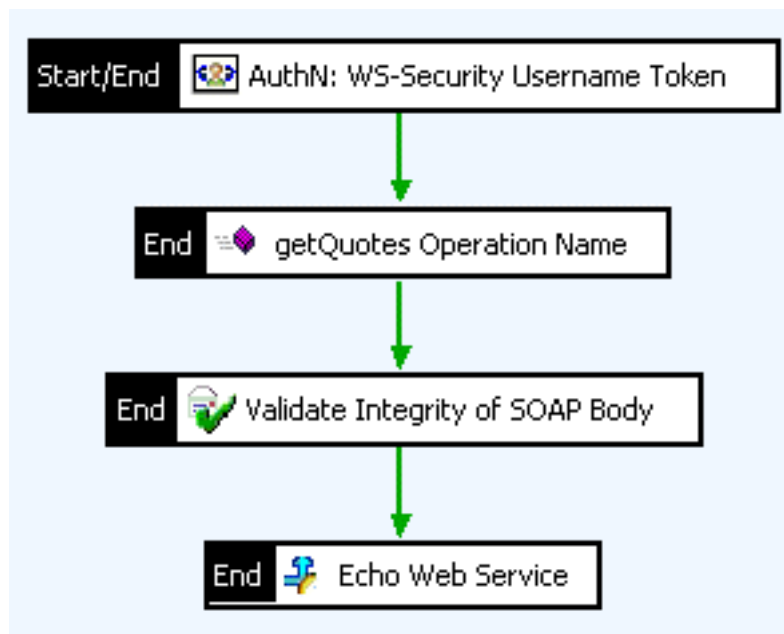
This tutorial walks you through a step-by-step example of how to build a custom-built message filter with the SDK, and integrate it into a policy. This tutorial shows how to build the two main aspects of an Addition filter: the server runtime component, and the Policy Studio configuration component. It then integrates these components into a policy, and shows how the filter adds the values of two parameters of a SOAP message together, and returns the result to the client. By the end of this tutorial, you should be able to write and test your own message filters by following a similar procedure.

Before going any further, however, it is important to explain exactly what are Oracle Enterprise Gateway filters and to see exactly how they can be wired together to create message processing policies.

Policies, Filters, and Message Attributes

A policy consists of a network of message filters where each filter is a modular executable unit that performs a specific type of processing on a message. The policy arranges these filters into sequences called paths. The filters then act as decision-making points along these paths, determining which filters are run on the message, and in what order.

The following four-node *policy* contains a single path with four *message filters*. The filter marked as **Start (AuthN: WS-Security Username Token)** is executed first. If this filter runs successfully, the next filter in the path (**getQuotes Operation Name**) is run, and so on until the last filter (**Echo Web Service**) in the path is executed.



When an HTTP request is received, it is converted into a set of message attributes. Each message attribute represents a specific characteristic of the HTTP request, such as the HTTP headers, HTTP body, and MIME parts, amongst others.

Every Filter declares the message attributes that it *requires*, *generates*, and *consumes* from the *attributes blackboard*. The blackboard contains all the available message attributes. When a filter *generates* message attributes it puts them up on the blackboard so that when another Filter *requires* them it can pull them off the blackboard. If a filter *consumes* a message attribute, it is wiped from the blackboard so that no other filter in the policy can use it.

The following table summarizes the attributes required and generated by the **Operation Name** resolver, which filters incoming requests based on their SOAP operation and namespace:

<i>Filter Name:</i>	Operation Name
<i>Description:</i>	Filters incoming SOAP requests based on their SOAP operation and namespace.
<i>Required Attributes:</i>	<code>content.body</code> <code>http.request.uri</code>
<i>Generated Attributes:</i>	<code>soap.request.method</code>

For more information on policy, policy building, filters, please refer to your Enterprise Gateway documentation. The following list summarizes the important concepts introduced in this section:

- **Policy:**
A network of interconnected message filters.
- **Message Filters:**
An executable unit that performs a specific type of processing on message attributes.
- **Message Attributes:**
Message attributes represent specific characteristics of a message.

Oracle Enterprise Gateway SDK Overview

The Oracle Enterprise Gateway Software Development Kit (SDK) comprises three Java packages that provide programmatic access to Oracle Enterprise Gateway policies, message objects, and the Entity Store. The following table describes these packages:

<i>Package</i>	<i>Description</i>
com.vordel.circuit	This package is responsible for implementing the core Oracle Enterprise Gateway policy. It includes the base classes for all message filters and their associated classes.
com.vordel.mime	Includes classes that encapsulate the message as it passes through the Oracle Enterprise Gateway policy. It also provides programmatic access to HTTP headers, HTTP body, request query string (if present), and MIME parts.
com.vordel.es	These classes provide access to the underlying Entity Store where all configuration data is stored.

Tutorial Prerequisites

You should read [Getting Started](#) to make sure you understand the concepts of policies and filters before continuing.

The Oracle Enterprise Gateway SDK requires a JDK 1.6, and is supported for the Windows, Linux, and Solaris packages.

Oracle Enterprise Gateway SDK Sample Overview

The Oracle Enterprise Gateway SDK ships with a working example of a message filter, called the `SimpleFilter`, that demonstrates how to use the SDK to build a filter. The filter extracts two integer parameters from a SOAP message, adds the integers, and returns the result of the addition in a SOAP response to the client.

This tutorial documents the steps required to build, integrate, configure, and test the supplied `SimpleFilter` and `SimpleProcessor` classes. The steps are as follows:

Step	Description
Step 1: Create TypeDocs	Every filter has an associated XML-based TypeDoc that contains the entity's type definition. It defines the configuration field names for that filter and their corresponding data types.
Step 2: Create Filter Class	Every message filter has an associated Filter class that encapsulates the configuration data for a particular instance of the filter. It also returns the corresponding Processor and Policy Studio classes.
Step 3: Create Processor Class	The Processor class is the server runtime component that is responsible for processing the message. Every message filter has an associated Processor and Filter class.
Step 4: Create Policy Studio Classes	All Filters are configured using the Policy Studio. Every Filter has a configuration wizard that enables you to set each of the fields defined in the entity that corresponds to that Filter. You can then add the filter to a policy to process messages.
Step 5: Build Classes	When the classes have been written, they must be built and added to the server and client classpaths.
Step 6: Load TypeDocs	The TypeDoc that is created for the filter in Step 1 must be registered with the Entity Store.
Step 7: Construct a Policy	This step constructs a policy that echoes messages back to the client, and then adds the newly created filter to it.
Step 8: Configure the SimpleFilter	This step uses the GUI component of the newly added filter to configure its configuration fields. It then tests the functionality of the filter (and its configuration) using the Oracle Service Explorer testing tool.

Step 1: Create the Typedocs

All configuration data is stored as entities in the Oracle Enterprise Gateway Entity Store. The Entity Store is an XML-based store that holds all configuration data required to run the Oracle core processing engine. Each configurable item has an entity type definition. The entity type definition is defined in an XML file known as the TypeDoc.

Entity types are analogous to class definitions in an object-oriented programming language. In the same way that in-

stances of a class can be created in the form of objects, an instance of an entity type can also be created. Therefore it is useful to think of the entity type defined in a TypeDoc as a header file, and the entity itself as a class instance. All entities and their entity type definitions are stored in the Entity Store.

Every filter requires specific configuration data to perform its processing on the message. For example, the `SimpleFilter`, which extracts the values of two elements from a SOAP message, and adds them together, must be primed with the names and namespaces of those two elements.

Because a filter is a configurable item, it requires a new XML typedoc to be written containing an entity type definition for it. The entity type for a filter contains a set of configuration parameters and their associated data types and default values.

When an instance of the filter is added to a policy using the Policy Studio, a corresponding entity instance is created and stored in the Entity Store. Whenever the filter instance is invoked, its configuration data is read from the entity instance in the Entity Store.

TypeDoc Syntax

The following example XML shows how the TypeDoc lists the various fields that form the configuration data for the Filter.

```
<entityStoreData>
  <entityType name="SimpleFilter" extends="Filter">
    <!-- Name of filter class that encapsulates the config data -->
    <constant name="class" type="string"
      value="com.vordel.example.filter.SimpleFilter"/>
    <!-- List of config fields, their types, and their default values -->
    <field ... />
    <field ... />
    <field ... />
  </entityType>
</entityStoreData>
```

All TypeDocs must obey the following simple rules:

- Extend the Filter type
- Define a constant Filter class
- List the configuration fields for the entity

SimpleFilter Elements and Attributes

The following table describes the important elements and attributes from the `SimpleFilter` TypeDoc listed above:

<i>Element</i>	<i>Attribute</i>	<i>Description</i>
<code><entityStoreData></code>		The topmost wrapper element for the entire type definition.
<code><entityType></code>		Contains the type definition, including all its fields and their types.
<code><entityType></code>	<code>name</code>	The unique name for this type.
<code><entityType></code>	<code>extends</code>	Entity definitions are hierarchical and can inherit from other higher level

<i>Element</i>	<i>Attribute</i>	<i>Description</i>
		types. All filters must extend the <code>Filter</code> type.
<code><constant></code>		A <code><constant></code> element is used to represent a read-only immutable property of the type.
<code><constant></code>	<code>name</code>	This attribute contains the name of the read-only property. In the example above, the named property is <code>class</code> , indicating that the value of this constant is the Java class that encapsulates the defined type. The name of this class must be specified as a <code><constant></code> .
<code><constant></code>	<code>type</code>	Specifies the type of the <code>value</code> attribute. In this case, the value is the name of a Java class, which is just a string.
<code><constant></code>	<code>value</code>	Contains the value of the named property, which is the name of the Java class that encapsulates this type <code>com.vordel.example.filter.SimpleFilter</code> .
<code><field></code>		Contains the definition of a single configuration field for this filter.
<code><field></code>	<code>name</code>	The name of the configuration field. You can see later in this tutorial how this name is used to get and set this property.
<code><field></code>	<code>type</code>	Specifies the data type of the named configuration field. For example, supported types include <code>string</code> , <code>boolean</code> , <code>encrypted</code> , and <code>integer</code> .
<code><field></code>	<code>cardinality</code>	Stipulates how many times this field can appear in an instance of the entity. For example, a cardinality of 1 means that this field can only occur once within an entity.
<code><field></code>	<code>default</code>	Specifies a default value for the configuration field, if appropriate.

SimpleFilter TypeDoc

The TypeDoc for the `SimpleFilter` is as follows:

```
<entityStoreData>
  <entityType name="SimpleFilter" extends="Filter">

    <!-- Name of Filter class that encapsulates this config entity -->
```

```

    <constant name="class" type="string"
              value="com.vordel.example.filter.SimpleFilter"/>

    <!-- List of config params, their types, and their default values -->
    <field name="param1" type="string" cardinality="1" default="a"/>
    <field name="param1Namespace" type="string"
           cardinality="1" default="http://startvbdotnet.com/web/" />
    <field name="param2" type="string" cardinality="1" default="b"/>
    <field name="param2Namespace" type="string"
           cardinality="1" default="http://startvbdotnet.com/web/" />
  </entityType>
</entityStoreData>

```

All type and related information for the Filter is contained in the top-level `<entityStoreData>` element. The Filter type declaration together with its field definitions and types are child elements of the `<entityType>` element. Each field name is specified in the `name` attribute of the `<field>` element, while the type and default value for the field are specified in the `type` and `default` attributes, respectively.

You can also provide internationalized log messages by specifying an `<entity>` block of type `InternationalizationFilter` within the `<entityStoreData>` elements.

Now that you understand how the configuration data for the filter is defined, you can create the filter class.

Step 2: Create the Filter Class

A filter class encapsulates the type information defined in an entity's type definition. There are class members that correspond to each of the fields in the type definition. At runtime, when the filter is invoked, the filter class is instantiated with the configuration data for the appropriate entity instance. The filter class is responsible for the following tasks:

- Storing member variables corresponding to fields in the type definition
- Specifying the message attributes it requires, consumes, and generates
- Returning the corresponding server runtime class (the Processor)
- Returning the corresponding Policy Studio class

SimpleFilter Class

The following code shows the members and methods of the `SimpleFilter.java` example:

```

package com.vordel.example.filter;

import com.vordel.circuit.DefaultFilter;
import com.vordel.circuit.FilterConfigureContext;
import com.vordel.circuit.MessageProperties;
import com.vordel.es.EntityStoreException;

/**
 * SimpleFilter contains the local name of the two
 * parameters (a and b) and contains the namespace that
 * these elements belong to (http://startvbdotnet.com/web/)
 */

```

```

public class SimpleFilter extends DefaultFilter {

    // element name of the first parameter
    String param1;
    // namespace of the first element
    String param1Namespace;
    // element name of the second parameter
    String param2;
    // namespace of the second parameter
    String param2Namespace;

    /**
     * Set the message attributes used by this filter
     */
    protected final void setDefaultProperties() {
        requiredProperties.add(MessageProperties.CONTENT_BODY);
    }

    /**
     * This method is called to set the config fields for the filter
     * @param ctx The configuration context for this filter
     * @param entity The entity object
     */
    public void configure(FilterConfigureContext ctx,
                        com.vordel.es.Entity entity)
        throws EntityStoreException {

        super.configure(ctx, entity);

        // read the settings for the processor
        param1 = entity.getStringValue("param1");
        param1Namespace = entity.getStringValue("param1Namespace");
        param2 = entity.getStringValue("param2");
        param2Namespace = entity.getStringValue("param2Namespace");
    }

    /**
     * Returns the server runtime Processor class associated
     * with this Filter class.
     */
    public Class getMessageProcessorClass() {
        return SimpleProcessor.class;
    }

    /**
     * Returns the GUI component for this Filter
     */
    public Class getConfigPanelClass() throws ClassNotFoundException {
        // Avoid any compile or runtime dependencies on SWT and other UI
        // libraries by lazily loading the class when required.
        return
            Class.forName("com.vordel.example.filter.simple.SimpleFilterUI");
    }
}

```

SimpleFilter TypeDoc

At this point, it is worth revisiting the entity definition for the `SimpleFilter` entity to see how the class members correlate to the defined fields.

```
<entityType name="SimpleFilter" extends="Filter">

  <!-- Name of Filter class that encapsulates this config entity -->
  <constant name="class" type="string"
    value="com.vordel.example.filter.SimpleFilter"/>

  <!-- List of config params, their types, and their default values -->
  <field name="param1" type="string" cardinality="1" default="a"/>
  <field name="param1Namespace" type="string"
    cardinality="1" default="http://startvbdotnet.com/web/" />
  <field name="param2" type="string" cardinality="1" default="b"/>
  <field name="param2Namespace" type="string"
    cardinality="1" default="http://startvbdotnet.com/web/" />
</entityType>
```

SimpleFilter Methods

The Filter class members (`param1`, `param1Namespace`, `param2`, and `param2Namespace`) directly correspond to the field definitions in the type definition. These members are populated in the `configure` method of the Filter class, which is called by the framework when the server is started up initially, and whenever the server is refreshed. The `Entity` class provides getter and setter methods for the different data types (for example, string, boolean, integer, and so on). For more details, see the [Entity Javadoc](#).

There are two more important methods implemented in this class: `setDefaultProperties` and `getMessageProcessorClass`. The `setDefaultProperties` method enables the Filter to define the message attributes that it *requires*, *generates*, and *consumes* from the attributes blackboard. The blackboard contains all the available message attributes. When a filter *generates* message attributes, it puts them up on the blackboard so that when another Filter *requires* them, it can pull them off the blackboard. If a filter *consumes* a message attribute, it is wiped from the blackboard so that no other filter in the policy can use it.

The attributes are stored in String arrays (`reqProps`, `genProps`, `conProps`), which are inherited from the `VariablePropertiesFilter` class. In the case of the `SimpleFilter` class, the `content.body` attribute is required because the SOAP parameters must be extracted from the body of the HTTP request.

The next important method here is the `getMessageProcessorClass` method, which returns the server runtime component (the Processor class) that is associated with this Filter class. Each Filter class has a corresponding Processor class, which is responsible for using the configuration data stored in the Filter class to process the message. The next step looks at the `SimpleProcessor` class to see how it acts on the data stored in the `SimpleFilter` class.

Finally, the corresponding Policy Studio configuration class is returned by the `getConfigPanelClass` method, which in this case is the `com.vordel.example.filter.simple.SimpleFilterUI` class. This class is described in detail in [Step 4](#) of this tutorial.

Step 3: Create Processor Class

The Processor class is responsible for performing the processing on the message. It uses the configuration data stored in the Filter class to determine how to process the message. It is important to note that this is the server runtime component of the filter that is returned by the `getMessageProcessorClass` of the Filter class described in the previous section.

Example Skeleton Code

The following skeleton code shows how the Processor *attaches* to the Filter class and uses its data to process the message. The following code is for illustration purposes only, and some of the `SimpleProcessor` code has been omitted.

```

package com.vordel.example.filter;

import java.io.ByteArrayInputStream;
import java.io.IOException;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

import com.vordel.circuit.Circuit;
import com.vordel.circuit.CircuitAbortException;
import com.vordel.circuit.Filter;
import com.vordel.circuit.Message;
import com.vordel.circuit.MessageProcessor;
import com.vordel.circuit.MessageProperties;
import com.vordel.es.EntityStore;
import com.vordel.mime.Body;
import com.vordel.mime.ContentType;
import com.vordel.mime.HeaderSet;
import com.vordel.mime.XMLBody;
import com.vordel.trace.Trace;

/**
 * This Processor acts as a simple Addition Web Service.
 * It extracts two parameters from a SOAP message and adds them together.
 * The result is then returned to the client.
 * The incoming message is expected in the following format:
 */
<?xml version="1.0" encoding="utf-8">
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Add xmlns="http://startvbdotnet.com/web/">
      <a>1</a>
      <b>1</b>
    </Add>
  </soap:Body>
</soap:Envelope>

The SimpleFilter contains the local name of the two parameters (a
and b), and contains the namespace that these elements belong to
(http://startvbdotnet.com/web/).
*/
public class SimpleProcessor extends MessageProcessor {

    /**
     * This method attaches the Filter to the Processor object.
     * This is called at startup and on every refresh.
     * This should contain server-side config/initialization.
     * For example, if this filter is required to establish
     * connections to any 3rd party products/servers, the
     * connection setup should be done here.
     * @param ctx Configuration context for the filter.
     * @param entity The Entity object
     */
    public void filterAttached(FilterConfigureContext ctx,
                               com.vordel.es.Entity entity)
        throws EntityStoreException {
        // nothing to do here for initialisation
        super.filterAttached(ctx, entity);
    }

    /**

```

```

* The invoke method performs the filter processing.
* @param c The circuit
* @param message The message
* @return true or false.
*/
public boolean invoke(Circuit c, Message message)
    throws CircuitAbortException {

    try {
        // Get the incoming request message as a DOM
        Document doc = getDOM(message);

        // Default result
        String result = "UNKNOWN";

        // Cast the filter member variable to a SimpleFilter so that
        // you may access the values stored in the SimpleFilter's
        // fields (for example, param1, param1Namespace, and so on).
        SimpleFilter f = (SimpleFilter)filter;

        // Look into the DOM to get the two parameters.
        // Get the 1st parameter
        NodeList param1 =
            doc.getElementsByTagNameNS(f.param1Namespace, f.param1);
        if (param1 == null || param1.getLength() <= 0)
            throw new CircuitAbortException(
                "Could not find " + f.param1 + "in message");
        // Get the value passed in the 1st parameter
        String a = getElementContent((Element)param1.item(0));

        // Get the 2nd parameter
        NodeList param2 =
            doc.getElementsByTagNameNS(f.param2Namespace, f.param2);
        if (param2 == null || param2.getLength() <= 0)
            throw new CircuitAbortException(
                "Could not find " + f.param2 + "in message");

        // Get the value of the 2nd parameter
        String b = getElementContent((Element)param2.item(0));

        // Calculate the result by adding the two parameter values
        result =
            Integer.toString(Integer.parseInt(a) + Integer.parseInt(b));

        // Set the response by setting the content body
        // to be the response
        HeaderSet responseHeaders = new HeaderSet();
        responseHeaders.putString("Content-Type", "text/xml");
        StringBuffer response = new StringBuffer(RESPONSE_START);
        response.append(result);
        response.append(RESPONSE_END);
        Body convertedBody =
            Body.create(responseHeaders, new ContentType("text/xml"),
                new ByteArrayInputStream(
                    response.toString().getBytes()));
        message.setProperty(
            MessageProperties.CONTENT_BODY, convertedBody);

        return true;
    }
    catch (IOException exp) {
        Trace.error("IOException in SimpleProcessor: " + exp.getMessage());
        return false;
    }
}

```

```
}
```

Processor Methods

There are two important methods that *must* be implemented by every Processor: the `filterAttached` method and the `invoke` method. The `filterAttached` method associates an appropriate Filter class with the Processor. The Processor can then access the configuration data stored in the Filter class. In this case, the `SimpleProcessor` attaches to the `SimpleFilter` class. The `filterAttached` method should contain any server-side initialization or configuration that is to be performed by the Filter, such as connecting to third-party products or servers.

The `invoke` method is responsible for using the data stored in the attached Filter class to perform the message processing. This method is called by the server as it executes the series of filters in any given policy. In the case of the `SimpleFilter`, the `invoke` method extracts the values of the `<a>` and `` elements from the SOAP message, and adds the two values together. The result is then returned to the client in a templated SOAP response.

It is important to note that the `invoke` method can have the following possible results:

Result	Description
True	If the filter processed the message successfully (for example, successful authentication, schema validation passed, and so on), the <code>invoke</code> method should return a true result, meaning that the next filter on the success path for the filter is invoked.
False	If the filter's processing fails (for example, the user was not authenticated, message failed integrity check, and so on), the <code>invoke</code> method should return false, meaning that the next filter on the failure path for the filter is invoked.
<code>CircuitAbortException</code>	If for some reason the filter cannot process the message at all (for example, if it can not connect to an Identity Management server to authenticate a user), it should throw a <code>CircuitAbortException</code> . If a <code>CircuitAbortException</code> is thrown in a policy, the designated Fault Processor (if any) is invoked instead of any successive filters on either the success or failure paths.

Now that you have a class to encapsulate the configuration data and another class to act on that data, it is now time to create some GUI classes where the user can configure the fields stored in the Filter class.

Step 4: Create Policy Studio Classes

The next step involves writing two GUI classes so that the fields defined in the `SimpleFilter` type definition can be configured. When the GUI classes and resources have been built, the visual components can be used in the Policy Studio to configure the Filter and add it to a policy.

SimpleFilter GUI Classes and Resources

The following table describes the GUI classes and resources that relate to the `SimpleFilter`:

Class or Resource	Description
<code>SimpleFilterUI.java</code>	This class is used to list the pages that are involved in a Filter's configuration screen. Each Filter has at least two pages: the main configuration page, and a page where log messages related to the filter can be customized. This class is returned by the <code>getConfigPanelClass</code> method of the <code>SimpleFilter</code> class.
<code>SimpleFilterPage.java</code>	This class defines the layout of the visual fields on the Filter's main configuration screen. For example, there are four text fields on the configuration screen for the <code>SimpleFilter</code> corresponding to the four fields defined in the entity's type definition.
<code>resources.properties</code>	This file contains all text that is displayed in the GUI configuration screen (for example, dialog titles, field names, and error messages). This means that the text can be customized or internationalized easily without the need for code change.
<code>simple.gif</code>	This image file is used as the icon to identify the Filter in the Management Console, and is displayed in the Filter Palette.

This step first looks at the `SimpleFilterUI` class, which is returned by the `getConfigPanelClass` method of the `SimpleFilter` class. It is responsible for the following:

- Listing the configuration pages that make up the interface for the filter
- Naming the category of filters to which this filter belongs
- Specifying the name of the images to use as the icons/images for this filter

SimpleFilterUI Class

The code for the `SimpleFilterUI` is as follows:

```
package com.vordel.example.filter.simple;

import java.util.Vector;

import org.eclipse.jface.resource.ImageDescriptor;
import org.eclipse.swt.graphics.Image;

import com.vordel.client.manager.Images;
import com.vordel.client.manager.filter.DefaultGUIFilter;
import com.vordel.client.manager.wizard.VordelPage;

/**
 * Filter configuration GUI for 'Simple' example filter.
 * This class shows how to code simple text fields for configuring a
 * custom filter.
 */
public class SimpleFilterUI
    extends DefaultGUIFilter
{

```



```

/**
 * Add the pages you want to show in the configuration wizard for the filter.
 */
public Vector<VordelPage> getPropertyPages() {
    Vector<VordelPage> pages = new Vector<VordelPage>();

    // Add the panel for configuring the specific fields
    pages.add(new SimpleFilterPage());

    // Add the page which allows the user to set the log strings for the
    // audit trail, for the pass/fail/error cases
    pages.add(createLogPage());

    return pages;
}

/**
 * Set the categories in which you want to display this Filter. The
 * categories define the sections of the palette in which the Filter
 * appears. The values returned should be the localized name of the
 * palette section, so ensure that the property is defined in the
 * resources.properties in this class' package. You will add this
 * file to the "Example Filters" category.
 */
public String[] getCategories() {
    return new String[]{_("FILTER_GROUP_EXAMPLE")};
}

/*
 * Register our custom images with the image registry
 */
private static final String IMAGE_KEY = "simpleFilter";
static {
    Images.getImageRegistry().put(IMAGE_KEY,
        Images.createDescriptor(SimpleFilterUI.class, "simple.gif"));
}

/**
 * The icon image needs to be added in images.properties in com.vordel.client.manager
 * the id used there is used as a reference here.
 * Use this method to get image id for the small icon image in Images.get(id), etc.
 */
public String getSmallIconId() {
    return IMAGE_KEY;
}

/**
 * Implement this method if you want to display a non-default image
 * for your filter in the circuit editor canvas and navigation tree.
 */
public Image getSmallImage() {
    return Images.get(IMAGE_KEY);
}

/**
 * Implement this method to display a non-default icon for your filter in
 * the palette.
 */
public ImageDescriptor getSmallIcon() {
    return Images.getImageDescriptor(IMAGE_KEY);
}
}

```

SimpleFilterUI Methods

The following table describes the important methods:

Method	Description
<code>public Vector getPropertyPages()</code>	Initializes a Vector of the Pages that makeup the total configuration screens for this Filter. Successive Pages are accessible by clicking the Next button on the Policy Studio configuration screen.
<code>public String[] getCategories()</code>	This method returns the names of the Filter categories that this Filter belongs to. The Filter is displayed under these categories in the Filter Palette in the Policy Studio. The <code>SimpleFilter</code> is added to the Example Filters category.
<code>public Image getSmallImage()</code>	The default image for the Filter, which is registered in the static block in the code above, can be overridden by returning a different image here.
<code>public ImageDescriptor getSmallIcon()</code>	The default icon for the Filter can be overridden by returning a different icon here.

A *Page* only represents a single configuration screen in the Policy Studio. You can chain together several Pages to form a series of configuration screens that together make up the overall configuration for a given Filter. By default, all Filters consist of two pages: one for the configuration fields for the Filter, and the other to allow per-Filter logging. However, there is no reason why more Pages can not be chained together. Successive Pages should be added to the configuration in the `getPropertyPages` method.

From looking at the `getPropertyPages` method of the `SimpleFilterUI`, it is clear that the `SimpleFilterPage` class forms one of the configuration screens (or pages) for the `SimpleFilter` filter. The `SimpleFilterPage` class is responsible for the layout of all the input fields that make up the configuration screen for the `SimpleFilter`.

SimpleFilterPage Class

The code for the `SimpleFilterPage` class is shown below:

```
package com.vordel.example.filter.simple;

import org.eclipse.swt.SWT;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.widgets.Composite;

import com.vordel.client.manager.wizard.VordelPage;

public class SimpleFilterPage extends VordelPage
{
    /**
     * Create the configuration page. Set the title and description
     * here. The title and description are maintained in the
     * resources.properties file for customization and
     * internationalization purposes.
     */
    public SimpleFilterPage() {
        // Call the super constructor with a unique name for this
        // page to bind it with its corresponding wizard.
        super("simplePage");
    }
}
```

```

        setTitle(_("SIMPLE_PAGE"));
        setDescription(_("SIMPLE_PAGE_DESCRIPTION"));
        setPageComplete(false);
    }

    /**
     * Get the unique identifier for the help page for this filter.
     * The id-to-page mapping is maintained in the
     * '/docs/res/help.jhm' file. The URL for the page to describe
     * the filter should be relative to the '/docs' directory.
     */
    public String getHelpID() {
        return "simple.help";
    }

    /**
     * Any post-processing of the configuration values can happen
     * here, before they are persisted to the Entity Store. If the
     * configuration is not complete and valid, notify the user here
     * with a dialog box and return false here, otherwise return true.
     *
     * @see com.vordel.client.manager.util.MsgBox
     */
    public boolean performFinish() {
        // Simple mutually independent values here, no checking required,
        // so return true
        return true;
    }

    /**
     * Create the main control for the Filter configuration dialog.
     * This will house the text fields for setting the particular
     * Field Values in the Entity.
     */
    public void createControl(Composite parent) {
        // Create a Panel with two columns
        GridLayout layout = new GridLayout();
        layout.numColumns = 2;
        Composite container = new Composite(parent, SWT.NULL);
        container.setLayout(layout);

        // Add controls to populate the appropriate Entity Fields
        // You use the localization keys for the field names and
        // descriptions which will map to entries in the
        // resources.properties file.
        createLabel(container, "SF_NAME");
        createTextAttribute(container, "name", "SF_NAME_DESC");

        createLabel(container, "SF_PARAM1");
        createTextAttribute(container, "param1", "SF_PARAM1_DESC");

        createLabel(container, "SF_PARAM1NS");
        createTextAttribute(container, "param1Namespace", "SF_PARAM1NS_DESC");

        createLabel(container, "SF_PARAM2");
        createTextAttribute(container, "param2", "SF_PARAM2_DESC");

        createLabel(container, "SF_PARAM2NS");
        createTextAttribute(container, "param2Namespace", "SF_PARAM2NS_DESC");

        // Finish up the page definition
        setControl(container);
        setPageComplete(true);
    }

```

```

}

```

SimpleFilterPage Methods

There are four important interface methods that must be implemented in this class:

Method	Description
<code>public SimpleFilterPage()</code>	The constructor performs some basic initialization, such as setting a unique ID for the page and setting the title and description for the page. The text representing the page title and description are kept in the <code>resources.properties</code> file so that they can be localized or customized easily, if necessary.
<code>public String getHelpID()</code>	<p>This method is called by the Policy Studio help system. There is a Help button on every configuration page in the Policy Studio. When this button is pressed, the help system is invoked. Every page has a help ID (for example, <code>simple.help</code>) associated with it, which is mapped to a help page. This mapping is defined in the <code>/plugins/com.vordel.rcp.policystudio.resources_(version ...)/contexts.xml</code> file under the directory where you have installed Policy Studio, for example:</p> <pre> <context id="simple_help"> <description>>Simple filter tes <topic label="Simple Filter" hr </context> </pre> <p>Please note a dot in a help ID is replaced by underscore in the <code>contexts.xml</code> file as in the example above. All URLs specified in the <code>contexts.xml</code> file are relative from the <code>/plugins/com.vordel.rcp.policystudio.resources_(version ...)</code> folder of your Policy Studio installation. You can add the lines from the example above to the <code>contexts.xml</code> file now, and you will check that the help page works at the end of this tutorial.</p>
<code>public boolean performFinish()</code>	This method gives you the chance to process the user-specified data before it is submitted to the Entity Store. For example, any validation on the data should be added to this method.
<code>public void createControl(Composite parent)</code>	This method is responsible for creating and ordering the input fields on the configuration page. Again, localization keys from the <code>resource.properties</code> file are used to give labels for the input fields. It is important to note that the <code>createTextAttribute</code> takes a String as its second

<i>Method</i>	<i>Description</i>
	parameter, which corresponds to a field defined for an entity (for example, param, param1Namespace, param2, and param2Namespace are all defined in the SimpleFilter entity type. When the user submits the values entered in these fields, the values are set to the corresponding fields in the entity instance in the Entity Store.

resources.properties File

Both the SimpleFilterUI and the SimpleFilterPage classes use localized keys for all text that is displayed on the configuration screen. This makes it a trivial task to localize or customize all text that is displayed in the Policy Studio. The localization keys and their corresponding strings are stored in the resources.properties file, which takes the following format:

```
# Palette category for example filters
FILTER_GROUP_EXAMPLE=Example Filters

# Title and Description for the SimpleFilter
SIMPLE_PAGE=Simple Filter Configuration
SIMPLE_PAGE_DESCRIPTION=Configure parameter values for the Simple Filter

#
# Field labels and descriptions
#
SF_NAME=Filter Name:
SF_NAME_DESC=The name of the Simple Filter
SF_PARAM1=Parameter 1
SF_PARAM1_DESC=the first parameter
SF_PARAM1NS=Parameter 1 Namespace
SF_PARAM1NS_DESC=the first parameter namespace field
SF_PARAM2=Parameter 2
SF_PARAM2_DESC=the second parameter
SF_PARAM2NS=Parameter 2 Namespace
SF_PARAM2NS_DESC=the second parameter namespace field
```

The final resource is the simple.gif image file, which is displayed as the icon for the SimpleFilter in the Policy Studio. You can see this icon a little later in this tutorial when you configure the SimpleFilter.

Now that all classes and resources have been written, it is now time to build the relevant JAR files and incorporate them into the product.

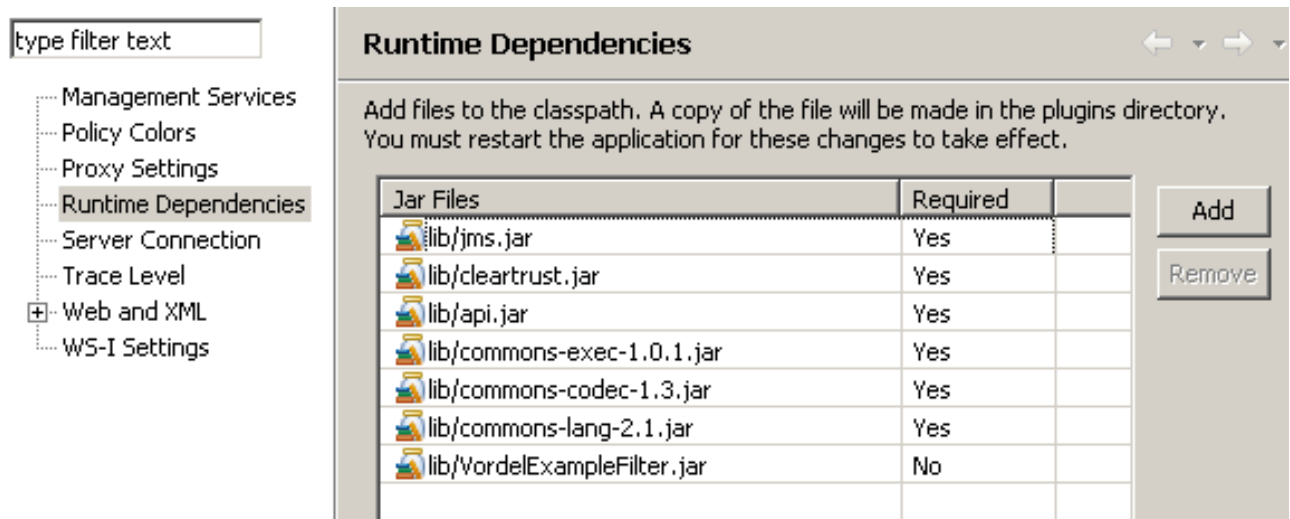
Step 5: Build Classes

You *must* perform the following steps to build the Java classes and resources described in the previous sections.

Important Note:

The classes must be built against a 1.6 JDK because this is used to build the Enterprise Gateway, which contains the JAR files for the Enterprise Gateway SDK API.

1. Build the classes and associated resources into a JAR file using your chosen build technology.
2. Place the new JAR in the `GW_HOME/ext/lib` directory, where `GW_HOME` refers to the root of your Enterprise Gateway installation.
3. In the Policy Studio main menu, select **Window -> Preferences -> Runtime Dependencies**, and click **Add** to browse to the new JAR, and add it to the list (for example, `GW_HOME/ext/lib/VordelExampleFilters.jar`).
4. Place any third-party JAR files used by your classes into the `GW_HOME/ext/lib`, and add them to the list of **Runtime Dependencies** in the Policy Studio.
5. Restart both the server and the Policy Studio.



For an example of building the `SimpleFilter` classes, see the `Ant build.xml` file supplied in the `SDK_HOME/example/filter/src` directory, where `SDK_HOME` points to the root of your SDK installation.

The Ant file builds the `SimpleFilter` classes and packages all associated resources into the `VordelExampleFilters.jar` file. You must then place this file into the `GW_HOME/ext/lib` folder, and add it to the **Runtime Dependencies** in the Policy Studio **Preferences**.

When both the server and the Policy Studio boot up, they automatically pick up the new JAR file. In the remaining steps of this tutorial, you can see how to configure a policy that includes the `SimpleFilter` and then test its functionality.

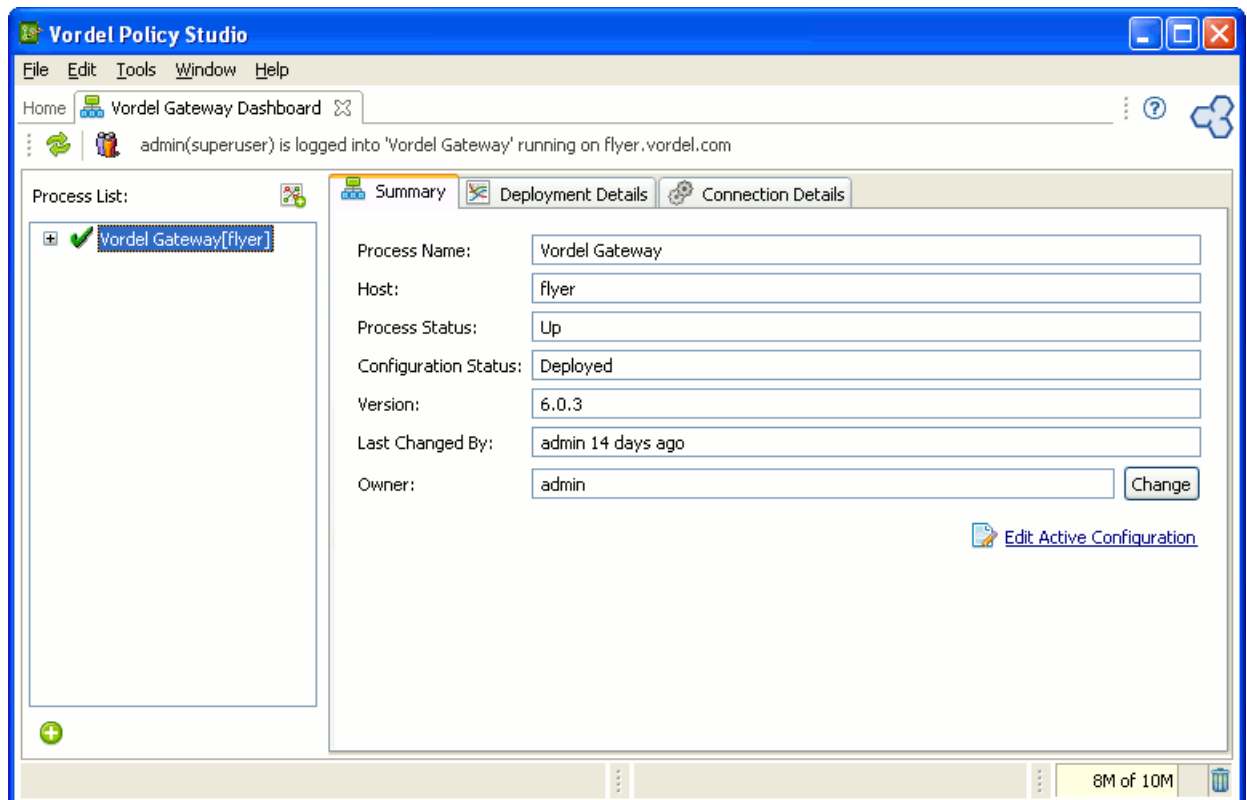
Step 6: Load TypeDocs

You must now register the type definition for the `SimpleFilter` with the Entity Store using the Policy Studio. When the entity type has been registered, it is guaranteed that any time the server needs to create an instance of the `SimpleFilter`, the instance contains the correct fields with the appropriate types.

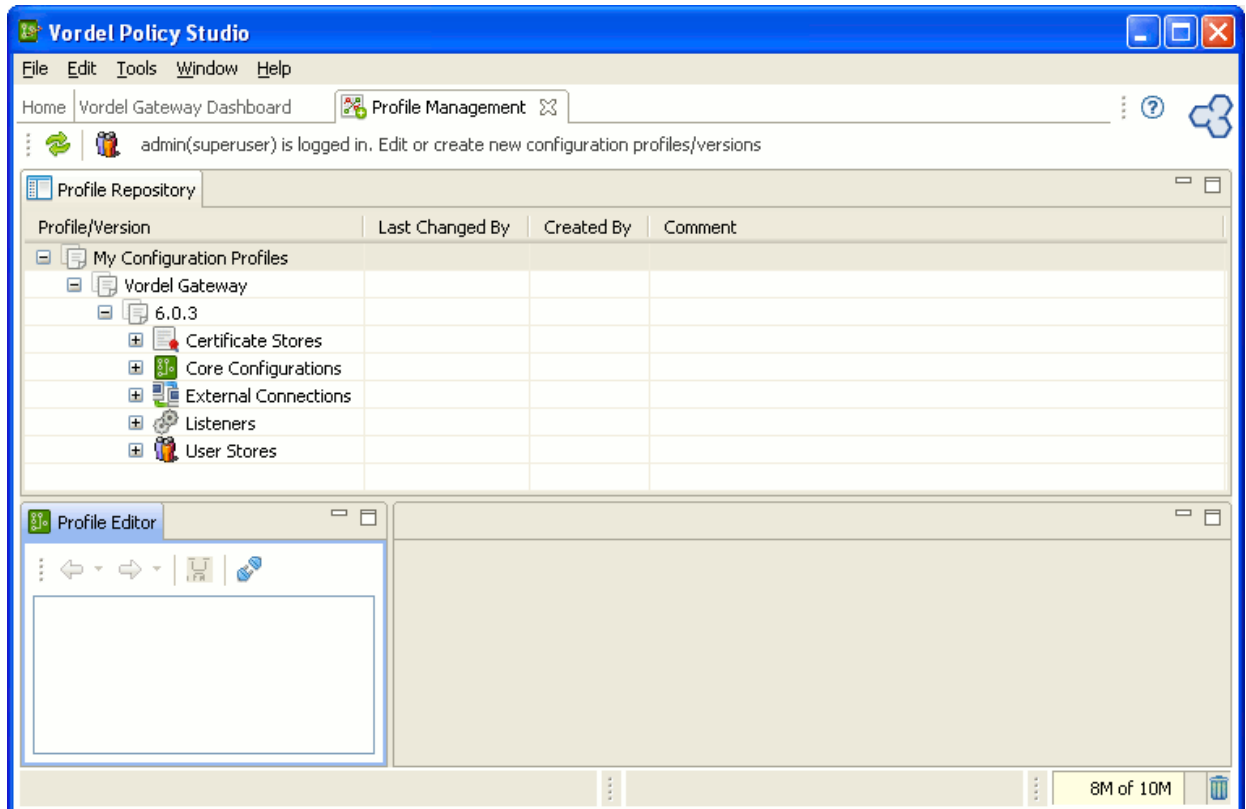
Register using the Policy Studio

To register the type definition using the Policy Studio, perform the following steps:

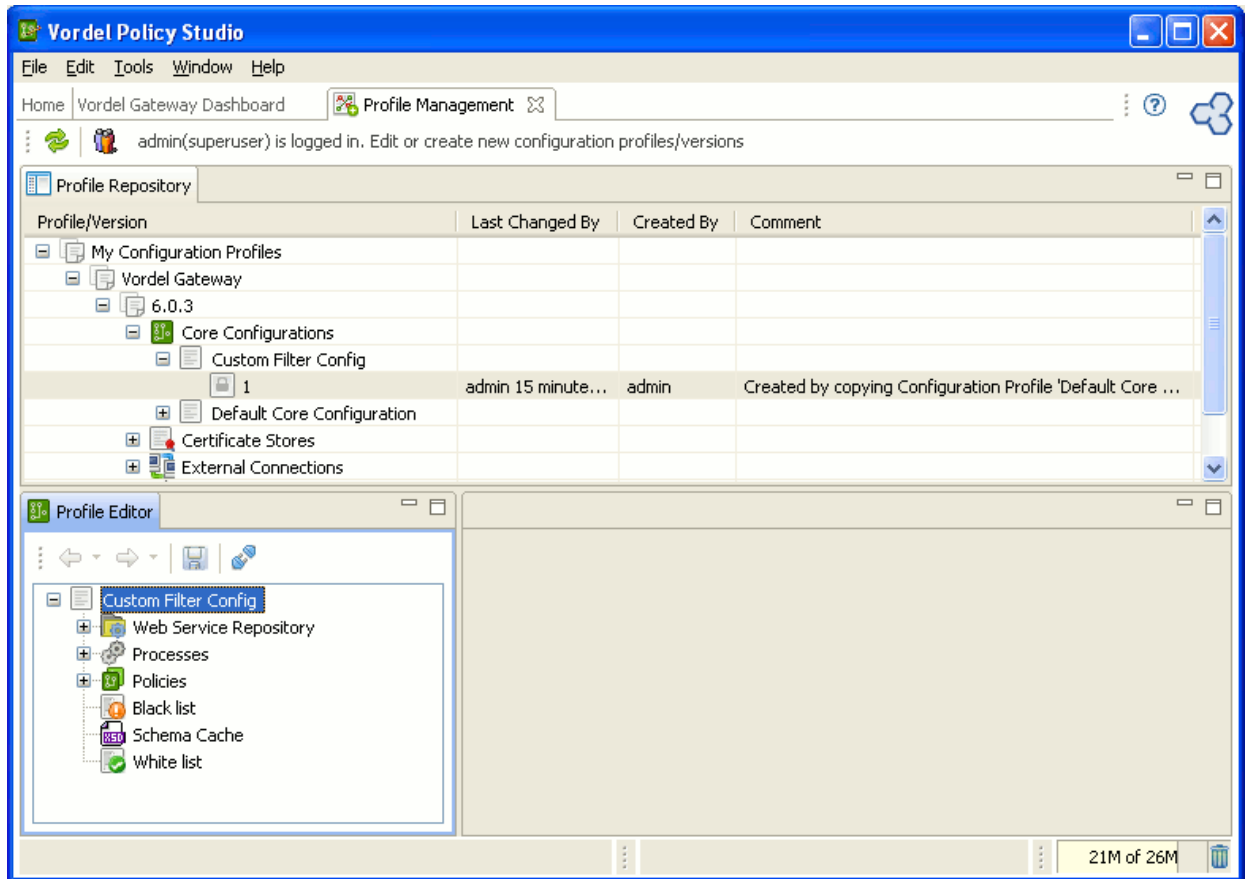
1. Start the Policy Studio and connect to the Enterprise Gateway. The following screen is displayed:



2. Select **Window -> Show View -> Profile Repository** from the main menu, and you should see a screen similar to the following:



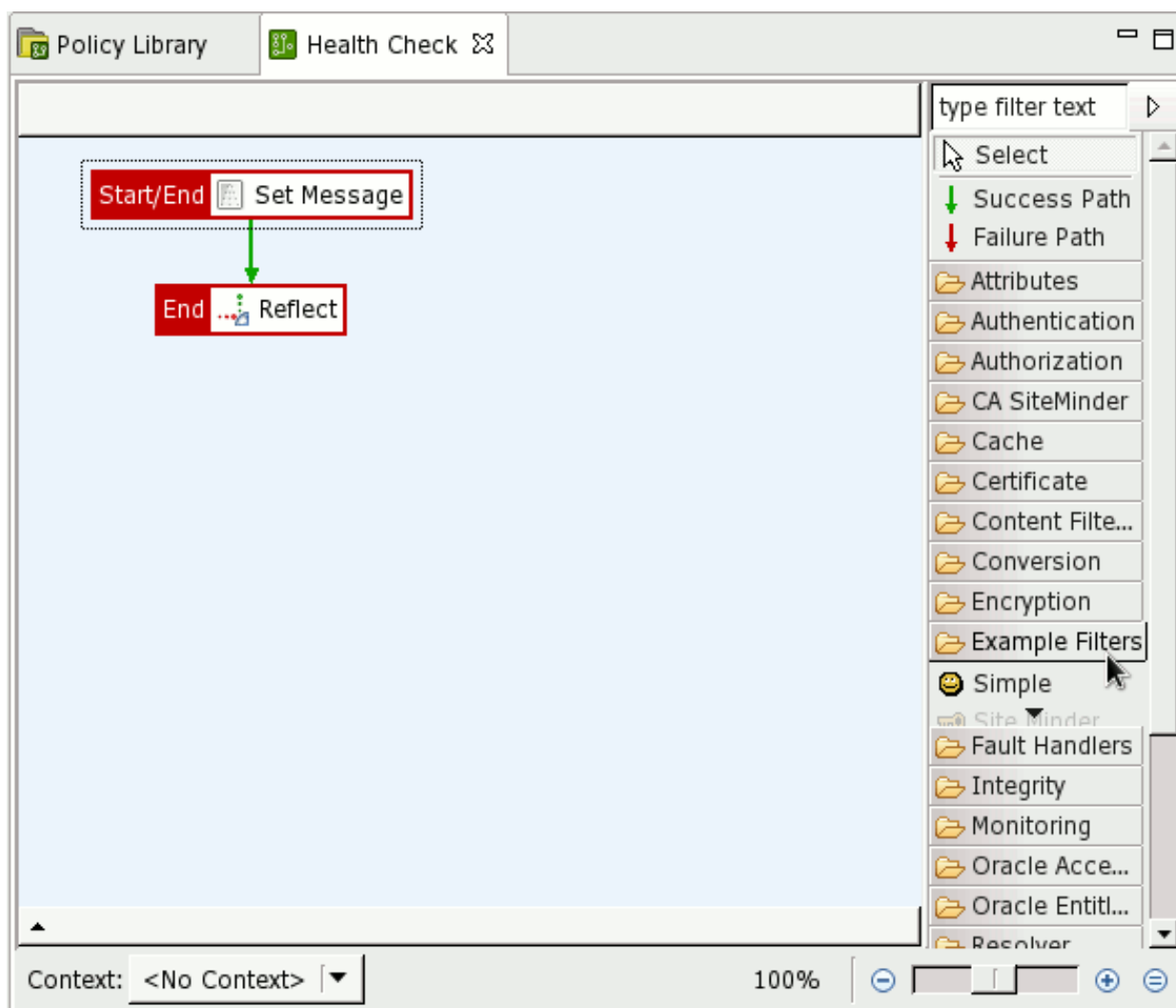
3. Create a copy of the active configuration. To do this, expand the **Core Configurations** tree node, and right-click the active configuration (for example, **Default Configuration**). Select **Create via Copy** in the context menu, and give the new configuration a meaningful name (for example, `Custom Filter Config`).
4. Right-click the new configuration, and select **Edit** from the context menu. You are asked to enter a passphrase. If this has not been changed from the default, you can leave the field blank and proceed. This displays the **Profile Editor** tab with **Custom Filter Config** as a root node. Right-click this root node, and select **Import Custom Filter Types** from the menu.



5. Browse to the location of the `sampleTypeSet.xml` file, which is located in the `/filter` directory of your Oracle Enterprise Gateway installation. A *TypeSet* file is used to group together one or more *TypeDocs* so that multiple *TypeDocs* can be added to the Entity Store in batch mode. The `sampleTypeSet.xml` file is displayed as follows:

```
<typeSet>
  <!-- SimpleFilter TypeDoc -->
  <typedoc file="SimpleFilter.xml"/>
</typeSet>
```

6. After selecting the *TypeSet*, the workspace refreshes. To verify that the filter is available, navigate to a pre-existing policy, and you should see the **Example Filters** category in the palette, with the filters contained inside.



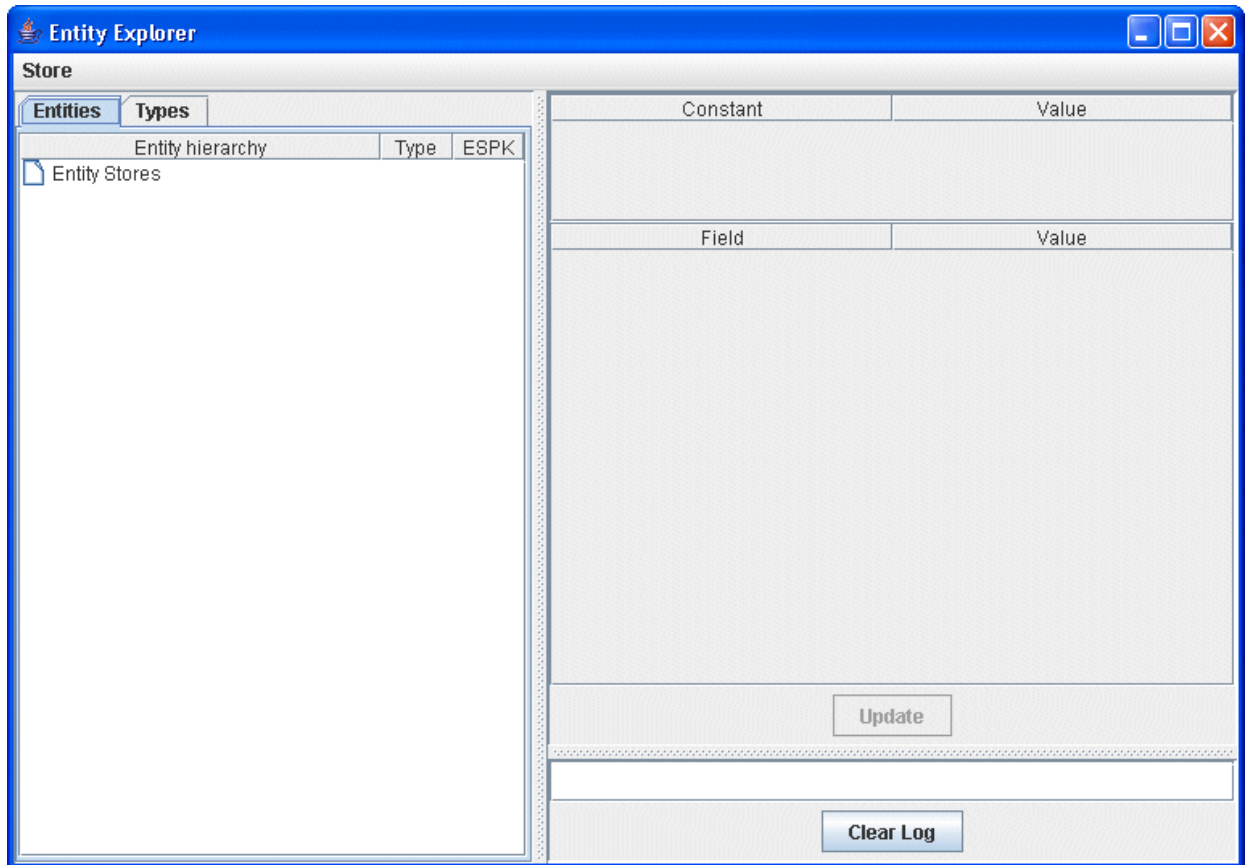
7. Right click the configuration, and commit this version. From now on, any further revisions of this version, or configurations copied from this configuration contain the new filter types.
8. Click the **Deployment Details** tab on the Dashboard, and select the running process and the core configuration. You can now deploy the new `Custom Filter Config` with the latest version containing the new types.
9. Click the **Deploy** button at the bottom right of the screen.

Verifying using the Entity Explorer

Another way to verify that your new filter has been installed is to use the **Entity Explorer**. You can use the **Entity Explorer** tool for browsing the entity types and entity instances that have been registered with the Entity Store.

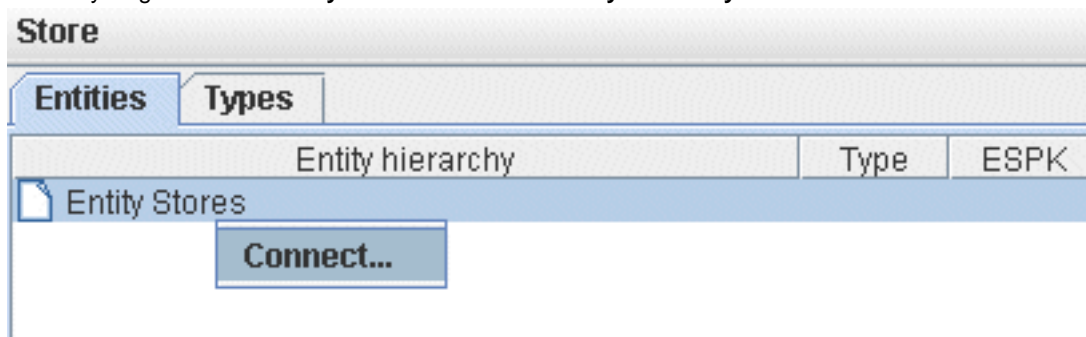
To verify that the filter has been installed, perform the following steps:

1. Start the Entity Explorer from the `/bin` directory of your installation using the `esexplorer` startup script. The Entity Explorer is displayed as follows:

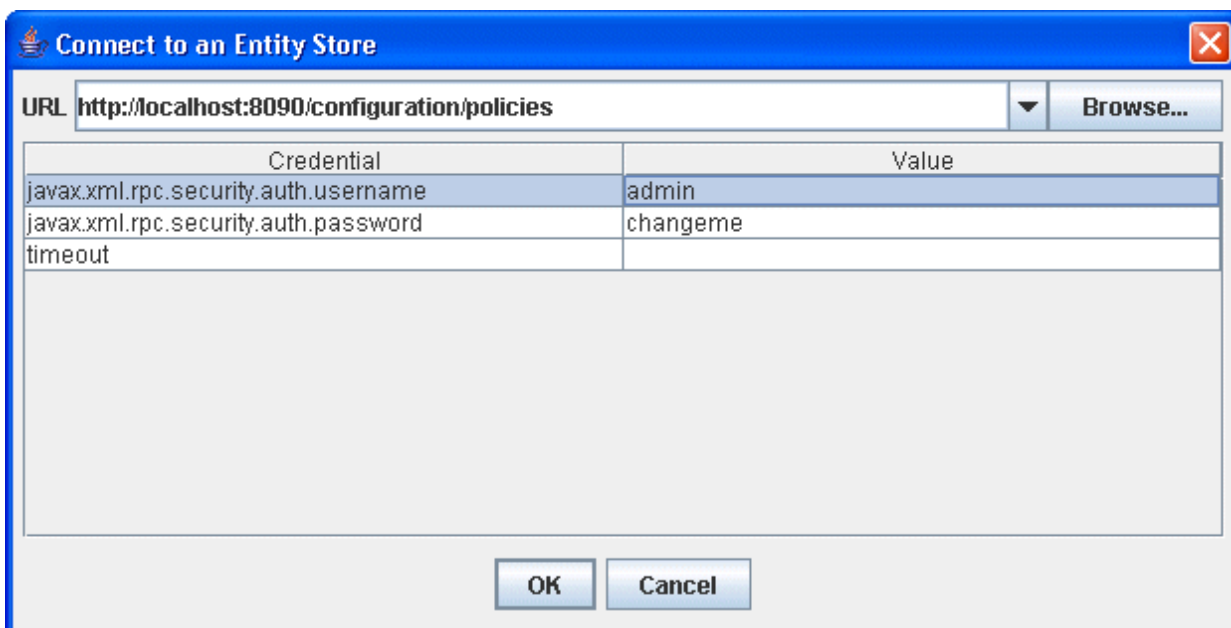


There are two main tabs on the Entity Store's interface: **Entities** and **Types**. The **Types** tab lists all currently defined entity types that have been registered with the Entity Store, while the **Entities** tab lists all the instances of entities that have been committed to the Entity Store (for example, configured filters).

2. You must first point the Entity Explorer at the management interface exposed by a running instance of the Enterprise Gateway. Right-click the **Entity Stores** node in the **Entity Hierarchy** tree:



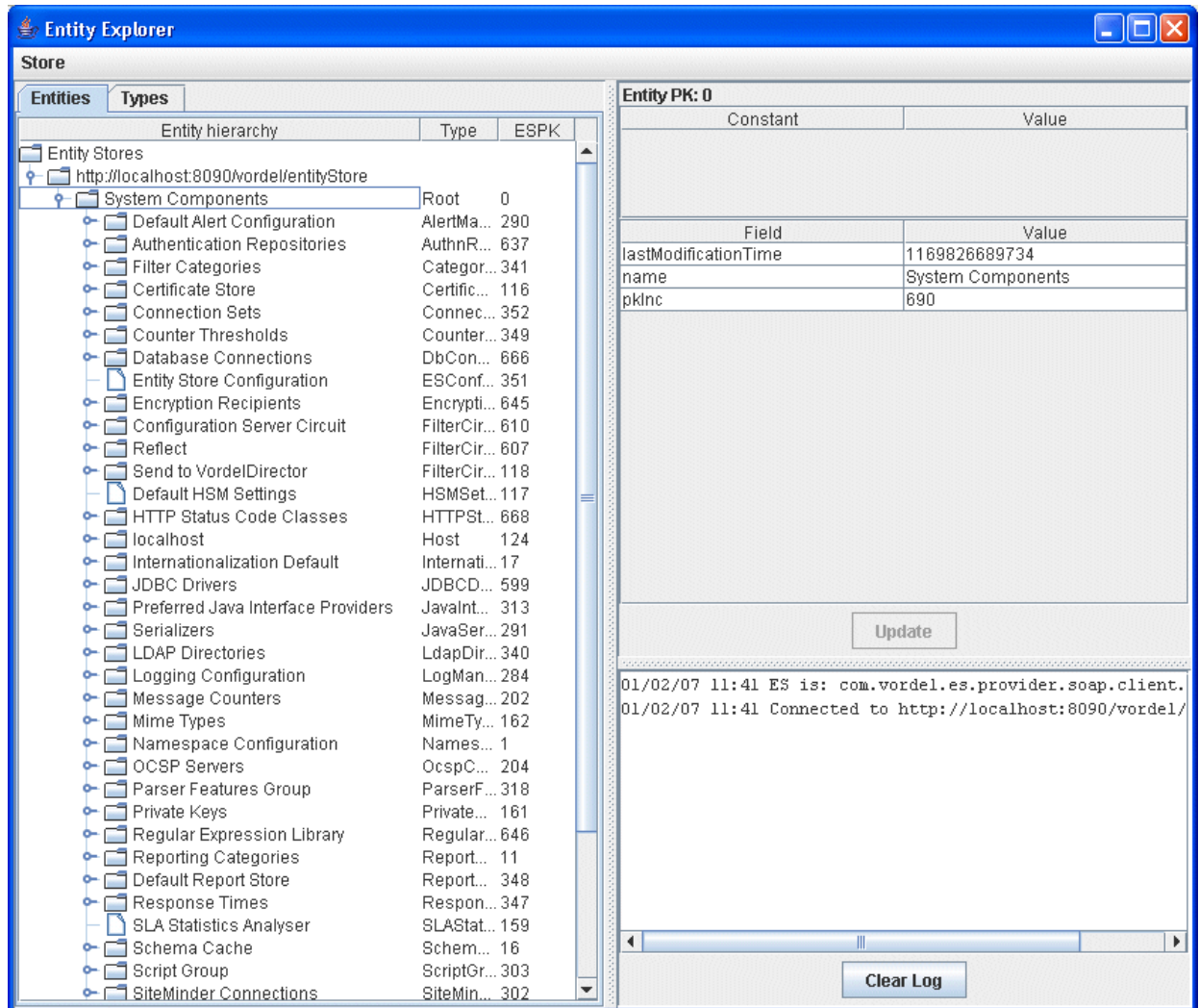
The **Connect to an Entity Store** dialog is displayed as follows:



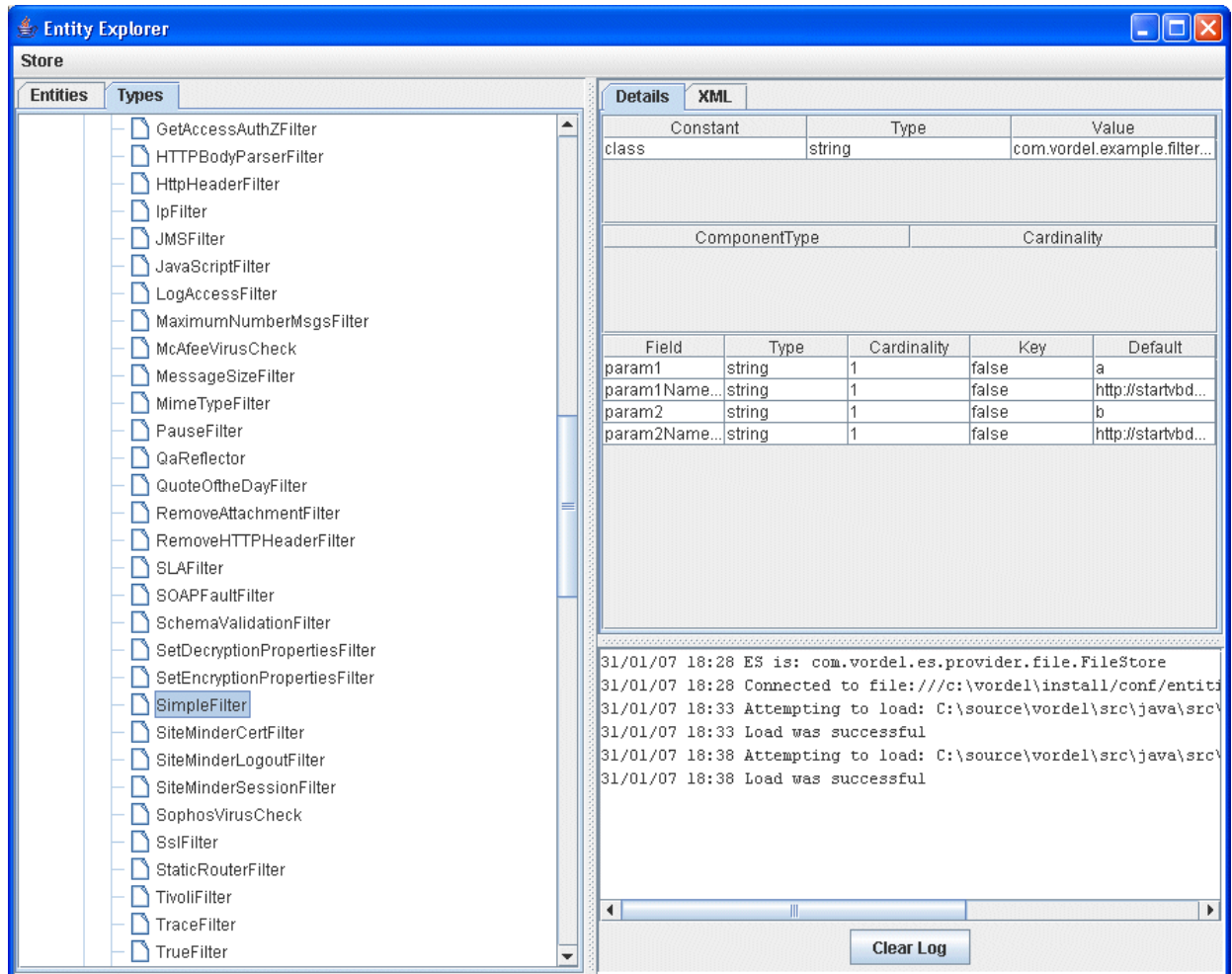
The dialog box titled "Connect to an Entity Store" has a blue title bar with a close button. It contains a URL field with the text "http://localhost:8090/configuration/policies" and a "Browse..." button. Below the URL field is a table with two columns: "Credential" and "Value". The table contains three rows: "javax.xml.rpc.security.auth.username" with value "admin", "javax.xml.rpc.security.auth.password" with value "changeme", and "timeout" with an empty value field. At the bottom of the dialog are "OK" and "Cancel" buttons.

Credential	Value
javax.xml.rpc.security.auth.username	admin
javax.xml.rpc.security.auth.password	changeme
timeout	

3. The Enterprise Gateway exposes a management service that interfaces to the underlying Entity Store. This is the preferred method of managing the Entity Store. You can now start the Enterprise Gateway (which connects to the Entity Store), and point the Entity Explorer at the management service exposed by the Enterprise Gateway. Start the Enterprise Gateway from the `/bin` directory of your product installation.
4. You can now configure the Entity Explorer to talk to the management service exposed by the Enterprise Gateway. By default, this service is available at the following URL, where `HOST` refers to the host name or IP address of the machine on which the Enterprise Gateway is running:
`http://HOST:8090/configuration/policies`.
5. Enter this address in the **URL** field of the **Connect to an Entity Store** dialog. If you have not already changed the default username and password for the entity store, use the default username `admin` with password `changeme`. Otherwise, specify the alternative username and password in the fields provided. Click **OK** to connect to the management service on the Enterprise Gateway. A log message is displayed in the Log panel at the bottom right corner of the screen to confirm that you are connected to the Enterprise Gateway at the specified URL.
6. Expand the Entity Store filename, and then expand the **System Components** node to display the list of entity instances stored in the Entity Store:



- Click the **Types** tab, expand the node representing the Enterprise Gateway's management interface, (for example, `http://my_host:8090/configuration/policies`), and expand the **Entity** node to display the list of registered entity types. Each of these entity types has a corresponding type definition. Expand the **Filter** node, and click the **SimpleFilter** entity type in the tree. The names and data types of the fields for this entity type can be seen under the **Details** tab on the right of the Entity Explorer. To view the type definition for this entity, click the **XML** tab. These details should match the filter that has just been added.



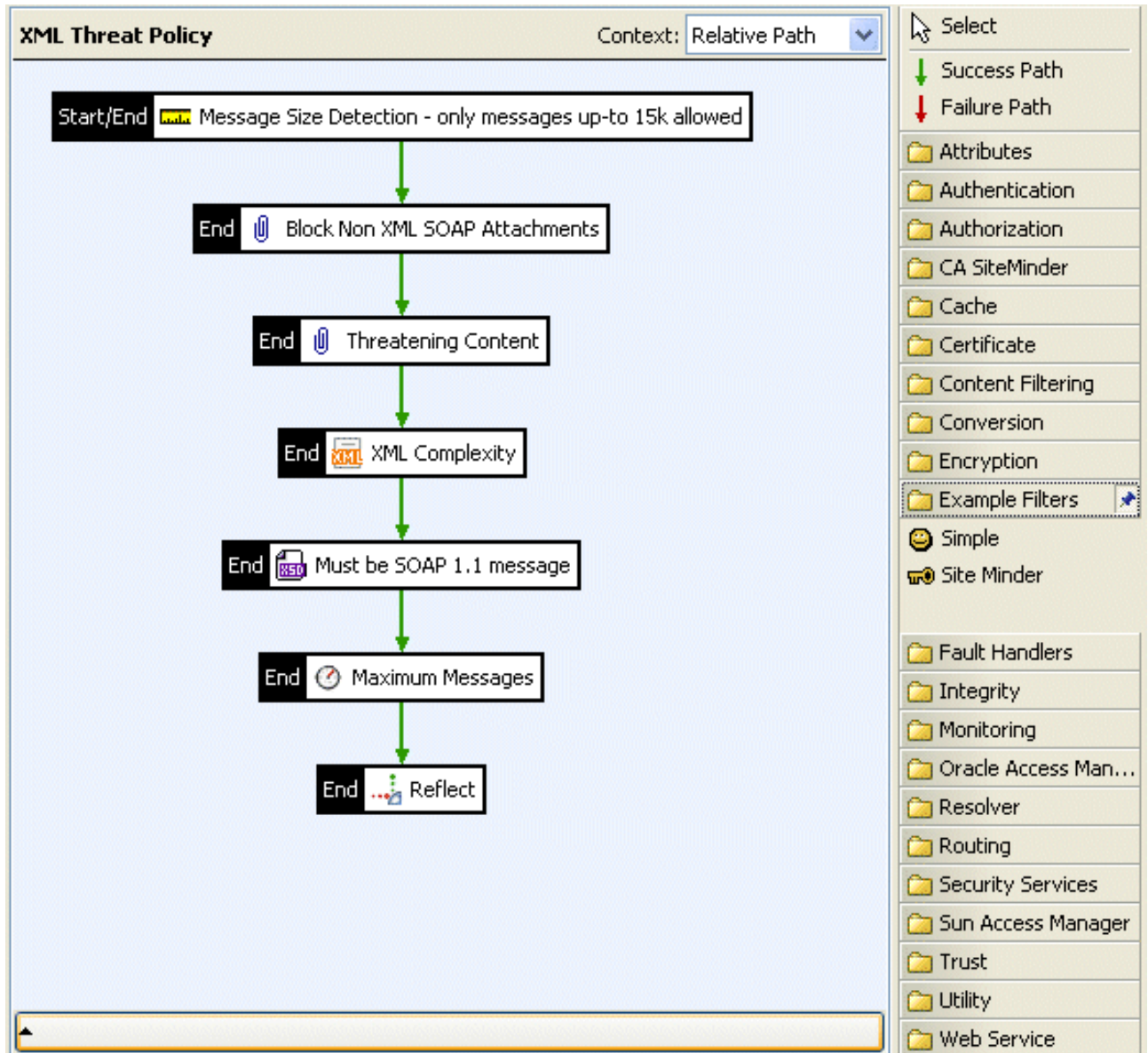
The Entity Store is now aware of the `SimpleFilter` type. You can now create an instance of a Filter class that encapsulates the fields defined in the `SimpleFilter` type.

The remaining steps in this tutorial show how to configure a policy that includes the `SimpleFilter`, and then tests its functionality.

Step 7: Construct a Policy

This section first shows how to build a simple policy that echoes messages back to the client. The next step then adds the **SimpleFilter** to the policy.

You can build policies using the Policy Editor in the **Policy Studio**. To build a policy, you can drag message filters from the filters palette on the left on to the policy canvas on the right. You can then link these filters using *success* or *failure paths* to create a network of filters. The following screenshot shows the Policy Editor screen:



The policy canvas is the large blank area on the screen, while the filters palette is the area on the right that contains the filters. Message filters are grouped together by category (for example, all the content-based filters are displayed together in one group, while all the authentication filters are displayed in a different group). You can build policies by dragging these filters and dropping them on to the canvas.

Important Note:

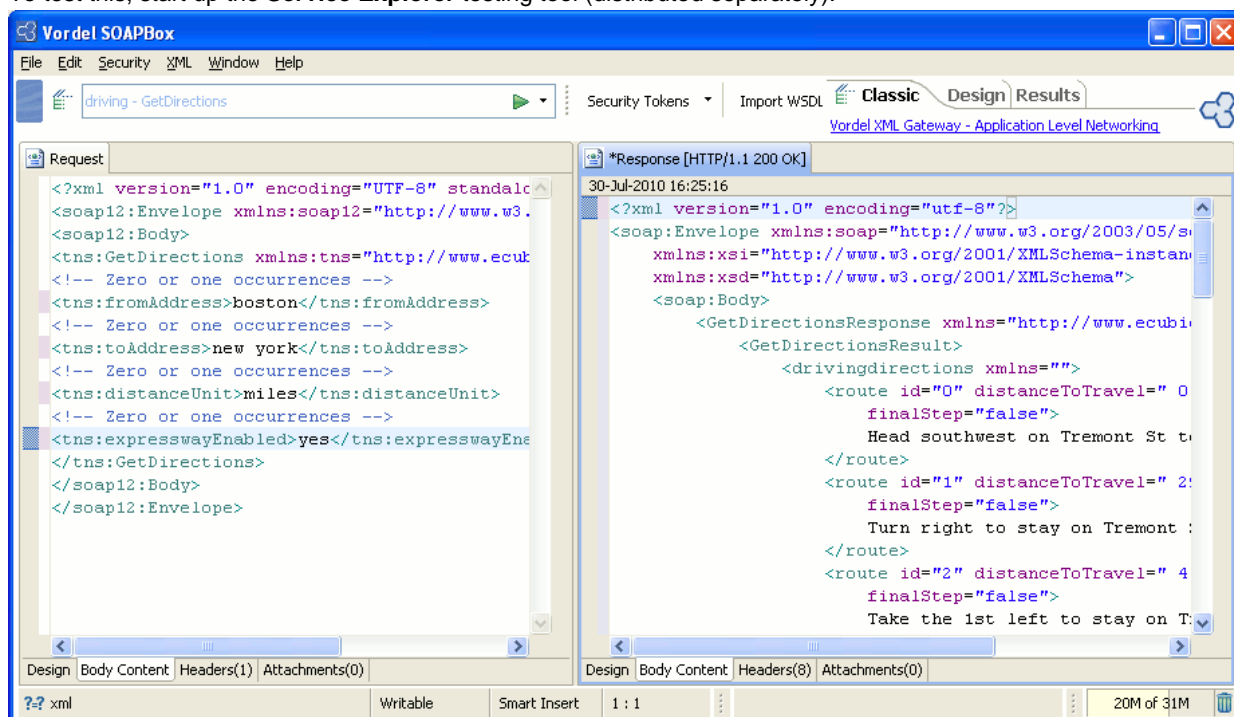
If you have followed the steps outlined above, you can see a new category of filters named **Example Filters** in the filter palette. The **Example Filters** category is expanded in the example screenshot.

Creating the Policy

To create a policy, perform the following steps:

1. Right-click **Policies** in the tree view to the left of the Policy Studio, and select **Add Policy**. Enter **Circuit 1** as the

- name of the new policy in the **Policy** dialog.
2. This example creates a policy containing only one filter: the **Reflect** filter. This filter simply echoes the client message back to the client. The **Reflect** filter is found in the **Utility** filter group. Drag this filter on to the editor.
3. Enter a name for the filter (or use the default) in the field provided. Select the default value (200) for the **HTTP response status code**, and click **Finish**. The policy needs to have a start filter, so right-click the **Reflect** filter, and choose **Set as Start**.
4. You must now configure the Process to invoke the new policy. On the **Services** tab in Policy Studio, select the Process (for example **Enterprise Gateway**) -> **Default Services**. Right-click **Path: /**, and select **Edit**.
5. Enter the following values on the **Configure Relative Path** dialog:
 - **Relative Path:**
Keep **/** in this field, meaning that the Process invokes the policy selected below for all requests received on this path.
 - **Policy:**
Select **Circuit 1** to configure the server to send all requests received on the path configured above to our newly configured policy.
6. To force the server to pick up the new configuration, you must deploy the configuration to the server. Click the **Deploy** or press F6.
7. To test this, start up the **Service Explorer** testing tool (distributed separately):

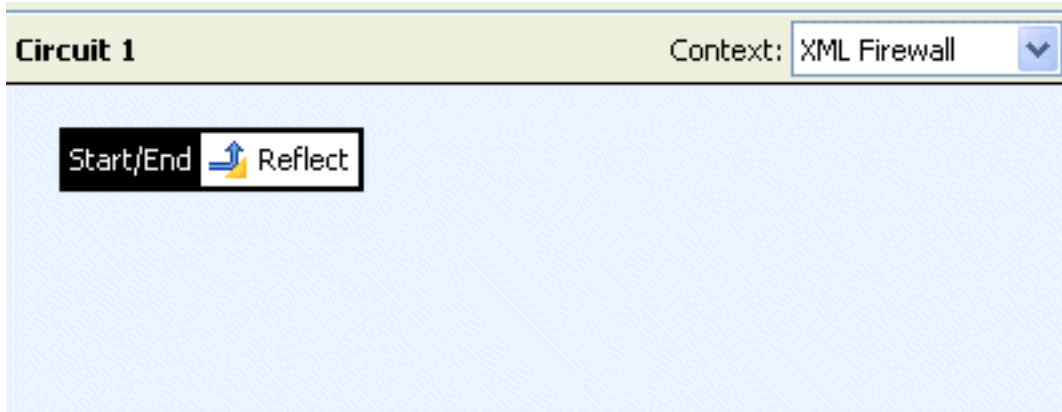


8. Assuming your HTTP interface is listening on port 8080, you can configure **Service Explorer** to send a request to: `http://HOST:8080/`, where **HOST** is the hostname or IP address of the machine on which the server is running. Note that you send to **/** because, earlier, you configured the firewall to filter requests received on this relative path.
9. Copy any SOAP message into the **Request** panel of **Service Explorer**. Click the triangular green **Send** button to send the message to the server, which echoes it back to the client using the **Reflect** filter configured earlier. When the message has been returned to **Service Explorer**, try changing the message slightly to assure yourself that the correct message is actually being returned.

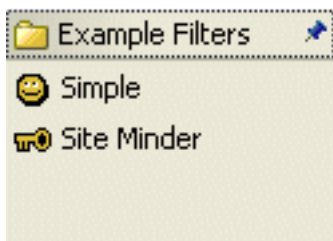
Finally, it is time to add the `SimpleFilter` to the policy, and to test its functionality.

Step 8: Configure the SimpleFilter

The final section of this tutorial adds the `SimpleFilter` to the policy built in the previous section. Currently, the policy consists of only one filter, the **Reflect** filter:



The **SimpleFilter** is found in the **Example Filters** category of the **Filter Palette** on the Policy Studio.



Configuring the Filter

To configure the new filter, perform the following steps:

1. Drag and drop the **SimpleFilter** on to the policy canvas. The configuration screen is displayed as follows:

Configure a new 'Simple' filter

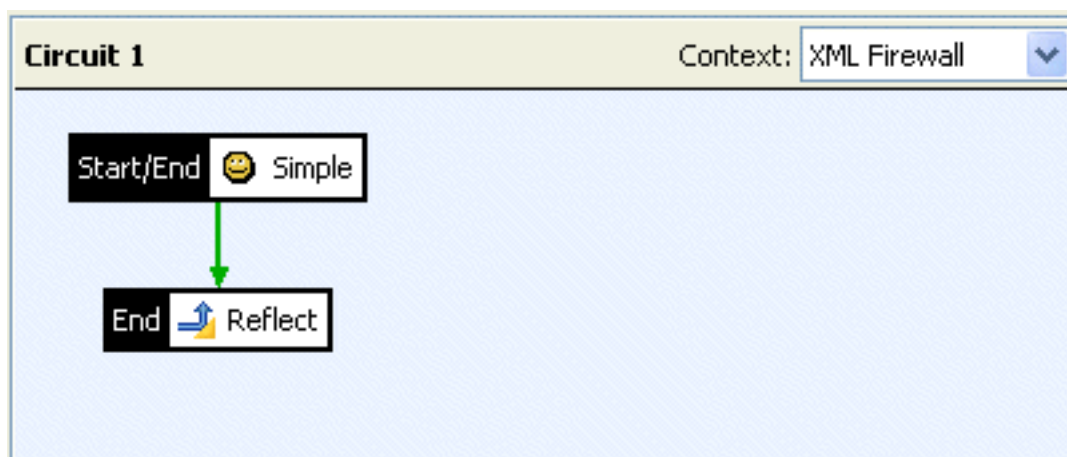
Simple Filter Configuration

Configure parameter values for the Simple Filter

Filter <u>N</u> ame:	Simple
Parameter 1	a
Parameter 1 Namespace	http://startvbdotnet.com/web/
Parameter 2	b
Parameter 2 Namespace	http://startvbdotnet.com/web/

Buttons: **Help** **< Back** **Next >** **Finish** **Cancel**

2. Because the type definition for the `SimpleFilter` entity contained default values, the input fields on the configuration screen are pre-populated with these default values.
3. Before completing the configuration, make sure the help system is working correctly. Remember that in [Step 4](#) of this tutorial, you added a mapping to the `contexts.xml` file (in the `/plugins/com.vordel.rcp.policystudio.resources_(version ...)` folder of your Policy Studio installation.) If you have not done so yet, this is explained in [Step 4](#). After restarting Policy Studio, you can try clicking the **Help** button while editing the `SimpleFilter` configuration.
4. Right-click the **Simple** node, and select **Set as Start** from the context menu. Connect the **Simple** node to the **Reflect** node with a *success* path. You can do this by clicking the **Success Path** arrow, and then clicking the **Simple** node, followed by clicking the **Reflect** node. The policy is now displayed as follows:



5. To force the server to pick up the new configuration, you must refresh the server. Click the **Deploy** button in Policy Studio (or press F6).
6. You can now test the configuration to make sure that it performs as expected (that it can correctly add the two numbers together). Load the appropriate SOAP message into the **Service Explorer** by selecting the **File -> Samples -> Add two numbers** menu option. The following SOAP message is loaded:

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Add xmlns="http://startvbdotnet.com/web/">
      <a>1</a>
      <b>2</b>
    </Add>
  </soap:Body>
</soap:Envelope>
```

Important Note:

Note the presence of the `<a>` and `` elements in the SOAP message, and the namespace declaration in the `<Add>` element. These elements and their corresponding namespaces match the values configured in the **SimpleFilter** earlier.

Make sure to send the message to the same address as before by entering `http://localhost:8080/` as the URL. The **WSDL** field is not needed and can be removed. Press the **Run** (Send) button when you have done this to send the message to the server.

The following response is returned to **Service Explorer**:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <AddResponse xmlns="http://startvbdotnet.com/web/">
      <AddResult>3</AddResult>
    </AddResponse>
```

```
</soap:Body>  
</soap:Envelope>
```

The value of the `<AddResult>` element is 3, which indicates that the newly added filter has worked successfully.

Conclusion

This tutorial described a working example of how to write a message processing filter using the Oracle Enterprise Gateway and how to integrate it into a policy. You should now try to build your own filter by following a similar sequence of steps to those outlined in this tutorial.

If you have any queries on the content of this document, please contact the [Support Team](#) with your questions.

Scripting Language Filter

Overview

The **Scripting Filter** uses [Java Specification Request \(JSR\) 223](http://java.sun.com/developer/technicalArticles/J2SE/Desktop/scripting/) [http://java.sun.com/developer/technicalArticles/J2SE/Desktop/scripting/] to embed a scripting environment in the Enterprise Gateway's core engine. This enables you to write bespoke JavaScript or Groovy code to interact with the message as it is processed by the Enterprise Gateway. You can get, set, and evaluate specific message attributes with this filter.

Because the scripting environment is embedded in the Enterprise Gateway engine, it has access to all Java classes on the Enterprise Gateway's classpath, including all JRE classes. If you wish to invoke a Java object, you must place its corresponding class file on the Enterprise Gateway classpath. The recommended way to add classes to the Enterprise Gateway classpath is to place them (or the JAR files that contain them) in the `INSTALL_DIR/ext/lib` folder. For more details, see the `readme.txt` in this folder.

Some typical uses of the **Scripting Filter** include the following:

- Check the value of a specific message attribute
- Set the value of a message attribute
- Remove a message attribute
- DOM processing on the XML request or response

Writing a Script Filter

To write a script filter, you must implement the `invoke()` method. This method takes a `com.vordel.circuit.Message` object as a parameter and returns a boolean result.

The Enterprise Gateway provides a **Script library** that contains a number of pre-written `invoke()` methods to manipulate specific message attributes. For example, there are `invoke()` methods to check the value of the `SOAPAction` header, to remove a specific message attribute, to manipulate the message using the DOM, and another to assign a particular role to a user.

You can access the script examples provided in the **Script library** by clicking the **Show script library** button on the filter's main configuration screen.

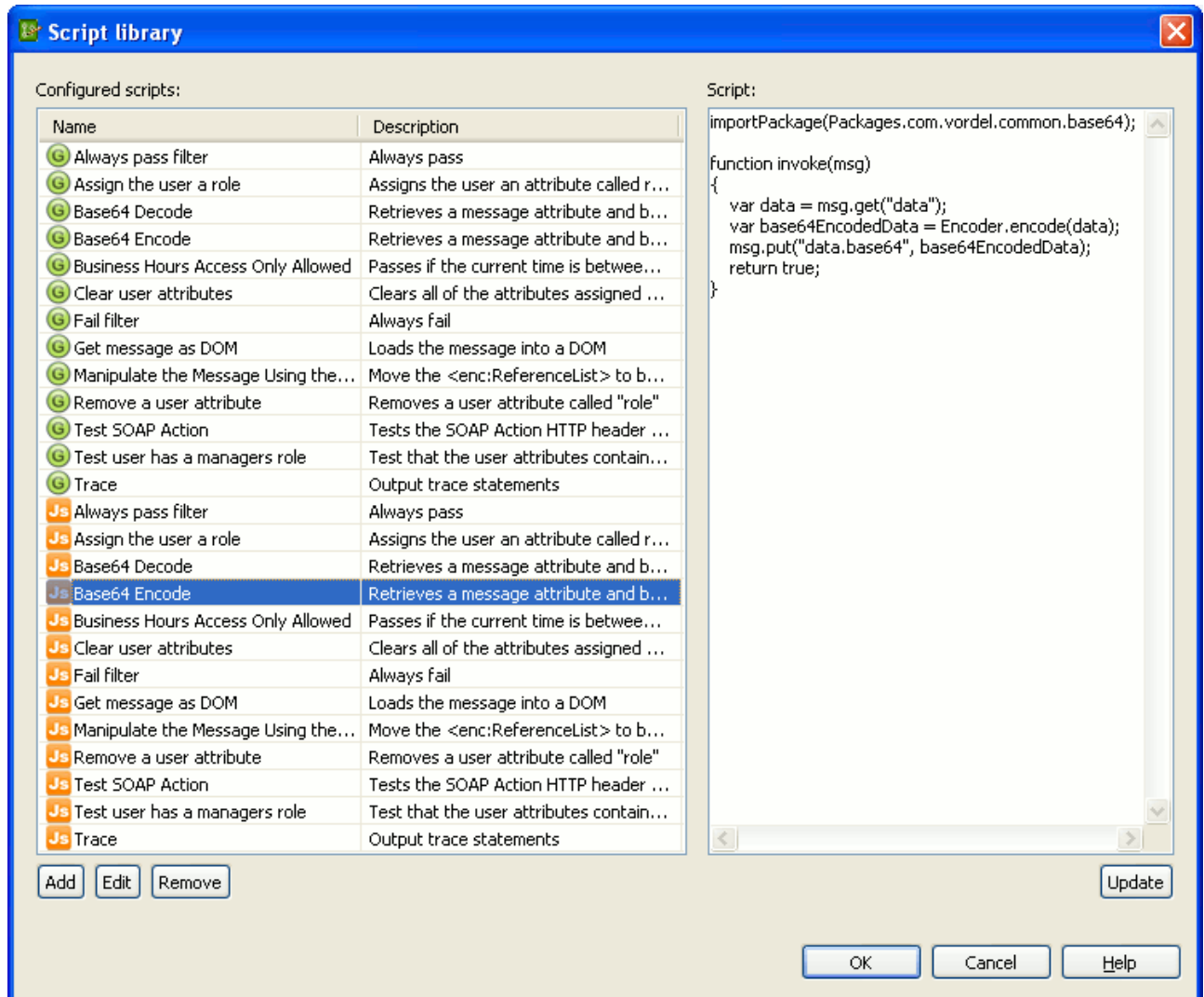
For a complete list of available message attributes, see the [Message Attribute Reference](#).

Configuring a Script Filter

You can write the JavaScript or Groovy code in the text area on the **Script** tab. A JavaScript function skeleton is displayed by default. Use this skeleton code as the basis for your JavaScript code. You can also load an existing JavaScript or Groovy script from the **Script library** by clicking the **Show script library** button.

On the **Script library** dialog, click any of the **Configured scripts** in the table to display the script in the text area on the right. You can edit a script directly in this text area. Make sure to click the **Update** button to store the updated script to the **Script library**.

You can add a new script to the library by clicking the **Add** button, which displays the **Script Details** dialog. Enter a **Name** and a **Description** for the new script in the fields provided. By default, the **Language** field is set to JavaScript, but you can also select Groovy from the drop-down list. You can then write the script in the **Script** text area.

**Important Note:**

When writing the JavaScript or Groovy code, you should note the following:

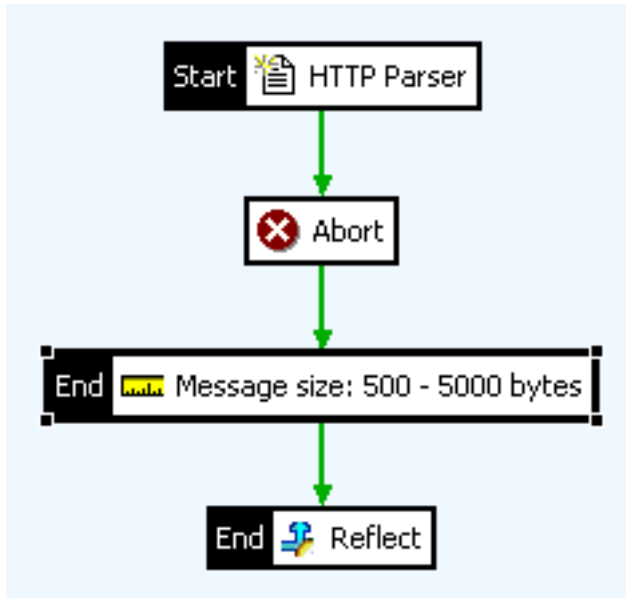
- The `invoke()` method *must* be implemented.
- The `invoke()` method takes a `com.vordel.circuit.Message` object as a parameter, and returns a boolean.
- You can obtain the value of a message attribute using the `getProperty` method of the `Message` object.

Abort Filter

Overview

The **Abort** filter can be used to force a policy to throw an exception. It can be used to test the behavior of the policy when an exception occurs.

For example, to quickly test how the policy behaves when a **Message Size** filter throws an exception, it is possible to place an **Abort** filter before it in the policy. The following policy diagram illustrates the setup:



Configuration

Enter a name for the filter in the **Name** field.

Configuration Web Service

Overview

This filter is only used by the configuration Management Service and should not need to be configured.

Copy/Modify Attributes

Overview

The **Copy/Modify Attributes** filter copies the values of message or user attributes to other message or user attributes. It is also possible to set the value of a message or user attribute to a user-specified value.

Configuration

The configured attribute-copying rules are listed in the table. To add a new rule, click the **Add** button.

The **Copy/Modify Attributes** screen can be used to copy a message or user attribute to a different message or user attribute. The **From Attribute** represents the source attribute, while the **To Attribute** represents the destination attribute.

The attribute value can be copied from 3 possible sources:

- **Message:**
Select this option to copy the value of a message attribute. The name of the source attribute should be specified in the **Name** field.
- **User:**
This option should be selected if a user attribute stored in the `attribute.lookup.list` is to be copied. Enter the name (and namespace if the attribute was extracted from a SAML attribute assertion) of the user attribute in the **Name** and **Namespace** fields.
If there are multiple values stored in the `attribute.lookup.list` for the attribute entered in the **Name** field, only the first value will be copied.
- **User Entered Value:**
Select this option to copy a user-specified value to an attribute. Enter the new attribute value in the **attribute** field. You can enter a property to represent the value of a message attribute instead of entering a specific value directly. The syntax for entering properties is as follows:
`${authentication.subject.id}`
In this case the value of the `authentication.subject.id` attribute is copied to the named attribute.

The message can be copied to either of 2 types of attributes:

- **Message:**
The attribute can be copied to any message attribute. The name of the attribute should be specified in the **Name** field.
- **User:**
Select this option if the attribute or value should be copied to a user attribute stored in the `attribute.lookup.list`. Specify the name and namespace (if necessary) of this attribute in the **Name** and **Namespace** fields.
If there are multiple values stored in the `attribute.lookup.list` for the attribute entered in the **Name** field of the **From attribute** section, the attribute value will be copied to the first occurrence of the attribute name in list.

Check the **Create list attribute** checkbox if the new attribute can contain several items, i.e. it is a list.

Execute External Process

Overview

The **Execute external process** filter enables you to execute an external process from a policy circuit. You can use this filter to execute any external process (for example, start an SSH session to connect to another machine, run a script, or send an SMS message).

Configuration

To configure the **Execute external process** filter, specify the following fields:

Name	Name of the filter to be displayed in a circuit. Defaults to Execute process .
-------------	---

Command tab

This tab includes the following fields:

Command to execute	Specify the full path to the command that you wish to execute (for example, <code>c:\cygwin\bin\mkdir.exe</code>).
Arguments	Click the Add button to add arguments to your command. Specify an argument in the Value field (for example, <code>dir1</code>), and click OK . Repeat these steps to add multiple arguments (for example, <code>dir2</code> and <code>dir3</code>).
Working directory	Specify the directory to run the command from. You can specify this using a property, which is expanded to the specified value at runtime. Defaults to <code>\${VINSTDIR}</code> , where <code>VINSTDIR</code> is the root of your Enterprise Gateway installation.
Expected exit code	Specify the expected exit code for the process when it has finished. Defaults to 0.
Kill if running longer than (ms)	Specify the number of milliseconds after which the running process is killed. Defaults to 60000.

Advanced tab

This tab includes the following fields:

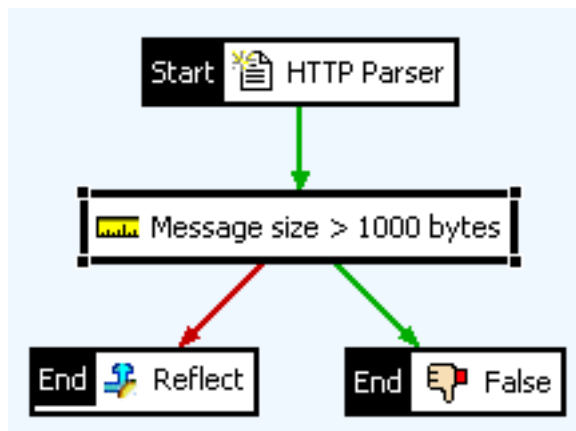
Environment variables to set	Click the Add button to add environment variables. In the Environment variable dialog, specify an Environment variable name (for example, <code>JAVA_HOME</code>) and a Value (for example, <code>c:\jdk1.6.0_18</code>), and click OK . Repeat to add multiple variables.
Block till process finished	Select whether to block until the process is finished in the checkbox. This is enabled by default.

False Filter

Overview

The **False** filter can be used to force a path in the policy to return false. This can be useful in cases where you want to create a *false positive* path in a policy.

The following policy parses the HTTP request and then runs a **Message Size** filter on the message to make sure that the message is no larger than 1000 bytes. If we want to make sure that the message cannot be greater than this size, we can connect a **False** filter to the *success* path of the **Message Size** filter. This means that an exception will be raised if a message exceeds 1000 bytes in size.



Configuration

Enter a name for the filter in the **Name** field.

HTTP Parser

Overview

The **HTTP Parser** parses the HTTP request headers and body. As such, it acts as a "barrier" in the policy to guarantee that the entire content has been received before any other filters are invoked. It requires the `content.body` attribute.

The **HTTP Parser** filter forces the server to do "store-and-forward" routing instead of the default "cut-through" routing, where the request is only parsed on-demand. This filter can be used as a simple test to ensure that the message is XML, for example.

Configuration

Enter a name for the filter in the **Name** field.

Invoke Policy per Message Body

Overview

In cases where Enterprise Gateway receives a multipart related MIME message, the **Invoke Policy per Message Body** filter can be used to pass each body part to a specified policy for processing.

So, for example, if other XML documents have been attached to an XML message (using the SOAP with Attachments specification perhaps), each of these documents can be passed to an appropriate policy where they can be processed by the full compliment of message filters.

Configuration

Complete the following fields:

Name:

Enter a name for the filter in this field.

Policy Shortcut:

Select the policy to invoke for each MIME body part in the message. Each body part will be passed to the selected policy in turn. The filter will fail if the selected policy fails for *any* of the passed body parts.

Maximum Level to Unzip:

In cases where a MIME body part is a MIME message itself, which may, in turn, contain more multi-part messages, the setting here determines how many levels of enveloped MIME messages to attempt to unzip. A default value of "2" levels ensures that the server will not attempt to unwrap unnecessarily *deep* MIME messages.

If one of the body parts is actually an archive file (e.g. tar or zip), this setting determines the maximum depth of files to unzip in cases where the archive file contains other archive files, which may contain others, and so on.

Locate XML Nodes

Overview

You can use the **Locate XML Nodes** filter to select a number of nodes from an XML message. The selected nodes are stored in a message attribute, which is typically used by a Signature or XML Encryption filter later in a circuit.

The primary use of the **Locate XML Nodes** filter is where a series of circuits is auto-generated by importing a WSDL (Web Services Description Language) file that contains WS-Policy assertions. Because there may be many different WS-Policy assertions that describe elements in the message that must be signed (for example, the **Locate XML Nodes** filter is used to build up the node list of elements). Eventually, this node list is passed into the **Sign Message** filter (using a message attribute) so that a single Signature can be created that covers all the relevant parts.

In general, however, you can also use this filter in similar cases where the message content that must be signed depends on the actual content of the message. For example, it is possible that a given circuit runs a number of XPath expressions on a message where each XPath expression is checking for a particular element. If that element is found, it can be marked as an element to be signed/encrypted by selecting that element in the **Locate XML Nodes** filter. This means that only a single Signature/XML Encryption filter must be configured, with each path feeding back into this filter and passing in the message attribute that contains the nodes set for each specific case.

Configuration

As explained earlier, nodes can be selected using any combination of **Node Locations**, **XPaths**, and/or **Message Attributes**. The following sections explain how to use each different mechanism and how to store the selected nodes in a message attribute.

Node Locations:

The simplest way to select nodes is using the pre-configured elements listed in the table on the **Node Locations** tab. The table is pre-populated with elements that are typically found in secured SOAP messages, including, the SOAP Body, WSSE Security Header, WS-Addressing headers, SAML Assertions, WS UsernameToken, and so on.

The elements selected here are found by traversing the SOAP message as a DOM and finding the element name with the correct namespace and with the selected index position (for example, the 1st *Signature* element from the `http://www.w3.org/2000/09/xmldsig#` namespace).

You can select the checkbox in the **Name** column of the table to select the corresponding node. You can select any number of **Node Locations** in this manner.

If you want to locate an element that is not already present in the table, you can add a new Node Location by clicking the **Add** button below the table. In the **Locate XML Nodes** dialog, enter the name of the element, its namespace, and its position in the message using the **Element Name**, **Namespace**, and **Index** fields.

If you wish to select this node for encryption purposes, you must select an appropriate **Encryption Type**. For example, WS-Security Policy mandates that when encrypting the SOAP Body that only its contents are encrypted and not the SOAP Body element itself. This means that the `<xenc:EncryptedData>` is inserted as a direct child of the SOAP Body element. In this case, you should select the **Encrypt Node Content** radio button.

However, in most other cases, it is typically the entire node that gets encrypted. For example, when encrypting a `<wsse:UsernameToken>`, the entire node should be encrypted. In this case, the `<EncryptedData>` element replaces the `<UsernameToken>` element. To encrypt the entire node in this manner, select the **Encrypt Node** radio button.

XPath Expressions:

In cases where you want to select nodes that exist under a more complicated element hierarchy, it may be necessary to use an XPath expression to locate the required nodes. The **XPaths** table is pre-populated with a number of XPath expressions to locate SOAP elements and common security elements, including SAML Assertions and SAML Responses.

To select an existing XPath expression, you can select the checkbox next to the **Name** of the appropriate XPath expres-

sion. You can select any number of XPath expressions in this manner.

To add a new XPath expression, click the **Add** button. You must enter a name for the XPath expression in the **Name** field. You can then enter the XPath expression in the **XPath Expression** field. For more information on configuring this dialog, see the [Configuring XPath Expressions](#) topic.

If you wish to select this node for encryption purposes, you must select an appropriate **Encryption Type**. For example, WS-Security Policy mandates that when encrypting the SOAP Body that only its contents are encrypted and not the SOAP Body element itself. This means that the `<xenc:EncryptedData>` is inserted as a direct child of the SOAP Body element. In this case, you should select the **Encrypt Node Content** radio button.

However, in most other cases, it is typically the entire node that gets encrypted. For example, when encrypting a `<wsse:UsernameToken>`, the entire node should be encrypted. In this case, the `<EncryptedData>` element replaces the `<UsernameToken>` element. To encrypt the entire node in this manner, select the **Encrypt Node** radio button.

Message Attribute:

Finally, you can also retrieve nodes that have been previously stored in a named message attribute. In such cases, another filter extracts nodes from the message and stores them in a named message attribute (for example, `node.list`). The **Locate XML Nodes** filter can then extract these nodes and store them in the message attribute configured in the **Message Attribute Name** field below:

Message Attribute in which to place list of nodes:

At runtime, the **Locate XML Nodes** filter locates and extracts the selected nodes from the message. It then stores them in the specified message attribute. For example, if you wish to sign the selected nodes, it would make sense to store the nodes in a message attribute called `sign.nodeList`, which would then be specified in the **Sign Message** filter. Alternatively, if you wish to encrypt the selected nodes, you could store the nodes in the `encrypt.nodeList` message attribute, which would then be specified in the **XML Encryption Properties** filter.

Finally, you must specify whether you want the selected nodes to overwrite any nodes that may already exist in the specified attribute, or if you want to append to any existing nodes. You can also decide to reset the contents of the message attribute. Select the appropriate radio button depending on your requirements.

Pause Filter

Overview

The **Pause** filter is mainly used for testing purposes. A **Pause** filter causes a policy to sleep for a specified amount of time.

Configuration

Enter an appropriate name for the filter in the **Name** field. When the filter is executed in a policy, it sleeps for the time specified in the **Pause for** field. The sleep time is specified in milliseconds.

Policy Shortcut

Overview

The **Policy Shortcut** filter enables you to reuse the functionality of one policy in another policy. For example, you could create a policy called **Security Tokens** that inserts various security tokens into the message. You can then create a policy that calls this policy using a **Policy Shortcut** filter.

In this way, you can adopt a design pattern of building up reusable pieces of functionality in separate policies, and then bringing them together when required using a **Policy Shortcut** filter. For example, you can create modular reusable policies to perform specific tasks, such as authentication, content-filtering, or logging, and call them as required using a **Policy Shortcut** filter.

For details on how to create a sequence of policy shortcuts in a single policy, see the [Policy Shortcut Chain](#) filter.

Configuration

Complete the following fields to configure the **Policy Shortcut** filter:

Name:

Enter an appropriate name for the filter.

Policy Shortcut:

Select the policy that you want to reuse from the tree. The policy in which this **Policy Shortcut** filter is configured calls the selected policy when it is executed.

Policy Shortcut Chain

Overview

The **Policy Shortcut Chain** filter enables you to run a series of configured policies in sequence without needing to wire up a policy containing several **Policy Shortcut** filters. This enables you to adopt a design pattern of creating modular reusable policies to perform specific tasks, such as authentication, content-filtering, or logging. You can then link these policies together into a single, coherent sequence using this filter.

Each policy in the **Policy Shortcut Chain** is evaluated in succession. The evaluation proceeds as each policy in the chain passes, until finally the filter exits with a pass status. If a policy in the chain fails, the entire **Policy Shortcut Chain** filter also fails at that point.

The **Policy Shortcut Chain** is available from the **Utility** category of filters. You can drag and drop this filter from the filter palette to the policy editor canvas in the Policy Studio.

General Configuration

Complete the following general setting:

Name:

Enter an intuitive name for the filter in this field. For example, the name might reflect the business logic of the policies that are chained together in this filter.

Add a Policy Shortcut

Click the **Add** button to display the **Policy Shortcut Editor** dialog, which enables you to add a policy shortcut to the chain. Complete the following settings in this dialog:

Shortcut Label:

Enter an appropriate name for this policy shortcut.

Evaluate this shortcut when executing the chain:

Select whether to evaluate this policy shortcut when executing a policy shortcut chain. When this option is selected, the policy shortcut has an **Active** status in the table view of the policy shortcut chain. This option is selected by default.

Choose a specific Policy to execute:

Select this option if you wish to choose a specific policy to execute. This option is selected by default.

Policy:

Click the browse button next to the **Policy** field, and select a policy to reuse from the tree (for example, **Health Check**). The policy in which this **Policy Shortcut Chain** filter is configured calls the selected policy when it is executed.

Choose a Policy to execute by label:

Select this option if you wish to choose a policy to execute based on a specific policy label. For example, this enables you to use the same policy on all requests or responses, and also enables you to update the assigned policy without needing to rewire any existing policy circuits. For more details, see the [Configuring Global Policies](#) topic.

Policy Label:

Click the browse button next to the **Policy Label** field, and select a policy label to reuse from the tree (for example, **Gateway request policy (Health Check)**). The policy in which this **Policy Shortcut Chain** filter is configured calls the selected policy label when it is executed.

Click **OK** when finished. You can click **Add** and repeat as necessary to add more policy shortcuts to the chain. You can alter the sequence in which the policies are executed by selecting a policy in the table and clicking the **Up** and **Down** buttons on the right. The policies are executed in the order in which they are listed in the table.

Edit a Policy Shortcut

Select an existing policy shortcut, and click the **Edit** button to display the **Policy Shortcut Editor** dialog. Complete the following settings in this dialog:

Shortcut Label:

Enter an appropriate name for this policy shortcut.

Evaluate this shortcut when executing the chain:

Select whether to evaluate this policy shortcut when executing a policy shortcut chain. When this option is selected, the policy shortcut has an **Active** status in the table view of the policy shortcut chain.

Policy / Policy Label:

Click the browse button next to the **Policy** or **Policy Label** field (depending on whether you chose a specific policy or a policy label when creating the policy shortcut). Select a policy or policy label to reuse from the tree (for example, **Health Check** or **Gateway request policy (Health Check)**). The policy in which this **Policy Shortcut Chain** filter is configured calls the selected policy or policy label when it is executed.

Quote of the Day

Overview

The **Quote of the Day** filter is a useful test utility for returning a simple SOAP response to a client. The Enterprise Gateway will wrap the quote in a SOAP response, which can then be returned to the client.

Configuration

Simply enter the quote in the **Quotes** textarea. This quote can be returned in a SOAP response to the client by setting the **Reflect** filter to be the successor of this filter in the policy.

The **Quote of the Day** filter can also load a file containing a list of quotes at run-time. In this case, a random quote from the file will be returned to the client in the SOAP response. Each quote should be delimited by a '%' character on a newline. This is analogous to the *BSD fortune format*. The format of this file is shown in the following example:

```
Most powerful is he who has himself in his own power.
%
All science is either physics or stamp collecting.
%
A cynic is a man who knows the price of everything and the value of nothing.
%
Intellectuals solve problems; geniuses prevent them.
%
If you can't explain it simply, you don't understand it well enough.
```

The quotes can, of course, be simply entered in this format into the **Quotes** textarea to achieve the same goal.

The following example shows a SOAP response returned by the Enterprise Gateway to a client who requested the **Quote of the Day** service:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header/>
  <s:Body xmlns:oracle="www.oracle.com">
    <oracle:getQuoteResponse>
      Every cloud has a silver lining
    </oracle:getQuoteResponse>
  </s:Body>
</s:Envelope>
```

Reflect Filter

Overview

The **Reflect** filter echoes the HTTP request headers, body, and attachments back to the client.

Configuration

Enter a name for the filter in the **Name** field. Specify an HTTP status response code to return to the client in the **HTTP Response Code Status** field.

Reflect Message And Attributes Filter

Overview

The **Reflect Message and Attributes** filter echoes the HTTP request headers, body, and attachments back to the client. It also echoes back the message attributes that were stored in the message at the time when the message completed the policy.

Configuration

Enter a name for the filter in the **Name** field.

Set Response Status

Overview

The **Set Response Status** filter is used to explicitly set the response status of a call. This status is then recorded as a message metric for use in reporting.

This filter is primarily used in cases where the Fault handler for a policy is actually a Policy Shortcut. If the Policy Shortcut passes, the overall fail status still exists. The **Set Response Status** filter can then be used to explicitly set the response status back to "pass", if necessary.

This filter should only be used under very rare circumstances and under advice from the Oracle support team.

Configuration

Name:

Enter an intuitive name for the filter in this field.

Response Status:

Set the response status to **Pass** or **Fail** by selecting the appropriate radio button.

Set Message Attribute

Overview

The simple **Set Message Attribute** filter allows you to set the value of a message attribute.

Configuration

Complete the following fields to configure the **Set Message Attribute** filter.

- **Name:**
Enter a name for the filter in the **Name** field.
- **Attribute Name:**
Enter the name of the message attribute in which you want to store a value.
- **Attribute Value:**
Enter the value of the message attribute specified above.

String Replace Filter

Overview

The **String Replace** filter enables you to replace all or part of the value of a specified message attribute. You can use this filter to replace any specified string or substring in a message attribute. For example, changing the `from` attribute in an email, or changing all or part of a URL.

Configuration

To configure the **String Replace** filter, specify the following fields:

Name	Name of the filter to be displayed in a circuit. Defaults to String Replace . This field is required.
Message Attribute	Select the name of the message attribute to be replaced from the drop-down list. This field is required. If the value of this attribute is not specified, a <code>MissingPropertyException</code> is thrown, which results in a <code>CircuitAbortException</code> .
Specify Destination Attribute	By default, the value of the specified Message Attribute is both the source and destination, and is therefore overwritten. If you wish to specify a different destination attribute, select this checkbox to enable the Destination Attribute field, and select a value from the drop-down list.
Replacement String	The string used to replace the value of the specified source attribute. You can specify this as a property, which is expanded to the specified attribute value at runtime (for example, <code>\${http.request.uri}</code>). This is a required field if you specify the Specify Destination Attribute .
Straight	A match string used to search the value of the specified source attribute. You can specify this as a property, which is expanded to the specified attribute value at runtime. If a straight (exact) match is found, it is replaced with the specified Replacement String .
Regexp	A match string, specified as a regular expression, used to search the value of the specified source attribute. You can specify this as a property, which is expanded to the specified attribute value at runtime. If a match is found, it is replaced with the specified Replacement String .
First Match	If a match is found, only replace the first occurrence.
All Matches	If a match is found, replace all occurrences.

Note:

The possible paths available through this filter are `True` (even if no replacement takes place), and `CircuitAbort`. Under certain circumstances, if the **Replacement String** contains a message attribute property, a `MissingPropertyException` can occur, which results in a `CircuitAbortException`.

Switch on Attribute Value

Overview

The **Switch on attribute value** filter enables you to switch to a specific policy based on the value of a configured message attribute. You can specify various switch cases (for example, contains, is, ends with, matches regular expression, and so on). Specified switch cases are evaluated in succession until a switch case is found, and the policy specified for that case is then executed. You can also specify a default policy, which is executed when none of the switch cases specified in the filter are found.

The **Switch on attribute value** filter is available from the **Utility** category of filters. You can drag and drop this filter from the filter palette on to the policy editor canvas in the Policy Studio, and configure it to meet the requirements of your policy.

Configuration

Complete the following configuration settings in the **Switch on attribute value** screen:

Name:

Enter an intuitive name for the filter. For example, the name might reflect the business logic of a specified switch case.

Switch on attribute:

Enter or select the name of the message attribute to switch on (for example, `http.request.path`). This filter examines the message attribute value, and switches to the specified policy if this value meets a configured switch case.

Case:

You can add, edit, and delete switch cases by clicking the appropriate button on the right. All configured switch cases are displayed in the table on this screen. For more details, see [Adding a Switch Case](#).

Default:

This field specifies the default behavior of the filter when none of the specified switch cases are found in the configured message attribute value. Select one of the following options:

Return result of calling the following policy	Click the browse button, and select a default policy to execute from the dialog (for example, XML Threat Policy). The filter returns the result of the specified policy. This option is selected by default.
Return true	The filter returns true.
Return false	The filter returns false.

Adding a Switch Case

To add a switch case, click the **Add** button, and configure the following fields in the dialog:

Comparison Type:

Select the comparison type that you wish to perform with the configured message attribute. The available options include the following:

- Contains
- Doesn't Contain
- Ends With

- Is
- Isn't
- Matches Regular Expression
- Starts With

All of these options are case insensitive, except for `Matches Regular Expression`.

Compare with:

Enter the value to compare the configured message attribute value with. For example, if you select a **Comparison Type** of `Matches Regular Expression`, enter the regular expression in this field.

Policy:

Click the browse button next to the **Policy** field, and select the policy to execute from the dialog (for example, **Remove All Security Tokens**). The selected policy is executed when this switch case is found.

Click **OK** when finished. You can click **Add**, and repeat as necessary to add more switch cases to this filter. The switch cases are examined in the order in which they are listed in the table. You can alter the sequence in which the switch cases are evaluated by selecting a policy in the table and clicking the **Up** and **Down** buttons on the right.

Time Filter

Overview

The **Time Filter** enables you to block or allow messages on a specified time of day and/or day of week. You can input the time of day directly in the **Time Filter** screen, configure message attributes to supply this information using the Java `SimpleDateFormat`, or specify a cron expression.

You can use the **Time Filter** in any policy circuit (for example, to block messages at specified times and/or days when a Web Service is not available, or has not been subscribed for by a consumer). In this way, this filter enables you to meter the availability of a Web Service and to enforce Service Level Agreements.

General Configuration

Configure the following general options:

Name:

Enter an appropriate name for this filter.

Block Messages:

Select this option if you wish to use this filter to block messages. This is the default option.

Allow Messages:

Select this option if you wish to use this filter to allow messages.

Basic Time Options

Select **Basic** if you wish to block or allow messages at specified times of the day. This is the default option. You can configure following settings:

User defined time:

Select this option to input the times to block or allow messages directly in this screen. This is the default option. Configure the following settings:

From	The time to start blocking or allowing messages from in hours, minutes, and seconds. Defaults to 9:00:00.
To	The time to end blocking or allowing messages in hours, minutes, and seconds. Defaults to 17:00:00.

Time from attribute:

Select this option to specify times to block or allow messages using configured message attributes. You can specify these attributes using properties, which are replaced at runtime with the values of the specified message attributes set in previous filters or messages. You must configure the following settings:

From	Message attribute that contains the time to start blocking or allowing messages from (for example, <code>\$(message.starttime)</code>). Defaults to a time of 9:00:00.
To	Message attribute that contains the time to end blocking or allowing messages (for example, <code>\$(message.endtime)</code>). Defaults to a time of 17:00:00.
Pattern	Message attribute that contains the time format based on

	the Java SimpleDateFormat class (for example, \$(message.pattern)). This enables you to format and parse dates in a locale-sensitive manner. Day, month, years, and milliseconds are ignored. Defaults to a format of HH:mm:ss.
--	---

Days:

If you wish to block or allow messages on specific days of the week, select the checkboxes for those days. For example, you may wish to block messages on Saturday and Sunday.

Advanced Time Options

Select **Advanced** if you wish to block or allow messages at specified times based on a cron expression. Configure the following setting:

Cron Expression:

Enter a cron expression or a message attribute that contains a cron expression in this field. For example, the following cron expression blocks all messages received on April 27 and 28 2011, at any time except those received between 10:00:01 and 10:59:59.

```
* * 0-9,11-23 27-28 APR ? 2011
```

The default value is `* * 9-17 * * ? *`, which specifies a time of 9:00:00 to 17:00:00 every day. For more details on cron expressions, see the [Policy Execution Scheduling](#) topic.

Trace Filter

Overview

The **Trace** filter outputs the current message attributes to the configured trace destination(s). By default, output is traced to the system console.

Configuration

Name:

Enter an appropriate name for the filter.

Trace Level:

Select the level at which you wish to trace output from the drop-down list. `DATA` tracing is the most verbose level, while `FATAL` is the least verbose.

Include Attributes:

Select this option to trace all current message attributes to the configured trace destination.

Include Body:

Select this option if you wish to trace the entire message body.

Indent XML:

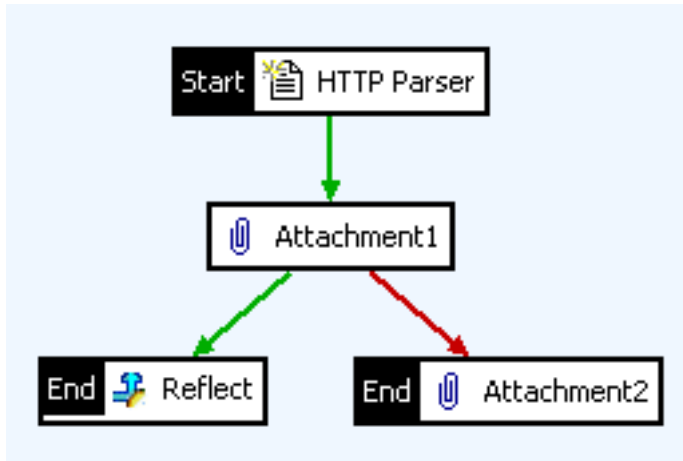
If this option is selected, the XML message is pretty-printed (indented) before it is output to the trace destination.

True Filter

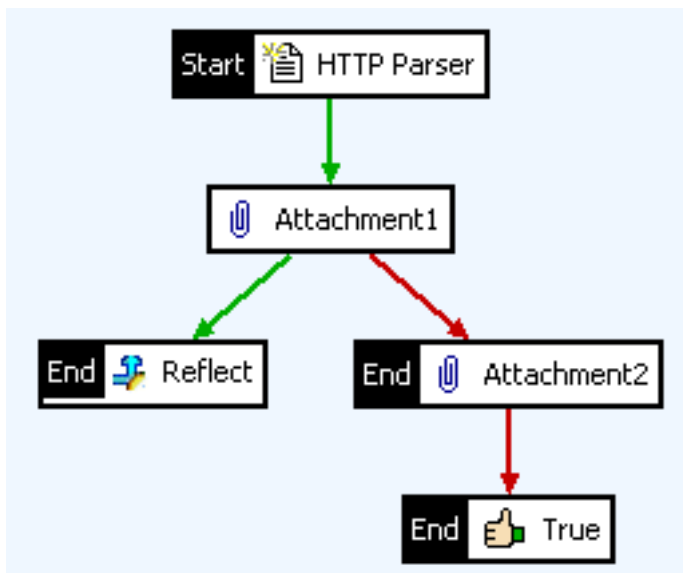
Overview

You can use the **True** filter to force a path in a policy to return true. For example, this can be useful in cases where you want to prevent a path from ending on a false case and consequently throwing an exception.

The following policy parses the HTTP request, and then runs **Attachment1** on the message. If **Attachment1** passes, the message is echoed back to the client by the **Reflect** filter. However, if **Attachment1** fails, the **Attachment2** filter is run on the message. Because this is an *end* node, if this filter fails, an exception is thrown.



By adding a **True** filter to the **Attachment2** filter, this path always ends on a true case, and so does not throw an exception if **Attachment2** fails.



Configuration

Enter an appropriate name for the filter in the **Name** field.

Web Service Filter

Overview

The **Web Service Filter** is used to control and validate requests to the Web Service and responses from the Web Service. Typically, this is automatically generated and populated as a **Service Handler** when a WSDL file is imported into the Web Services Repository. For example, if you import the WSDL file for a Web Service named `ExampleService`, a **Service Handler for ExampleService** filter is automatically generated. However, you can also configure a **Web Service Filter** manually.

In cases when the imported WSDL file contains WS-Policy assertions, a number of policies are automatically created to generate the filters required to generate the relevant security tokens (for example, SAML, WS-Security UsernameToken, and WS-Addressing headers). These policies perform the necessary cryptographic operations (for example, signing and encrypting) to meet the security constraints stipulated by the WS-Policy assertions.

General Settings

Name:

Enter an intuitive name for the filter in this field.

Web Service Context:

Click the browse button to the right of this field to display the hierarchy of Web Services that have been added to the Web Services Repository. Select the checkbox for a Web Service to set the context.

Routing

When routing to a service, you can specify a direct connection to the Web Service endpoint by using the URL in the WSDL or you can override this URL by entering a URL in the field provided. Alternatively, in cases where the routing behavior is more complex, you can delegate to a custom routing policy, which takes care of the added complexity. The top-level radio buttons on the **Routing** tab allow for these alternative routing configurations.

Direct Connection to Service Endpoint:

Select this option to route to either the URL specified in the WSDL or a URL. The radio buttons in the **Routing Details** group enable you to choose between using the URL in the WSDL and providing an override. When providing an override, you can enter the new URL in the **URL** field. Alternatively, you can specify the URL as a property so that the URL is built dynamically at runtime from the specified message attributes (for example `${host}:${port}`, or `${http.destination.protocol}://${http.destination.host}:${http.destination.port}`).

In both cases, you can configure the connection details, such as SSL and other authentication schemes, for the direct connection using the fields in the **Connection Details** group. For more details, see the [Connection Filter](#) tutorial.

Delegate to Routing Policy:

If you wish to use a dedicated routing policy to send messages on to the Web Service, you can select this radio button. Click the browse button next to the **Routing Policy** field. Select the checkbox for the policy that you want to use to route messages, and click the **OK** button.

Validation

The WSDL for a Web Service contains information about the SOAP Action, SOAP Operation, and the data types of the message parts used in a particular SOAP operation. The Enterprise Gateway creates the following implicit validation incoming requests for the Web Service:

- **SOAPAction HTTP Header:**

If a Web Service requires clients to send a certain SOAPAction HTTP header in all requests, the Enterprise Gateway can check the value of this header in the incoming request against the value specified in the WSDL.

- **SOAP Operation and Namespace:**
The WSDL defines the SOAP Operation and namespace to be used in the SOAP request. The SOAP Operation is defined as the first child element of the SOAP Body element. The Enterprise Gateway can check the value of this element in an incoming SOAP request and its namespace against the values specified in the WSDL.
- **Relative Path:**
The filter ensures that requests for this Web Service are received on the same URL as that specified in the `<service>` block of the WSDL.

It is also common for a WSDL document to contain an XML Schema that defines the format and types of the message parts in the request. This is usually the case for document/literal style SOAP requests, where a complete XML Schema is embedded or imported into the `<wsdl:types>` block of the WSDL.

When using a WSDL to import a service into the Web Services Repository, the Policy Studio can extract the XML Schema from the WSDL and configure the Enterprise Gateway to validate incoming requests against it. Select the **Use WSDL Schema** if you want to validate incoming requests against the Schema in the WSDL.

Alternatively, you can create a custom-built policy to validate the contents of incoming requests. To do this, select the **Delegate to Validation Policy** radio button, and then click the browse button next to the **Message Validation Policy** field. Select the policy that you want to use to validate requests, and click the **OK** button.

Message Interception Points

The configuration settings on the **Message Interception Points** tab determine how the request and response messages for the service are processed as they pass through the Enterprise Gateway. Several message interception points are exposed to enable you to hook into different stages of the Enterprise Gateway's request processing cycle.

At each of these interception points, it is possible to run policies that are specific to that stage of the request processing cycle. For example, you can configure a logging policy to run just before the request has been sent to the Web Service and then again just after the response has been received.

Typically, the configuration settings on this screen are automatically configured when importing a service into the Web Services Repository based on information contained in the WSDL. In cases where the WSDL contains WS-Policy assertions, a number of policies are automatically generated and hooked up to perform the relevant security operations on the message. For example, policies are created to insert SAML assertions, WS-Security Username Tokens, WS-Addressing headers, and WS-Security Timestamps into the message. Similarly, filters are created to sign and encrypt the outbound message, if necessary, and to decrypt and validate the signature on the response from the Web Service.

Order of Execution:

It is important to note the following in terms of the order of execution of the message interception points:

- The interception points are executed in the following order:
 1. Request from Client
 2. User-defined Request Hooks
 3. Request to Service
 4. Response from Service
 5. User-defined Response Hooks
 6. Response to Client
- In steps 1, 3, 4, and 6, the execution order is as follows:
 - A) Before Operation-specific Policy
 - B) Operation-specific Policy Shortcuts
 - C) After Operation-specific Policy
- The overall order of all the message interception points is given in the sequence below.

1. Request from Client:

This is the first message interception point, which enables you to run a policy against the request as it is received by the

Enterprise Gateway. Typically, this is where authentication and authorization events should occur.

1A) Before Operation-specific Policy:

This is usually where authentication policies should be configured because it is the earliest point in the request cycle that you can hook into. To select a policy to run at this point, click the browse button, and select the checkbox next to a previously configured policy.

1B) Operation-specific Policy Shortcuts:

If you want to run policies that are specific to the different operations exposed by the Web Service, click the **Edit** button at the bottom of the table to set this up. For example, you may want to perform different validation on requests for the different operations.

On the **Policy Shortcut Editor** dialog, enter the **Operation Namespace** and **Operation Name** in the fields provided. Enter a regular expression used to match the value of the SOAPAction HTTP header in the **SOAPAction Regular Expression** field. Finally, select the policy to run requests for this operation by clicking the browse button next to the **Policy Shortcut** field. Select the checkbox next to the policy that you want to run.

1C) After Operation-specific Policy:

This enables you to run a policy on the request *after* all the operation-level policies have been executed on the request. Select the appropriate policy as described earlier by clicking the browse button.

2. User-defined Request Hooks:

Users should primarily use this interception point to hook in their own custom-built *request* processing policies.

User-defined Request Policy:

Browse to your custom-built request processing policy using the browse button as before.

3. Request to Service:

This enables you to alter the message before it is routed to the Web Service. For example, if the service requires the message to be signed and encrypted, you can configure the necessary policies here.

3A) Before Operation-specific Policy:

This enables run policies on the message *before* the operation-level policies are run. Select the policy to run as outlined in the previous sections.

3B) Operation-specific Policy Shortcuts:

Operation-level policies on the request to the Web Service can be run here. For example, if the input policy for a particular operation requires the body to be signed and encrypted, a **Locate XML Nodes** filter can be run here to mark the required nodes.

3C) After Operation-specific Policy:

This is the last interception point available before the message is routed on to the Web Service. For example, if certain operation-level policies have been run to mark parts of the message to be signed and encrypted, the signing and encrypting filters should be run here.

4. Response from Service:

This is executed on the response returned from the Web Service.

4A) Before Operation-specific Policy:

If the response from the Web Service is encrypted, this interception point enables you to decrypt the message *before* any of the operation-level policies are run on the decrypted message.

4B) Operation-specific Policy Shortcuts:

The policies configured at this point run on specific operation-level responses (for example, `getHelloResponse`) from the Web Service.

4C) After Operation-specific Policy:

This should be used to run policies *after* the operation-level policies have been run. For example, this is the appropriate point to place an XML Signature Verification filter.

5. User-defined Response Hooks:

You should primarily use this interception point to hook in custom-built *response* processing policies.

User-defined Response Policy:

Browse to your custom-built response processing policy using the browse button as before.

6. Response to Client:

This enables you to process the response before it is returned to the client.

6A) Before Operation-specific Policy:

As before, this enables you to process the message with a policy before the operation-level policies are run on the response.

6B) Operation-specific Policy Shortcuts:

The policies listed here are run on each operation response.

6C) After Operation-specific Policy:

This is the very last point at which you can run policies to process the response message before it is returned to the client. For example, if you are required to return a signed and encrypted response message to the client, the signing and encrypting should be done at this point.

WSDL

You can expose an imported WSDL file to clients of the Enterprise Gateway. A client can retrieve a WSDL for a service by appending the WSDL name to the query string of the relative path on which the service is accepting requests. For example, if the service is accepting requests at the URL `http://server:8080/services/getHello`, the client can retrieve the WSDL on the following URL:

`http://server:8080/services/getHello?WSDL`

Important Note:

When the Enterprise Gateway returns the WSDL to the client, it dynamically modifies the service URL of the original WSDL to point to the machine on which the Enterprise Gateway is running. For example, the original WSDL contains the following service element, where `www.service.com` resolves to an internal IP address that is not accessible to the public Internet:

```
<wsdl:service name="GetHelloService">
  <wsdl:port name="GetHelloServiceSoap" binding="tns:ServiceSoap">
    <soap:address location="http://www.service.com/getHello"/>
  </wsdl:port>
</wsdl:service>
```

When the Enterprise Gateway returns this WSDL to the client, it dynamically modifies the value of the `location` attribute to point to the name of the machine hosting the Enterprise Gateway. In the following example, the `location` attribute has been modified to point to the Enterprise Gateway instance running on port 8080 on the `Oracle_SERVER` host:

```
<wsdl:service name="GetHelloService">
  <wsdl:port name="GetHelloServiceSoap" binding="tns:ServiceSoap">
    <soap:address location="http://Oracle_SERVER:8080/getHello"/>
  </wsdl:port>
</wsdl:service>
```

When the client receives the WSDL, it can automatically generate the SOAP request for the `getHello` service, which it then sends to the `Oracle_SERVER` machine on port 8080.

Complete the following fields if you wish to expose the WSDL for this service to clients.

Advertise WSDL to the Client:

Select this option if you wish to publish the WSDL for the selected Web Service. It is important to note that the exposed WSDL represents a *virtualized* view of the back-end Web Service. In this way, clients can retrieve the WSDL from the Enterprise Gateway, generate SOAP requests, and send these requests to the Enterprise Gateway. The Enterprise Gateway then routes the requests on to the Web Service.

WSDL Access Policy:

If you want to configure a policy to control or monitor access to the WSDL for this service, you can select the policy by clicking the browse button to the right of this field. Select the policy that you want to use to run on requests to retrieve the WSDL.

Monitoring

The fields on this tab enable you to configure whether this Web Service stores usage metrics data to a database. This information can then be used by Service Monitor to produce reports showing how and who is calling this Web Service. The following fields are available on this tab:

- **Monitor service usage:**
Select this option if you want to store message metrics for this Web Service.
- **Monitor service usage per client:**
Select this option if you want to generate reports monitoring which authenticated clients are calling which Web Services.
- **Monitor client usage:**
If you want to generate reports on authenticated clients, but are not interested in what Web Services they are calling, select this option, and unselect the **Monitoring service usage per client** checkbox.
- **Which attribute is used to identify the client?:**
Enter the message attribute to use to identify authenticated clients. The default is `authentication.subject.id`, which stores the identifier of the authenticated user (for example, the username or user's X.509 Distinguished Name).

Return WSDL

Overview

The **Return WSDL** filter returns a WSDL file from the Web Services Repository. This filter is configured automatically when auto-generating a policy from a WSDL file and is not normally manually configured.

For details on how to auto-generate a policy from a WSDL file, see the [Web Services Repository](#) topic. For details on how to identify services in the Web Service Repository, see the [Set Web Service Context](#) topic.

Configuration

Enter a name for the filter in the **Name** field.

Set Web Service Context

Overview

The **Set Web Service Context** filter is used in a policy to determine the service to obtain resources from in the Web Service Repository. For example, by pointing this filter at a pre-configured `getQuote` service in the Web Service Repository, the policy knows to return the WSDL for this particular service when a WSDL request is received. The **Return WSDL** filter is used in conjunction with this filter to achieve this.

The **Set Web Service Context** filter is configured automatically when auto-generating a policy from a WSDL file and is not normally manually configured. For a detailed example, see the [Web Services Repository](#) tutorial.

Configuration

Name:

Enter a name for the filter in the **Name:** field.

Service WSDL:

Select a service definition (WSDL file) from the Web Service Repository by selecting the checkbox for the desired service. For more details on adding services to the Web Services Repository, see the [Web Services Repository](#) tutorial.

Monitoring:

The fields on this tab enable you to configure whether this Web Service stores usage metrics data to a database. This information can be used by Service Monitor to produce reports showing how and who is calling this Web Service. The following fields are available on this tab:

- **Monitor service usage:**
Select this option if you want to store message metrics for this Web Service.
- **Monitor service usage per client:**
Select this option if you want to generate reports monitoring which authenticated clients are calling which Web Services.
- **Monitor client usage:**
If you want to generate reports on authenticated clients, but are not interested in what Web Services they are calling, select this option and unselect the **Monitoring service usage per client** checkbox.
- **Which attribute is used to identify the client?:**
Enter the message attribute to use to identify authenticated clients. The default is `authentication.subject.id`, which stores the identifier of the authenticated user (for example, the username or user's X.509 Distinguished Name).

Certificate Chain Check

Overview

Whenever the Enterprise Gateway receives a client's X.509 certificate, either in an XML Signature or as part of an SSL handshake, it needs to determine whether or not that certificate can be trusted. For example, it is a trivial task for a user to generate a structurally sound X.509 certificate. This certificate can then be used to negotiate mutually authenticated connections to publicly available services.

Clearly, this scenario represents a security nightmare for IT administrators - we can't just allow any user to generate their own certificate and use it on the Internet. The server must be able to *trust* the authenticity of the client certificate. Furthermore, it must be able to verify that the certificate originated from a trusted source. To do this a server can perform a *certificate chain check* on the client certificate.

The main purpose of certificate chain validation is to ensure that a certificate has been issued by a trusted source. Typically, in a Public-Key Infrastructure (PKI), a Certificate Authority (CA) is responsible for issuing and distributing certificates. The whole infrastructure is based on the premise of *transitive trust* - if everybody trusts the CA, then everybody transitively trusts the certificates issued by that CA. If entities only trust certificates that have been issued by the CA, they can then reject certificates which have been self-generated by clients.

When a CA issues a certificate, it digitally signs the certificate and inserts a copy of its own certificate into it. This is called a certificate chain. Whenever an application (such as the Enterprise Gateway) receives a client certificate it can extract the issuing CA's certificate from it, and run a certificate chain check to determine whether or not it should trust the CA. If it trusts the CA, it will also trust the client certificate.

The question then begs itself - how does the Enterprise Gateway *trust* a Certificate Authority? The Enterprise Gateway maintains a repository of both trusted CA certificates, and trusted server certificates for use in SSL communications. In order to trust a certain CA, that CA's certificate must be imported into the **Oracle Trusted Certificate Store**.

Configuration

The **Policy Studio** provides an easy-to-use interface for configuring certificate chain validation. This interface allows you to amalgamate CA and server certificates into groups such that if an incoming client certificate has been issued by any of the CAs in the group, the Enterprise Gateway will trust the certificate. Simply enter a name for the group in the **Group Name** field. To populate the new group, simply click the **Add/Edit** button.

By selecting a group from this dropdown, the members of this group will be displayed in the **Certificate Alias** table. To add and/or remove members from the selected group, click the **Add/Edit** button.

Certificates can be added to and removed from new or existing groups using the **Configure Trusted Certificate Groups** dialog which is displayed on clicking the **Add/Edit** button.

The **Configure Trusted Certificate Groups** dialog consists of 2 main tables. The first table lists all certificates currently in the **Trusted Certificate Store**, i.e. those that are trusted by the Enterprise Gateway. The second table lists the members of the group selected in the **Group Name** field.

To add a certificate to a trusted group, simply select it from the **Certificate Store** table, and click the **Add ->** button. The certificate will now appear in the group certificates table. Similarly to remove a certificate from the group, select it from the group certificates table and click the **<- Remove** button. The certificate will now be removed from the group table.

It is also possible to add, remove, and view certificates in the **Trusted Certificate Store** using this dialog. To add a certificate to the **Trusted Certificate Store**, click the **Add** button, which displays the **Import Certificate** dialog.

Browse to the location of the CA certificate file, and enter an **Alias** for the certificate. This **Alias** will be used to uniquely identify the certificate within the Enterprise Gateway.

A certificate can be removed by simply selecting the certificate in the **Trusted Store** table, and then clicking the **Remove**

button. The certificate will be removed from the table, and will no longer be trusted by the Enterprise Gateway.

Finally, it is also possible to examine the details of any one of the certificates in the **Trusted Certificate Store**. To do this, again select a certificate from the **Trusted Certificate** table, and then click on the **View** button.

Certificate Validation

Overview

Whenever the Enterprise Gateway receives an X.509 certificate, either as part of the SSL handshake or as part of the XML message itself, it is important to be able to determine whether that certificate is legitimate or not. Certificates can be revoked by their issuers if it becomes apparent that the certificate is being used maliciously. Such certificates should never be trusted, and so it is very important that the Enterprise Gateway can perform certificate validation.

The Enterprise Gateway uses the following methods/protocols to validate certificates:

OCSP - Online Certificate Status Protocol

OCSP is an automated certificate checking network protocol. The Enterprise Gateway can query the OCSP responder for the status of a certificate. The responder returns whether the certificate is still trusted by the CA that issued it.

CRL - Certificate Revocation Lists

A CRL is a signed list indicating a set of certificates that are no longer considered valid (i.e. revoked certificates) by the certificate issuer. The Enterprise Gateway can query a CRL to find out if a given certificate has been revoked - if the certificate is present in the CRL, it should not be trusted.

XKMS - XML Key Management Services

XKMS is an XML-based protocol for (amongst other things) establishing the trustworthiness of a certificate over the Internet. The Enterprise Gateway can query an XKMS responder to determine whether or not a given certificate can be trusted or not.

Configuration

The Enterprise Gateway can check that the validity of a client certificate using any of the following methods:

1. [OCSP - Online Certificate Status Protocol](#)
2. [CRL - Certificate Revocation Lists](#)
3. [XKMS - XML Key Management Services](#)

Note:- In order to validate a certificate using either an or CRL lookup, the issuing CA's certificate should be trusted by the Enterprise Gateway. This is because for a CRL lookup, the CA's public key is needed to verify the signature on the CRL, and for an OCSP request, the protocol stipulates that the CA's public key must be submitted as part of the request. The issuing CA's public key is not always present in issued certificates, so it is necessary to retrieve it from the Enterprise Gateway's certificate store instead.

OCSP - Online Certificate Status Protocol

1. Enter or select a name for the validation rule in the **Name** field.
2. Select **OCSP** from the **Type** dropdown.
3. Optionally enter a description of the rule in the **Description** field.
4. Select a group of OCSP Responders from the **URL Group** field. The Enterprise Gateway will attempt to connect to the Responders in the selected group in a round-robin fashion. It will attempt to connect to the Responders with the highest priority first, before connecting to Responders with a lower priority. URL Groups can be added, edited, and removed by selecting the **Add**, **Edit**, and **Remove** buttons respectively.
Take a look at the [Configuring URL Groups](#) section below for more information on adding and editing URL groups.
5. Enter the user name of a User whose key will be used to sign status requests sent to the OCSP responder in the **User Name** field.
6. Enter the corresponding password for this user in the **Password** field.
7. If the OCSP Responder signs the OCSP response, and you wish to validate this signature, select the **Validate Re-**

sponse checkbox.

CRL - Certificate Revocation Lists

1. Enter or select a name for the validation rule in the **Name** field.
2. Select *CRL* from the **Type** dropdown.
3. Optionally enter a description of the rule in the **Description** field.
4. Select a previously configured LDAP directory from the **LDAP directory** dropdown list, or add a new one using the **Add** button.

XKMS - XML Key Management Services

1. Enter or select a name for the validation rule in the **Name** field.
2. Select *XKMS* from the **Type** dropdown.
3. Optionally enter a description of the rule in the **Description** field.
4. Enter the URL of the XKMS Responder in the **URL** field.
5. Enter the user name of a User whose key will be used to sign status requests sent to the XKMS responder in the **User Name** field.
6. Enter the corresponding password for this user in the **Password** field.

Configuring URL Groups

The Enterprise Gateway can make connections on a round-robin basis to the URLs listed in a URL group, thus enabling a high degree of failover to external servers. URL groups can be configured by selecting the **Add** and/or **Edit** buttons.

The Enterprise Gateway will attempt to connect to the listed servers according to the priorities assigned to them. So, for example, let's assume there are two "High" priority URLs, one "Medium" URL, and a single "Low" URL configured. Assuming the Enterprise Gateway can successfully connect to the two "High" priority URLs, it will alternate requests between these two URLs only in a round-robin fashion. The other group URLs will not be used at all. If, however, both of the "High" priority URLs become unavailable, the Enterprise Gateway will then try to use the "Medium" priority URL, and only if this fails will the "Low" priority URL be used.

So, in general, the Enterprise Gateway will attempt to round-robin requests over URLs of the same priority, but will use higher priority URLs before lower priority ones. When a new URL is added to the group it is automatically given the highest priority. Priorities can then be changed by selecting the URL and clicking the **Up** and **Down** buttons.

Individual URLs can be added and edited by selecting the URL from the table and clicking on the **Add** and **Edit** buttons respectively.

The following fields should be completed:

- **URL:**
Enter the full URL of the external server.
- **Timeout:**
Specify the timeout in seconds for connections to the specified server.
- **Time:**
Whenever the server becomes unavailable for whatever reason (maintenance, for example), no attempt will be made to connect to that server until the time specified here has elapsed. In other words, once a connection failure has been detected, the next connection to that URL will be made after this amount of time.
- **Username:**
If the specified server requires clients to authenticate to it over 2-way SSL, a **User** must be selected here for authentication.

- **Password:**
Enter the password for this user.
- **Host/IP:**
If the specified server sits behind a proxy server, the host name or IP address of the proxy server must be entered here.
- **Port:**
Enter the port on which the proxy is listening.

Database Connection

Overview

The details entered on the **Configure Database Connection** dialog specify how the Enterprise Gateway connects to the database. The Enterprise Gateway maintains a JDBC pool of database connections to avoid the overhead of setting up and tearing down connections to service simultaneous requests. This pool is implemented using *Apache Commons DB-CP (Database Connection Pools)*. The settings in the **Advanced - Connection pool** section of this screen configure the database connection pool. For details on how the fields on this screen correspond to specific DBCP configuration settings, see the [Database Connection Pool Settings](#) table.

Configuring the Database Connection

Configure the following fields on this screen:

Name:

Enter a name for the database connection in the **Name** field.

URL:

Enter the fully qualified URL of the location of the database.

User Name:

The username to use to access the database.

Password:

The password for the user specified in the **User Name** field.

Initial pool size:

The initial size of the DBCP pool when it is first created.

Maximum number of active connections:

The maximum number of active connections that can be allocated from the JDBC pool at the same time. The default maximum is 8 active connections.

Maximum number of idle connections:

The maximum number of active connections that can remain idle in the pool without extra connections being released. The default maximum is 8 connections.

Minimum number of idle connections:

The minimum number of active connections that can remain idle in the pool without extra connections being created. The default minimum is 8 connections.

Maximum wait time:

The maximum number of milliseconds that the pool waits (when there are no available connections) for a connection to be returned before throwing an exception, or -1 to wait indefinitely. The default time is 10000ms, and a value of -1 indicates an indefinite time to wait.

Time between eviction:

The number of milliseconds to sleep between runs of the thread that evicts unused connections from the JDBC pool.

Number of tests:

The number of connection objects to examine from the pool during each run of the evictor thread. The default number of objects is 3.

Minimum idle time:

The minimum amount of time an object may sit idle in the pool before it is eligible for eviction by the idle object evictor (if any).

Database Connection Pool Settings

The table below shows the correspondence between the fields in the **Advanced - Connection pool** section of the screen and the Apache Commons DBCP configuration properties:

<i>Field Name</i>	<i>DBCP Configuration Property</i>
URL	url
User Name	username
Password	password
Initial pool size	initialSize
Maximum number of active connections	maxActive
Maximum number of idle connections	maxIdle
Minimum number of idle connections	minIdle
Maximum wait time	maxWait
Time between eviction	timeBetweenEvictionRunsMillis
Number of tests	numTestsPerEvictionRun
Minimum idle time	minEvictableIdleTimeMillis

Connection Validation

By default, when the Enterprise Gateway makes a connection, it performs a simple connection validation query. This enables the Enterprise Gateway to test the database connection before use, and to recover if the database goes down (for example, if there is a network failure, or if the database server reboots). The Enterprise Gateway validates connections by running a simple SQL query (for example, a `SELECT 1` query with MySQL). If it detects a broken connection, it creates a new connection to replace it.

Database Query

Overview

The **Database Statement** dialog enables you to enter an SQL query, stored procedure, or function call that the Enterprise Gateway runs to return a specific user's profile from a database.

Configuration

The following fields should be completed on this screen:

Name:

Enter a name for this database query here.

Database Query:

Enter the actual SQL query, stored procedure, or function call in the text area provided. When executed, the query should return a single user's profile. The following are examples of SQL statements and stored procedures:

```
select * from users where username=${authentication.subject.id}
{ call load_user (${authentication.subject.id}, ${out.param}) }
{ call ${out.param.cursor} := p_test.f_load_user(${authentication.subject.id}) }
```

These examples show that properties in the form of message attributes can be used in the query. The property that is most commonly used in this context is the `authentication.subject.id`, which holds the identity of the authenticated user. For a complete list of message attributes, see the Message Attribute Reference.

Statement Type:

The database can take the form of an SQL query, stored procedure, or function call, as shown in the above examples. Select the appropriate radio button depending on whether the database query is an SQL **Query** or a **Stored procedure/function call**

Table Structure:

To process the result set that is returned by the database query, the Enterprise Gateway needs to know whether the user's attributes are structured as rows or columns in the database table.

The following example of a database table shows the user's attributes (Role, Dept, and Email) structured as table columns:

<i>Username</i>	<i>Role</i>	<i>Dept</i>	<i>Email</i>
Admin	Administrator	Engineering	admin@org.com
Tester	Testing	QA	tester@org.com
Dev	Developer	Engineering	dev@org.com

In the following table, the user's attributes have been structured as name-value pairs in table rows:

<i>Username</i>	<i>Attribute Name</i>	<i>Attribute Value</i>
Admin	Role	Administrator
Admin	Dept	Engineering
Admin	Email	admin@org.com
Tester	Role	Testing
Tester	Dept	QA
Tester	Email	tester@org.com
Dev	Role	Developer
Dev	Dept	Engineering
Dev	Email	dev@org.com

If the user's attributes are structured as column names in the database table, select the **attributes as column names** radio button. If the attributes are structured as name-value pair in table rows, select the **attribute name-value pairs in rows** option.

Configuring LDAP Directories

General Configuration

A filter that uses an LDAP directory to authenticate a user or retrieve attributes for a user must have an LDAP directory associated with it. The **Configure LDAP Server** screen is used to configure connection details of the LDAP directory.

When a filter that uses an LDAP directory is run for the first time after a server refresh/restart, the server will bind to the LDAP directory using the connection details configured on the **Configure LDAP Server** dialog. Usually the connection details include the username and password of an administrator user who has read access to all users in the LDAP directory for whom we wish to retrieve attributes or authenticate.

To configure LDAP connection information:

1. Select or enter a name for the LDAP filter in the **Filter Name** dropdown list.
2. Enter the location of the LDAP directory in the **URL** field. The URL is a combination of the protocol (LDAP), the IP address of the host machine and the port number for the LDAP service. By default, port 389 is reserved for LDAP connections. The following is an example of a valid LDAP directory URL:

`ldap://192.168.0.45:389`

Authentication Configuration

If the configured LDAP directory requires clients to authenticate to it, you must select the appropriate authentication method in the **Authentication Type** field. When the Enterprise Gateway connects to the LDAP directory, it will be authenticated using the selected method. The Enterprise Gateway can authenticate to an LDAP directory using the following methods:

- [None](#)
- [Simple](#)
- [Digest-MD5](#)

It is important to note that if any of the following methods are to connect to the LDAP server over SSL, then that server's SSL certificate must be imported into the **Oracle Trusted Certificate Store**.

None:

No authentication credentials need be submitted to the LDAP server for this method. In other words, the client connects anonymously to the server. Typically a client is only allowed to perform "read" operations when connected anonymously to the LDAP server. It is not necessary to enter any details for this authentication method.

Simple:

Simple authentication involves sending a user name and corresponding password in clear-text to the LDAP server. Since the password is passed in clear-text to the LDAP server, it is recommended to connect to the server over an encrypted channel, for example, over SSL.

It is not necessary to specify a **Realm** for the *Simple* authentication method. The realm is only used when a hash of the password is supplied (i.e. for Digest-MD5). However, in cases where the LDAP server contains multiple realms, and the specified user name is present in more than one of these realms, then it is at the discretion of the specific LDAP server as to which user name will actually *bind* to it.

Click the **SSL Enabled** checkbox to force the Enterprise Gateway to connect to the LDAP directory over SSL. In order to successfully establish SSL connections with the LDAP directory, the directory's certificate must be imported into the Enterprise Gateway's certificate store.

Digest-MD5:

With *Digest-MD5* authentication, the server generates some data and sends it to the client. The client encrypts this data with its password according to the MD5 algorithm. The LDAP server then uses the client's stored password to decrypt the data and hence authenticate the user.

The **Realm** field is optional here, but may be necessary in cases where the LDAP server contains multiple realms. If a realm is specified here, the LDAP server will attempt to authenticate the user for the specified realm only.

Signature Location

Overview

It is possible for a given XML message to contain several XML Signatures. For example, consider an XML document (e.g. a company policy approval form) which must be digitally signed by a number of users (e.g. department managers) before being submitted to the ultimate Web Service (e.g. a company policy approval Web Service). Such a message will contain several XML Signatures by the time it is ready to be submitted to the Web Service.

In such cases, where multiple signatures will be present within a given XML message, it is necessary to specify which signature the Enterprise Gateway should use in the validation process.

Configuration

The Enterprise Gateway can extract the signature from an XML message using several different methods. The signature can be extracted:

- [Using WS-Security Actors](#)
- [From the SOAP header](#)
- [Using XPath](#)

Select the most appropriate method from the **Signature Location** dropdown. Your selection will depend on the types of SOAP messages that you expect to receive. For example, if incoming SOAP messages will contain an XML Signature within a WS-Security block, you should choose this option from the dropdown.

Using WS-Security Actors:

If the signature is present in a WS-Security block:

1. Select *WS-Security block* from the **Signature Location** dropdown list.
2. Select a SOAP Actor from the **Select Actor/Role(s)** dropdown. Each Actor uniquely identifies a separate WS-Security block. By selecting *Current actor only* from the dropdown, the WS-Security block with no Actor will be taken.
3. In cases where there may be multiple signatures within the WS-Security block, it is necessary to extract one using the **Signature Position** field.

The following is a skeleton version of a message where the XML Signature is contained within the *sample* WS-Security block, i.e. `soap-env:actor="sample"`.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
                  s:actor="sample">
      <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" id="s1">
        ....
      </dsig:Signature>
    </wsse:Security>
  </s:Header>
  <s:Body>
    <ns1:getTime xmlns:ns1="urn:timeservice">
      </ns1:getTime>
    </s:Body>
  </s:Envelope>
```

SOAP Header:

If the signature is present in the SOAP Header:

1. Select *SOAP message header* from the **Signature Location** dropdown list.
2. If there is more than one signature in the SOAP Header, then it is necessary to specify which signature the Enterprise Gateway should use. Specify the appropriate signature by setting the **Signature Position** field.

The following is an example of an XML message where the XML Signature is contained within the SOAP header:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" id="s1">
      ...
    </dsig:Signature>
  </s:Header>
  <s:Body>
    <ns1:getTime xmlns:ns1="urn:timeservice">
      ...
    </ns1:getTime>
  </s:Body>
</s:Envelope>
```

Using XPath:

Finally, an XPath expression can be used to locate the signature.

1. Select *Advanced (XPath)* from the **Signature Location** dropdown list.
2. Select an existing XPath expression from the dropdown, or add a new one by clicking on the **Add** button. XPath expressions can also be edited or removed with the **Edit** and **Remove** buttons respectively.

The default *First Signature* XPath expression takes the first signature from the SOAP Header. The expression is as follows:

<i>XPath Expression:</i>	
//s:Envelope/s:Header/dsig:Signature[1]	

To edit this expression, click the **Edit** button to display the **Enter XPath Expression** dialog.

An example of a SOAP message containing an XML Signature in the SOAP header is provided below. The following XPath expression instructs the Enterprise Gateway to extract the first signature from the SOAP header:

XPath Expression:	
//s:Envelope/s:Header/dsig:Signature[1]	

Because the elements referenced in the expression (`Envelope` and `Signature`) are *prefixed* elements, you must define the namespace mappings for each of these elements as follows:

Prefix	URI
s	http://schemas.xmlsoap.org/soap/envelope/
dsig	http://www.w3.org/2000/09/xmldsig#

```
<?xml version="1.0" encoding="UTF-8"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" id="s1">
      ....
    </dsig:Signature>
  </s:Header>
  <s:Body>
    <product xmlns="http://www.oracle.com">
      <name>SOA Product*</name>
      <company>Company</company>
      <description>Web Services Security</description>
    </product>
  </s:Body>
</s:Envelope>
```

When adding your own XPath expressions, you must be careful to define any namespace mappings in a manner similar to that outlined above. This avoids any potential clashes that might occur where elements of the same name, but belonging to different namespaces are present in an XML message.

What To Sign

Overview

The **What To Sign** section enables the administrator to define the exact content that must be signed for a SOAP message to pass the corresponding filter. The purpose of this configuration section is to ensure that the client has signed something meaningful (part of the SOAP message) instead of some arbitrary data that would pass a blind signature validation.

This prevents clients from simply pasting technically correct, but unrelated signatures into messages in the hope that they pass any blind signature verification. For example, the user may be able to generate a valid XML Signature over any arbitrary XML document. Then by including the signature and XML portion into a malicious SOAP message, the signature passes a blind signature validation, and the harmful XML is allowed to reach the Web Service.

The **What To Sign** section ensures that clients must sign a part of the SOAP message, and therefore prevents them from pasting arbitrary XML Signatures into the message. This section enables you to use any combination of **Node Locations**, **XPath Expressions**, **XPath Predicates**, and/or **Message Attribute** to specify message content that must be signed. This topic describes how to configure each of the corresponding tabs displayed in this section.

ID Configuration

With WSU IDs, an ID attribute is inserted into the root element of the nodeset that is to be signed. The XML Signature then references this ID to indicate to verifiers of the signature the nodes that were signed. The use of WSU IDs is the default option because they are WS-I compliant.

Alternatively, a generic ID attribute (that is not bound to the WSU namespace) can be used to dereference the data. The ID attribute is inserted into the top-level element of the nodeset to be signed. The generated XML Signature can then reference this ID to indicate what nodes were signed.

You can also use `AssertionID` attributes when signing SAML assertions. The following options provide more details and examples of the different styles of IDs that are available.

Use WSU IDs:

Select this option to reference the signed data using a `wsu:Id` attribute. In this case, a `wsu:Id` attribute is inserted into the root node of the nodeset that is signed. This id is then referenced in the generated XML Signature as an indication of what nodes were signed. The following example shows the correlation:

```
<s:Envelope xmlns:s="...">
  <s:Header>
    <wsse:Security xmlns:wsse="...">
      <dsig:Signature xmlns:dsig="..." Id="Id-00000112e2c98df8-0000000000000004">
        <dsig:SignedInfo>
          <dsig:CanonicalizationMethod
            Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <dsig:SignatureMethod
            Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
          <dsig:Reference URI="#Id-00000112e2c98df8-0000000000000003">
            <dsig:Transforms>
              <dsig:Transform
                Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
            </dsig:Transforms>
            <dsig:DigestMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
            <dsig:DigestValue>xChPoiWJJrrPZkbXN8FPB8S4U7w=</dsig:DigestValue>
          </dsig:Reference>
        </dsig:SignedInfo>
      </dsig:Signature>
    </wsse:Security>
  </s:Header>
  <s:Body>
    ...
  </s:Body>
</s:Envelope>
```

```

<dsig:SignatureValue>KG4N .... /9dw==</dsig:SignatureValue>
<dsig:KeyInfo Id="Id-00000112e2c98df8-0000000000000005">
  <dsig:X509Data>
    <dsig:X509Certificate>
      MIID ... ZiBQ==
    </dsig:X509Certificate>
  </dsig:X509Data>
</dsig:KeyInfo>
</dsig:Signature>
</wsse:Security>
</s:Header>
<s:Body xmlns:wsu="..." wsu:Id="Id-00000112e2c98df8-0000000000000003">
<vs:getProductInfo xmlns:vs="http://www.oracle.com">
  <vs:Name>Enterprise Gateway</vs:Name>
  <vs:Version>11.1.1.5.0</vs:Version>
</vs:getProductInfo>
</s:Body>
</s:Envelope>

```

In the above example, a `wsu:Id` attribute has been inserted into the `<s:Body>` element. This `wsu:Id` attribute is then referenced by the `URI` attribute of the `<dsig:Reference>` element in the actual Signature.

When the Signature is being verified, the value of the `URI` attribute can be used to locate the nodes that have been signed.

Use IDs:

Select this option to use generic IDs (that are not bound to the WSU namespace) to dereference the signed data. Under this schema, the `URI` attribute of the `<Reference>` points at an `ID` attribute, which is inserted into the top-level node of the nodeset that is signed. Take a look at the following example, noting how the `ID` specified in the Signature matches the `ID` attribute that has been inserted into the `<Body>` element, indicating that the Signature applies to the entire contents of the SOAP Body.

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
      Id="Id-0000011a101b167c-00000000000000013">
      <dsig:SignedInfo>
        <dsig:CanonicalizationMethod
          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
        <dsig:SignatureMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
        <dsig:Reference URI="#Id-0000011a101b167c-00000000000000012">
          <dsig:Transforms>
            <dsig:Transform
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          </dsig:Transforms>
          <dsig:DigestMethod
            Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
          <dsig:DigestValue>JCy0JoyhVZYzmrLrl92nxfrl+zQ=</dsig:DigestValue>
        </dsig:Reference>
      </dsig:SignedInfo>
      <dsig:SignatureValue>.....<dsig:SignatureValue>
    </dsig:Signature>
    <dsig:KeyInfo Id="Id-0000011a101b167c-00000000000000014">
      <dsig:X509Data>
        <dsig:X509Certificate>.....</dsig:X509Certificate>
      </dsig:X509Data>
    </dsig:KeyInfo>
  </soap:Header>
  <s:Body>
    <vs:getProductInfo xmlns:vs="http://www.oracle.com">
      <vs:Name>Enterprise Gateway</vs:Name>
      <vs:Version>11.1.1.5.0</vs:Version>
    </vs:getProductInfo>
  </s:Body>
</soap:Envelope>

```



```

        </dsig:X509Data>
      </dsig:KeyInfo>
    </dsig:Signature>
  </soap:Header>
<soap:Body Id="Id-0000011a101b167c-00000000000000012">
  <product version="11.1.1.5.0">
    <name>Enterprise Gateway</name>
    <company>Oracle</company>
    <description>SOA Security and Management</description>
  </product>
</soap:Body>
</soap:Envelope>

```

Use SAML IDs for SAML Elements:

This ID option is specifically intended for use where a SAML assertion is to be signed. When this option is selected, an `AssertionID` attribute is inserted into a SAML 1.1 assertion, or a more generic ID attribute is used for a SAML 2.0 assertion.

Node Locations

Node Locations are perhaps the simplest way to configure the message content that must be signed. The table on this screen is pre-populated with a number of common SOAP security headers, including the SOAP Body, WS-Security block, SAML assertion, WS-Security UsernameToken and Timestamp, and the WS-Addressing headers. For each of these headers, there are several namespace options available. For example, you can sign both a SOAP 1.1 and/or a SOAP 1.2 block by distinguishing between their namespaces.

On the **Node Locations** tab, you can select one or more nodesets to sign from the default list. You can also add more default nodesets by clicking the **Add** button. Enter the **Element Name**, **Namespace**, and **Index** of the nodeset in the fields provided. The **Index** field is used to distinguish between two elements of the same name that occur in the same message.

XPath Configuration

You can use an XPath expression to identify the nodeset (the series of elements) that must be signed. To specify that nodeset, select an existing XPath expression from the table, which contains several XPath expressions that can be used to locate nodesets representing common SOAP security headers, including SAML assertions. Alternatively, you can add a new XPath expression using the **Add** button. XPath expressions can also be edited and removed with the **Edit** and **Remove** buttons.

An example of a SOAP message is provided below. The following XPath expression indicates that all the contents of the SOAP body, including the `Body` element itself, should be signed:

```
/soap:Envelope/soap:Body/descendant-or-self::node()
```

You must also supply the namespace mapping for the `soap` prefix, for example:

Prefix	URI
soap	http://schemas.xmlsoap.org/soap/envelope/

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <product xmlns="http://www.oracle.com">
      <name>SOA Product</name>
      <company>Company</company>
      <description>Web Services Security</description>
    </product>
  </soap:Body>
</soap:Envelope>
```

XPath Predicates

Select this option if you wish to use an XPath transform to reference the signed content. You must select an **XPath Predicate** from the table to do this. The table is pre-populated with several XPath predicates that can be used to identify common security headers that occur in SOAP messages, including SAML assertions.

To illustrate the use of XPath predicates, the following example shows how the SOAP message is signed when the default *Sign SOAP Body* predicate is selected:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <vs:getProductInfo xmlns:vs="http://ww.oracle.com">
      <vs:Name>Enterprise Gateway</vs:Name>
      <vs:Version>11.1.1.5.0</vs:Version>
    </vs:getProductInfo>
  </s:Body>
</s:Envelope>
```

The default XPath expression (Sign SOAP Body) identifies the contents of the SOAP Body element, including the Body element itself. The following is the XML Signature produced when this XPath predicate is used:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <dsig:Signature id="Sample" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
      <dsig:SignedInfo>
```

```

...
<dsig:Reference URI="">
  <dsig:Transforms>
    <dsig:Transform
      Algorithm="http://www.w3.org/TR/1999/REC-xpath-19991116">
      <dsig:XPath xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
        ancestor-or-self::soap:Body
      </dsig:XPath>
    </dsig:Transform>
    <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n"/>
  </dsig:Transforms>
  ...
</dsig:Reference>
</dsig:SignedInfo>
...
</dsig:Signature>
</s:Header>
<s:Body>
  <vs:getProductInfo xmlns:vs="http://ww.oracle.com">
    <vs:Name>Enterprise Gateway</vs:Name>
    <vs:Version>11.1.1.5.0</vs:Version>
  </vs:getProductInfo>
</s:Body>
</s:Envelope>

```

This XML Signature includes an extra `Transform` element, which has a child `XPath` element. This element specifies the XPath predicate that validating applications must use to identify the signed content.

Message Attribute

Finally, you can use the contents of a message attribute to determine what must be signed in the message. For example, you can configure a [Locate Nodes](#) filter to extract certain content from the message and store it in a particular message attribute. You can then specify this message attribute on the **Message Attribute** tab.

To do this, select the **Extract nodes from message attribute** checkbox, and enter the name of the attribute that contains the nodes in the field provided.

Configuring XPath Expressions

Overview

The Enterprise Gateway uses XPath expressions in a number of ways, for example, to locate an XML Signature in a SOAP message, to determine what elements of an XML message to validate against an XML Schema, to check the content of a particular element within an XML message, amongst many more uses.

There are two ways to configure XPath expressions on this screen:

- [Manual Configuration](#)
- [XPath Wizard](#)

Manual Configuration

If you are already familiar with XPath and wish to configure the expression manually, complete the following fields, using the examples below if necessary:

1. Enter or select a name for the XPath expression in the **Name** dropdown.
2. Enter the XPath expression to use in the **XPath Expression** field.
3. In order to resolve any prefixes within the XPath expression, the namespace mappings (i.e. **Prefix**, **URI**) should be entered in the table.

Let's take a look at an example. Consider the following SOAP message:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" id="sample">
      .....
      .....
      .....
      .....
    </dsig:Signature>
  </soap:Header>
  <soap:Body>
    <prod:product xmlns:prod="http://www.oracle.com">
      <prod:name>SOA Product*</prod:name>
      <prod:company>Company</prod:company>
      <prod:description>WebServices Security</prod:description>
    </prod:product>
  </soap:Body>
</soap:Envelope>
```

The following XPath expression evaluates to true if the *<name>* element contains the value *Enterprise Gateway*:
XPath Expression: `//prod:name[text()='Enterprise Gateway']`

In this case, it is necessary to define a mapping for the *prod* namespace as follows:

Prefix	URI
prod	http://www.oracle.com

Let's look at another example. This time the element that is to be examined belongs to a default namespace. Consider the following SOAP message:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" id="sample">
      .....
    </dsig:Signature>
  </soap:Header>
  <soap:Body>
    <product xmlns="http://www.company.com">
      <name>SOA Product</name>
      <company>Company</company>
      <description>WebServices Security</description>
    </product>
  </soap:Body>
</soap:Envelope>
```

The following XPath expression evaluates to true if the `<company>` element contains the value *Company*:

XPath Expression: `//ns:company[text()='Company']`

Since the `<company>` element actually belongs to the default (xmlns) namespace, i.e. `http://www.company.com`, it is necessary to make up an arbitrary prefix, *ns*, for use in the XPath expression and assign it to `http://www.company.com`. This is necessary to distinguish between potentially several default namespaces which may exist throughout the XML message. The following mapping illustrates this:

Prefix	URI
ns	http://www.oracle.com

Returning a NodeSet:

Both of the examples above dealt with cases where the XPath expression evaluated to a Boolean value. For example, the expression in the above example asks, "Does the `<company>` element in the `http://www.oracle.com` namespace contain a text node with the value 'oracle'?"

It is sometimes necessary to use the XPath expression to return a subset of the XML message. For example, when using an XPath expression to determine what nodes should be signed in a signed XML message, or when retrieving the node-set to validate against an XML Schema.

The Enterprise Gateway ships with such an XPath expression: one that returns "All Elements inside SOAP Body". To view this expression, select it from the **Name** field. It appears as follows:

XPath Expression: /soap:Envelope/soap:Body/*

This XPath expression simply returns all child elements of the SOAP <Body> element. To construct and test more complicated expressions, administrators are advised to use the **XPath Wizard**.

XPath Wizard

The **XPath Wizard** assists administrators in creating correct and accurate XPath expressions. The wizard allows administrators to load an XML message and then run an XPath expression on it to determine what nodes are returned. To launch the **XPath Wizard**, click on the **XPath Wizard Button** on the **XPath Expression** dialog.

To use the XPath Wizard, simply enter (or browse to) the location of an XML file in the **File** field. The contents of the XML file will appear in the main window of the wizard. Enter an XPath expression in the **XPath** field and click the **Evaluate** button to run the XPath against the contents of the file. If the XPath expression returns any elements (or returns true), those elements will be highlighted in the main window.

If you are not sure how to write the XPath expression yourself, you can simply select an element in the main window. An XPath expression to isolate this element is automatically generated and displayed in the **Selected** field. If you wish to use this expression, select the **Use this path** button, and click **OK**.

MIME/DIME Settings

Overview

The **MIME/DIME Settings** dialog lists a number of default common content types that are used when transmitting MIME messages. The Enterprise Gateway's **Content Type** filter can be configured to accept or block messages containing specific MIME types. Therefore, the contents of the MIME types library acts as the set of all MIME types that the Enterprise Gateway can filter messages with.

All of the MIME types listed in the table are available for selection in the **Content Type** filter. For example, you can configure this filter to only accept XML-based types, such as `application/xml`, `application/*+xml`, `text/xml`, and so on. Similarly, you can block certain MIME types (for example, `application/zip`, `application/octet-stream`, and `video/mppeg`). For more details on configuring this filter, see the [Content Type](#) filter topic.

Configuration

To configure the **MIME/DIME Settings**, in the Policy Studio main menu, select **Settings -> Settings -> MIME/DIME**. This displays the **MIME/DIME Settings** dialog. Alternatively, in the toolbar, click the drop-down option on the **Settings** button, and select **MIME/DIME**.

The **MIME/DIME Settings** dialog lists the actual MIME types on the left column of the table, together with their corresponding file extensions (where applicable) on the right column.

To add a new MIME type, click the **Add** button. In the **Configure MIME/DIME Type** dialog, enter the new content type in the **MIME or DIME Type** field. If the new type has a corresponding file extension, enter this extension in the **Extension** field. Click the **OK** button when finished.

Similarly, existing types can be edited and removed by clicking on the **Edit** and **Delete** buttons, respectively.

Configuring Connection Groups

Overview

A **Connection Group** consists of a number of external servers that the Enterprise Gateway connects to (for example, RSA Access Manager servers for authorization). The Enterprise Gateway attempts to connect to all the servers in the group in a round-robin fashion, therefore providing a high degree of failover. If one or more servers are unavailable, the Enterprise Gateway can still connect to an alternative server.

The Enterprise Gateway attempts to connect to the listed servers according to the priorities assigned to them. For example, assume there are two High priority servers, one Medium priority server, and one Low priority server configured. Assuming the Enterprise Gateway can successfully connect to the two High priority servers, it alternates requests between these two servers only in a round-robin fashion. The other group servers are not used. However, if both High priority servers become unavailable, the Enterprise Gateway then tries to use the Medium priority server, and only if this fails is the Low priority server used.

Connection Groups are available in Policy Studio on the **External Connections** tab according to the filter from which they are available. For example, Connection Sets under the **RSA ClearTrust Connection Sets** node are available in the RSA Access Manager filter. For more details, see the [RSA Access Manager Authorization](#) topic.

Configuring a Connection Group

You can configure a Connection Group using the **Connection Group** dialog. The external servers are listed in order of priority in the table on the **Connection Group** dialog. The Enterprise Gateway attempts to connect to the server at the top of the list first. If this server is not available, a connection attempt is made to the second server, and so on until an available server is contacted. If none of the listed servers are available, the client is not authorized and a SOAP fault is returned to the client.

You can increase or decrease the priorities of the listed external servers using the **Up** and **Down** buttons. You can add, edit, and delete Access Manager servers using the **Add**, **Edit**, and **Remove** buttons.

Configuring a Connection

You can configure a single connection using the **Connection Configuration** dialog. To configure a single **Access Manager Connection**, perform the following steps:

1. Enter the name or IP address of the machine hosting the selected Access Manager server in the **Location** field.
2. Enter the **Port** on which the specified Access Manager server is listening.
3. Select a suitable **Timeout** for connections to this server.
4. Select the appropriate **Connection Type** for the Enterprise Gateway to use when connecting to the specified Access Manager server. Connections between the Enterprise Gateway and the Access Manager server can be made in the clear, over Anonymous SSL, or Mutual SSL Authentication (two-way SSL).
5. If **SSL Authentication** is selected, you must select an **SSL mutual authentication certificate**. This certificate is then used to authenticate to the Access Manager server.

Configuring URL Groups

Overview

The Enterprise Gateway can make connections on a round-robin basis to the URLs listed in a URL group, thus enabling a high degree of failover to external servers (for example, Entrust GetAccess, OCSP, SAML PDP, or XKMS).

The Enterprise Gateway attempts to connect to the listed servers according to the priorities assigned to them. For example, assume there are two High priority URLs, one Medium URL, and one Low URL configured. Assuming the Enterprise Gateway can successfully connect to the two High priority URLs, it alternates requests between these two URLs only in a round-robin fashion. The other group URLs are not used. However, if both of the High priority URLs become unavailable, the Enterprise Gateway then tries to use the Medium priority URL, and only if this fails is the Low priority URL used.

In general, the Enterprise Gateway attempts to round-robin requests over URLs of the same priority, but uses higher priority URLs before lower priority ones. When a new URL is added to the group, it is automatically given the highest priority. You can then change priorities by selecting the URL, and clicking the **Up** and **Down** buttons.

You can add and edit URLs by selecting the URL from the table, and clicking on the **Add** and **Edit** buttons.

Configuration

Configure the following fields in the **URL Configuration** dialog:

- **URL:**
Enter the full URL of the external server.
- **Timeout:**
Specify the timeout in seconds for connections to the specified server.
- **Retry After:**
Whenever the server becomes unavailable for whatever reason (for example, maintenance), no attempt is made to connect to that server until the time specified here has elapsed. In other words, when a connection failure is detected, the next connection to that URL is after this amount of time.
- **SSL Certificate:**
If the specified server requires clients to authenticate to it over two-way SSL, you must select an **SSL Certificate** from the Trusted Certificate Store for authentication.
- **Host/IP:**
If the specified server sits behind a proxy server, you must enter the host name or IP address of the proxy server.
- **Port:**
Enter the port on which the proxy is listening.

LDAP User Search

Configure Directory Search

The **User Search** dialog is used to search a given LDAP directory for a unique user according to the criteria configured in the fields on this dialog.

Base Criteria:

The value entered here tells the Enterprise Gateway where it should begin searching the LDAP directory. For example, it may be appropriate to search for a given user under the "C=IE" tree in the LDAP hierarchy.

Query Search Filter:

The value entered here is what the Enterprise Gateway will use to determine whether it has obtained a successful match or not. In this case, since we are searching for a specific user, we can use the username of an authenticated user (i.e. the value of the `authentication.subject.id` message attribute to lookup in the LDAP directory. We must also specify the object class that defines users for the particular type of LDAP directory that we are searching against. For example, object classes representing users amongst common LDAP directories are "inetOrgPerson", "givenName", and "User".

So, for example, to search for an authenticated user against Microsoft's Active Directory, you might specify the following as the **Query Search Filter**:

```
(objectclass=User)(cn=${authentication.subject.id})
```

Search Scope:

The checkboxes here indicate the depth of the LDAP tree that you wish to search. The choice selected here will depend largely on the structure of your LDAP directory.

Configuring a Transparent Proxy

Overview

On Linux systems with the `TProxy` kernel option enabled, you can configure the Enterprise Gateway as a *transparent proxy*. This enables the Enterprise Gateway to present itself as having the server's IP address from the point of view of the client, and/or having the client's IP address from the point of view of the server. This can be useful for administrative or network infrastructure purposes (for example, to keep using existing client/server IP addresses, and for load-balancing).

You can configure transparent proxy mode both for inbound and outbound Enterprise Gateway connections:

- Incoming interfaces can listen on IP addresses that are not assigned to any interface on the local host.
- Outbound calls can present the originating client's IP address to the destination server.

Both of these options act independently of each other.

Configuring Transparent Proxy Mode for Incoming Interfaces

To enable transparent proxy mode on an incoming interface, perform the following steps:

1. In the Policy Studio tree view, expand the **Processes -> Oracle Enterprise Gateway** nodes.
2. Right-click your service, and select **Add Interface -> HTTP** or **HTTPS** to display the appropriate dialog (for example, **Configure HTTP Interface**).
3. Select the checkbox labeled **Transparent Proxy (allow bind to foreign address)**. When selected, the value in the **Address** field can specify any IP address, and incoming traffic for the configured address/port combinations is handled by the Enterprise Gateway.

For more details on configuring interfaces, see [Configuring HTTP Services](#).

Configuring Transparent Proxy Mode for Outgoing Calls

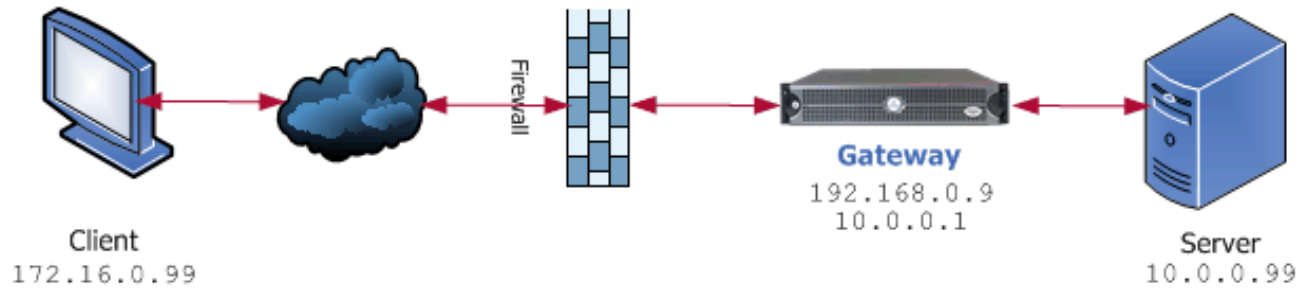
Transparent proxy mode for outgoing calls must be enabled at the level of a connection filter in a circuit. To enable transparent proxy mode for outbound calls, perform the following steps:

1. Ensure that your circuit contains a connection filter (for example, **Connect to URL** or **Connection**, available from the **Routing** category in the filter palette).
2. In your connection filter, select the **Advanced** tab.
3. Select the checkbox labeled **Transparent Proxy (present client's IP address to server)**. When selected, the IP address of the original client connection that caused the circuit to be invoked is used as the local address of the TCP connection to the destination server.

For more details on configuring connection filters, see [Connection](#) and [Connect to URL](#).

Configuration Example

A typical configuration example of transparent proxy mode is shown as follows:



In this example, the remote client's address is `172.16.0.99`, and it is attempting to connect to the server at `10.0.0.99`, port 80. The front-facing firewall is configured to route traffic for `10.0.0.99` through the Enterprise Gateway at address `192.168.0.9`. The server is configured to use the Enterprise Gateway at address `10.0.0.1` as its default IP router.

The Enterprise Gateway is multi-homed, and sits on both the `192.168.0.0/24` and `10.0.0.0/24` networks. It is configured with a listening interface at address `10.0.0.99:80`, with transparent proxy mode switched on, as shown in the following **Configure HTTP Interface** dialog:

The screenshot shows the **Configure HTTP Interface** dialog box. It contains the following settings:

- Port:** 80
- Address:** 10.0.0.99
- Protocol:** IPv4
- Backlog:** 64
- Idle Timeout:** 60000
- Active Timeout:** 60000
- Maximum Memory Per Request:** 16777216
- Trace level:** From System Settings
- ☐ Allow address reuse (use SO_REUSEADDR socket option)
- ☒ Transparent Proxy - allow bind to foreign address

At the bottom of the dialog are three buttons: **OK**, **Cancel**, and **Help**.

The Enterprise Gateway accepts the incoming call from the client, and processes it locally. However, there is no communication with the server yet. The Enterprise Gateway can process the call to completion and respond to the client—it is masquerading as the server.

If the Enterprise Gateway invokes a connection filter when processing this call (with transparent proxying enabled), the

connection filter consults the originating address of the client, and binds the local address of the new outbound connection to that address before connecting. The server then sees the incoming call on the Enterprise Gateway originating from the client (172.16.0.99), rather than either of the Enterprise Gateway's IP addresses. The following dialog shows the example configuration for the **Connect to URL** filter:

Configure a new 'Connect to URL' filter

Connect to URL
Route message to the specified URL

Name:

URL:

Trusted Certificates Client SSL Authentication HTTP Authentication Kerberos Authentication **Advanced**

Advanced Settings

☐ Send via proxy

☒ Transparent Proxy (present client's IP address to server)

Ciphers:

HTTP Host Header

☐ Use Host header specified by client ☒ Generate new Host header

The result is a transparent proxy, where the client sees itself as connecting directly to the server, and the server sees an incoming call directly from the client. The Enterprise Gateway processes two separate TCP connections, one to the client, one to the server, with both masquerading as the other on each connection.

Note: Either side of the transparent proxy is optional. By configuring the appropriate settings for the incoming interface or the connection filter, you can masquerade only to the server, or only to the client.

Message Attribute Reference

attribute.lookup.list

<i>Name</i>	attribute.lookup.list
<i>Description</i>	User attributes can be retrieved from a variety of sources, including LDAP directories, databases, Oracle Entity Store, SAML attribute assertions, and so on. All retrieved attributes are stored in the attribute.lookup.list attribute, where they can be looked up at a later stage in the circuit.
<i>Type</i>	java.util.HashMap
<i>Generated By</i>	Extract Certificate Attributes Retrieve Attributes from Database Retrieve Attributes from Directory Server Retrieve Attributes from SAML Attribute Assertion Retrieve Attributes from SAML PDP Retrieve Attributes from User Store SiteMinder Authorization
<i>Consumed By</i>	
<i>Required By</i>	Attribute Authorization Insert SAML Attribute Assertion

attribute.subject.format

<i>Name</i>	attribute.subject.format
<i>Description</i>	The format of the subject ID that is used to lookup attributes (for example, X.509 DName or username).
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes Retrieve Attributes from Directory Server Retrieve Attributes from SAML Attribute Assertion SiteMinder Authorization
<i>Consumed By</i>	
<i>Required By</i>	Insert SAML Attribute Assertion

attribute.subject.id

<i>Name</i>	attribute.subject.id
<i>Description</i>	The ID of the subject that is used to look up user attributes. This can either be an X.509 Distinguished Name (DName) or a username.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes Retrieve Attributes from Directory Server Retrieve Attributes from SAML Attribute Assertion SiteMinder Authorization

<i>Consumed By</i>	
<i>Required By</i>	Insert SAML Attribute Assertion

authentication.cert

<i>Name</i>	authentication.cert
<i>Description</i>	The certificate that was used to authenticate the client.
<i>Type</i>	java.security.cert.X509Certificate
<i>Generated By</i>	HTTP Header Authentication
<i>Consumed By</i>	
<i>Required By</i>	

authentication.issuer.format

<i>Name</i>	authentication.issuer.format
<i>Description</i>	The format of the authentication.issuer.id attribute.
<i>Type</i>	java.lang.String
<i>Generated By</i>	HTTP Header Authentication SSL Authentication
<i>Consumed By</i>	
<i>Required By</i>	

authentication.issuer.id

<i>Name</i>	authentication.issuer.id
<i>Description</i>	Contains the ID of the issuer of the authenticated client's certificate. This is usually either the X.509 DName or the username of the issuer of the subject's certificate.
<i>Type</i>	java.lang.String
<i>Generated By</i>	HTTP Header Authentication SSL Authentication SAML Authentication XML-Signature Verification
<i>Consumed By</i>	
<i>Required By</i>	

authentication.issuer.orig.format

<i>Name</i>	authentication.issuer.orig.format
<i>Description</i>	Format of the authentication.issuer.orig.id attribute. This is the format of the issuer's ID before any credential mapping

	was done to the identifier, e.g. DName to username.
<i>Type</i>	<code>java.lang.String</code>
<i>Generated By</i>	HTTP Header Authentication
<i>Consumed By</i>	
<i>Required By</i>	

authentication.issuer.orig.id

<i>Name</i>	<code>authentication.issuer.orig.id</code>
<i>Description</i>	The ID of the issuer of the subject's credential before any credential mapping took place. An example of credential mapping would involve mapping an issuer's username to a DName.
<i>Type</i>	<code>java.lang.String</code>
<i>Generated By</i>	HTTP Header Authentication
<i>Consumed By</i>	
<i>Required By</i>	

authentication.method

<i>Name</i>	<code>authentication.method</code>
<i>Description</i>	The method used by the client to authenticate to Enterprise Gateway.
<i>Type</i>	<code>java.lang.String</code>
<i>Generated By</i>	HTTP Basic Authentication HTTP Digest Authentication HTTP Header Authentication SAML Authentication SSL Authentication SiteMinder Certificate Authentication SiteMinder Session Validation XML-Signature Authentication
<i>Consumed By</i>	
<i>Required By</i>	Insert SAML Authentication Assertion Retrieve Attributes from SAML PDP SAML PDP Authorization

authentication.subject.format

<i>Name</i>	<code>authentication.subject.format</code>
<i>Description</i>	The format of the subject's ID, e.g. X.509 DName or user-name.
<i>Type</i>	<code>java.lang.String</code>

<i>Generated By</i>	HTTP Basic Authentication HTTP Digest Authentication HTTP Header Authentication Retrieve Attributes from SAML Attribute Assertion Retrieve Attributes from SAML PDP Retrieve Attributes from Database Retrieve Attributes from Directory Server Retrieve Attributes from User Store SAML Authentication SSL Authentication SiteMinder Certificate Authentication SiteMinder Session Validation XML-Signature Authentication
<i>Consumed By</i>	
<i>Required By</i>	Retrieve Attributes from Database Retrieve Attributes from Directory Server Retrieve Attributes from SAML Attribute Assertion Retrieve Attributes from SAML PDP Retrieve Attributes from User Store SAML PDP Authorization SAML Authorization Tivoli Authorization

authentication.subject.id

<i>Name</i>	<code>authentication.subject.id</code>
<i>Description</i>	Contains the ID of the authenticated subject (for example, the username supplied by the client).
<i>Type</i>	<code>java.lang.String</code>
<i>Generated By</i>	HTTP Basic Authentication HTTP Digest Authentication HTTP Header Authentication Retrieve Attributes from SAML Attribute Assertion Retrieve Attributes from SAML PDP Retrieve Attributes from Database Retrieve Attributes from Directory Server Retrieve Attributes from User Store SAML Authentication SSL Authentication SiteMinder Certificate Authentication SiteMinder Session Validation XML-Signature Authentication
<i>Consumed By</i>	
<i>Required By</i>	Retrieve Attributes from Database Retrieve Attributes from Directory Server Retrieve Attributes from SAML Attribute Assertion Retrieve Attributes from SAML PDP Retrieve Attributes from User Store

authentication.subject.orig.format

<i>Name</i>	authentication.subject.orig.format
<i>Description</i>	The ID of the subject before any credential mapping was performed. For example, the subject's ID may be mapped from an X.509 DName to the username stored in an external database. In this case, the subject's original ID was a DName.
<i>Type</i>	java.lang.String
<i>Generated By</i>	HTTP Basic Authentication HTTP Digest Authentication HTTP Header Authentication
<i>Consumed By</i>	
<i>Required By</i>	

authentication.subject.orig.id

<i>Name</i>	authentication.subject.orig.id
<i>Description</i>	Contains the ID of the authenticated subject before credential mapping is performed (for example, from username to DName).
<i>Type</i>	java.lang.String
<i>Generated By</i>	HTTP Basic Authentication HTTP Digest Authentication HTTP Header Authentication
<i>Consumed By</i>	
<i>Required By</i>	

authentication.subject.password

<i>Name</i>	authentication.subject.password
<i>Description</i>	If a user authenticates to the Enterprise Gateway using a username and password combination (either with HTTP digest/basic authentication or with a WS-Security Username token), the user's password is stored in this attribute.
<i>Type</i>	java.lang.String
<i>Generated By</i>	HTTP Basic Authentication HTTP Digest Authentication
<i>Consumed By</i>	
<i>Required By</i>	

authentication.ws.wsblockinfo

<i>Name</i>	authentication.ws.wsblockinfo
-------------	-------------------------------

<i>Description</i>	Contains a WS-Security <Header> block that has been extracted from a message.
<i>Type</i>	<code>java.lang.String</code>
<i>Generated By</i>	Extract WSS Header
<i>Consumed By</i>	
<i>Required By</i>	

cert.basic.constraints

<i>Name</i>	<code>cert.basic.constraints</code>
<i>Description</i>	If the subject is a Certificate Authority (CA), and the BasicConstraints extension exists, this attribute gives the maximum number of CA certificates that may follow this certificate in a certification path. A value of zero indicates that only an end-entity certificate may follow in the path. This contains the value of pathLenConstraint if the BasicConstraints extension is present in the certificate and the subject of the certificate is a CA, otherwise its value is -1. If the subject of the certificate is a CA and pathLenConstraint does not appear, there is no limit to the allowed length of the certification path.
<i>Type</i>	<code>java.lang.Integer</code>
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.extended.key.usage

<i>Name</i>	<code>cert.extended.key.usage</code>
<i>Description</i>	A string representing the OBJECT IDENTIFIERS of the ExtKeyUsageSyntax field of the extended key usage extension (OID = 2.5.29.37). It indicates a purpose for which the certified public key may be used, in addition to, or instead of, the basic purposes indicated in the key usage extension field.
<i>Type</i>	<code>java.lang.String</code>
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.hash.md5

<i>Name</i>	<code>cert.hash.md5</code>
<i>Description</i>	An MD5 hash of the certificate.

Type	java.lang.String
Generated By	Extract Certificate Attributes
Consumed By	
Required By	

cert.hash.sha1

Name	cert.hash.sha1
Description	An SHA1 hash of the certificate.
Type	java.lang.String
Generated By	Extract Certificate Attributes
Consumed By	
Required By	

cert.issuer.alternative.name

Name	cert.issuer.alternative.name
Description	An alternative name for the certificate issuer from the <code>IssuerAltName</code> extension (OID = 2.5.29.18)
Type	java.lang.String
Generated By	Extract Certificate Attributes
Consumed By	
Required By	

cert.issuer.id

Name	cert.issuer.id
Description	The Distinguished Name (DName) of the issuer of the certificate.
Type	java.lang.String
Generated By	Extract Certificate Attributes
Consumed By	
Required By	

cert.issuer.id.c

Name	cert.issuer.id.c
Description	The <code>c</code> attribute of the issuer of the certificate, if it exists.
Type	java.lang.String

<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.issuer.id.cn

<i>Name</i>	cert.issuer.id.cn
<i>Description</i>	The cn attribute of the issuer of the certificate, if it exists.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.issuer.id.emailaddress

<i>Name</i>	cert.issuer.id.emailaddress
<i>Description</i>	The email or emailaddress attribute of the issuer of the certificate, if it exists.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.issuer.id.l

<i>Name</i>	cert.issuer.id.l
<i>Description</i>	The l attribute of the issuer of the certificate, if it exists.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.issuer.id.o

<i>Name</i>	cert.issuer.id.o
<i>Description</i>	The o attribute of the issuer of the certificate, if it exists.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	

<i>Required By</i>	
--------------------	--

cert.issuer.id.ou

<i>Name</i>	cert.issuer.id.ou
<i>Description</i>	The ou attribute of the issuer of the certificate, if it exists.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.issuer.id.st

<i>Name</i>	cert.issuer.id.st
<i>Description</i>	The st attribute of the issuer of the certificate, if it exists.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.key.usage.cRLSign

<i>Name</i>	cert.key.usage.cRLSign
<i>Description</i>	Set to true or false if the key can be used for cRLSign .
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.key.usage.dataEncipherment

<i>Name</i>	cert.key.usage.dataEncipherment
<i>Description</i>	Set to true or false if the key can be used for dataEncipherment .
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.key.usage.decipherOnly

<i>Name</i>	cert.key.usage.decipherOnly
<i>Description</i>	Set to true or false if the key can be used for decipherOnly.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.key.usage.digitalSignature

<i>Name</i>	cert.key.usage.digitalSignature
<i>Description</i>	Set to true or false if the key can be used for digital signature.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.key.usage.encipherOnly

<i>Name</i>	cert.key.usage.encipherOnly
<i>Description</i>	Set to true or false if the key can be used for encipherOnly.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.key.usage.keyAgreement

<i>Name</i>	cert.key.usage.keyAgreement
<i>Description</i>	Set to true or false if the key can be used for key-Agreement.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.key.usage.keyCertSign

<i>Name</i>	cert.key.usage.keyCertSign
<i>Description</i>	Set to true or false if the key can be used for keyCertSign.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.key.usage.keyEncipherment

<i>Name</i>	cert.key.usage.keyEncipherment
<i>Description</i>	Set to true or false if the key can be used for keyEncipherment.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.key.usage.nonRepudiation

<i>Name</i>	cert.key.usage.nonRepudiation
<i>Description</i>	Set to true or false if the key can be used for non-repudiation.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.not.after

<i>Name</i>	cert.not.after
<i>Description</i>	Not after date for the validity of the certificate.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.not.before

<i>Name</i>	<code>cert.not.before</code>
<i>Description</i>	Not before date for the validity of the certificate.
<i>Type</i>	<code>java.lang.String</code>
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.serial.number

<i>Name</i>	<code>cert.serial.number</code>
<i>Description</i>	Certificate serial number.
<i>Type</i>	<code>java.lang.String</code>
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.signature.algorithm

<i>Name</i>	<code>cert.signature.algorithm</code>
<i>Description</i>	The signature algorithm for the certificate signature.
<i>Type</i>	<code>java.lang.String</code>
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.subject.alternative.name

<i>Name</i>	<code>cert.subject.alternative.name</code>
<i>Description</i>	An alternative name for the subject from the SubjectAltName extension (OID = 2.5.29.17).
<i>Type</i>	<code>java.lang.String</code>
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.subject.id

<i>Name</i>	<code>cert.subject.id</code>
<i>Description</i>	The Distinguished Name (DName) of the subject of the cer-

	certificate.
Type	java.lang.String
Generated By	Extract Certificate Attributes
Consumed By	
Required By	

cert.subject.id.c

Name	cert.subject.id.c
Description	The c attribute of the subject of the certificate, if it exists.
Type	java.lang.String
Generated By	Extract Certificate Attributes
Consumed By	
Required By	

cert.subject.id.cn

Name	cert.subject.id.cn
Description	The cn attribute of the subject of the certificate, if it exists.
Type	java.lang.String
Generated By	Extract Certificate Attributes
Consumed By	
Required By	

cert.subject.id.emailaddress

Name	cert.subject.id.emailaddress
Description	The emailaddress attribute of the subject of the certificate, if it exists.
Type	java.lang.String
Generated By	Extract Certificate Attributes
Consumed By	
Required By	

cert.subject.id.l

Name	cert.subject.id.l
Description	The l attribute of the subject of the certificate, if it exists.
Type	java.lang.String

<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.subject.id.o

<i>Name</i>	cert.subject.id.o
<i>Description</i>	The o attribute of the subject of the certificate, if it exists.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.subject.id.ou

<i>Name</i>	cert.subject.id.ou
<i>Description</i>	The ou attribute of the subject of the certificate, if it exists.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.subject.id.st

<i>Name</i>	cert.subject.id.st
<i>Description</i>	The st attribute of the subject of the certificate, if it exists.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	
<i>Required By</i>	

cert.version

<i>Name</i>	cert.version
<i>Description</i>	The version of the certificate.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract Certificate Attributes
<i>Consumed By</i>	

<i>Required By</i>	
--------------------	--

certificate

<i>Name</i>	certificate
<i>Description</i>	Used to store a certificate in addition to the certificate stored in the authentication.cert attribute. For example, the Find Certificate filter can extract a certificate from an LDAP directory, HTTP header, or the User Store. By default, it sets this certificate to the certificate attribute.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Find Certificate HTTP Header Authentication Integrity XML-Signature Verification SAML Authentication XML-Signature Verification SAML PDP XML-Signature Response Verification SAML PDP XML-Signature Response Verification SSL Authentication XML-Signature Authentication
<i>Consumed By</i>	
<i>Required By</i>	Create Thumbprint from Certificate Extract Certificate Attributes SiteMinder Certificate Authentication

certificate.thumbprint

<i>Name</i>	certificate.thumbprint
<i>Description</i>	Contains a human-readable thumbprint (or fingerprint), which is generated from the X.509 certificate stored in the certificate message attribute.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Create Thumbprint from Certificate
<i>Consumed By</i>	
<i>Required By</i>	

certificates

<i>Name</i>	certificates
<i>Description</i>	Contains an array of X.509 certificates for use in the Certificate Chain Check and Certificate Validation filters.
<i>Type</i>	java.util.ArrayList
<i>Generated By</i>	Find Certificate HTTP Header Authentication Integrity XML-Signature Verification

	SAML Authentication XML-Signature Verification SAML PDP XML-Signature Response Verification SAML PDP Response XML-Signature Verification SSL Authentication XML-Signature Authentication
<i>Consumed By</i>	
<i>Required By</i>	Certificate Chain File-based CRL Certificate Validation LDAP-based CRL Certificate Validation OCSP Certificate Validation SiteMinder Certificate Authentication XKMS Certificate Validation

circuit.abort.exception

<i>Name</i>	circuit.abort.exception
<i>Description</i>	<p>Whenever a filter throws an exception, the exception is stored in the <code>circuit.abort.exception</code> message attribute. The exception can then be used, for example, to return customized SOAP faults that describe the verbose details of the error.</p> <p>It is important to note that a filter only throws an exception if it cannot carry out its core task. For example, an authentication filter throws an exception if it cannot connect to the authentication repository, a Schema validation filter aborts if the request is not XML, and an attribute retrieval filter aborts if it cannot connect to the user profile store (for example, database, LDAP, and so on).</p>
<i>Type</i>	java.lang.String
<i>Generated By</i>	The <code>circuit.abort.exception</code> can be thrown by all filters.
<i>Consumed By</i>	
<i>Required By</i>	The <code>circuit.abort.exception</code> can be used as a property in appropriate filters. For example, you can specify the <code>\${circuit.abort.exception}</code> property in the Set Message filter, which can then be returned to the client using a Reflect filter.

content.body

<i>Name</i>	content.body
<i>Description</i>	Contains the parsed body of the incoming HTTP request.
<i>Type</i>	com.vordel.mime.Body
<i>Generated By</i>	Load File
<i>Consumed By</i>	
<i>Required By</i>	Attachment Filtering Connection

	Content Validation GetAccess Authorization Insert SAML Attribute Assertion Insert SAML Authentication Assertion Insert SAML Authorization Assertion Integrity XML-Signature Verification Throttling McAfee Virus Scanner Operation Name Resolver Reflect Reflect Message and Attributes Retrieve Attribute from HTTP Header Retrieve Attribute from Message Retrieve Attributes from SAML Attribute Assertion SAML Authorization Save to File Schema Validation Sign Message SiteMinder Session Validation SSL Authentication XML Complexity XML-Decryption XML-Encryption XML-Signature Authentication XSLT Transformation Web Service Filter
--	---

decryption.properties

<i>Name</i>	decryption.properties
<i>Description</i>	Indicates the XML-Encrypted block(s) to decrypt. The actual decryption is performed by the XML-Decryption filter.
<i>Type</i>	java.util.Map
<i>Generated By</i>	XML-Decryption Settings
<i>Consumed By</i>	
<i>Required By</i>	XML-Decryption

encryption.properties

<i>Name</i>	encryption.properties
<i>Description</i>	Allows the user to encrypt (part of) the message for a number of recipients so that only those recipients can to decrypt the encrypted data. The encryption is performed by the XML-Encryption filter.
<i>Type</i>	java.util.Map
<i>Generated By</i>	XML-Encryption Settings
<i>Consumed By</i>	
<i>Required By</i>	XML-Encryption

http.destination.host

<i>Name</i>	http.destination.host
<i>Description</i>	The host on which the destination Web Service is running.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Dynamic Router Static Router Web Service Filter
<i>Consumed By</i>	
<i>Required By</i>	Connection

http.destination.port

<i>Name</i>	http.destination.port
<i>Description</i>	The port on which the destination Web Service is listening.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Dynamic Router Static Router Web Service Filter
<i>Consumed By</i>	
<i>Required By</i>	Connection

http.destination.protocol

<i>Name</i>	http.destination.protocol
<i>Description</i>	Indicates the protocol to use when routing messages to the destination Web Service. Typically, the protocol is either HTTP or HTTPS.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Dynamic Router Static Router Web Service Filter
<i>Consumed By</i>	
<i>Required By</i>	Connection

http.headers

<i>Name</i>	http.headers
<i>Description</i>	Contains a list of all HTTP headers from the incoming request.
<i>Type</i>	com.vordel.mime.HeaderSet
<i>Generated By</i>	Connection Web Service Filter

<i>Consumed By</i>	
<i>Required By</i>	Add HTTP Header Connection HTTP Basic Authentication HTTP Digest Authentication HTTP Header Authentication Reflect Reflect Message and Attributes Remove HTTP Header Return WSDL SOAPAction Validate HTTP Headers Web Service Filter

http.request.clientaddr

<i>Name</i>	http.request.clientaddr
<i>Description</i>	Contains the IP address of the client machine from which the HTTP request was sent to Enterprise Gateway.
<i>Type</i>	java.net.InetSocketAddress
<i>Generated By</i>	
<i>Consumed By</i>	
<i>Required By</i>	IP Address

http.request.connection.error

<i>Name</i>	http.request.connection.error
<i>Description</i>	If an error occurs when Enterprise Gateway is routing on to the target Web Service, the error status will be recorded in this attribute.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Connection Web Service Filter
<i>Consumed By</i>	
<i>Required By</i>	

http.request.clientcert

<i>Name</i>	http.request.clientcert
<i>Description</i>	Contains the certificate used by the client in the HTTP request.
<i>Type</i>	java.security.cert.X509Certificate
<i>Generated By</i>	
<i>Consumed By</i>	

<i>Required By</i>	
--------------------	--

http.request.host

<i>Name</i>	http.request.host
<i>Description</i>	Contains the hostname on which the HTTP request from the client is received by the Enterprise Gateway.
<i>Type</i>	java.lang.String
<i>Generated By</i>	
<i>Consumed By</i>	
<i>Required By</i>	

http.request.port

<i>Name</i>	http.request.port
<i>Description</i>	Contains the port on which the HTTP request from the client is received by the Enterprise Gateway.
<i>Type</i>	java.lang.String
<i>Generated By</i>	
<i>Consumed By</i>	
<i>Required By</i>	

http.request.protocol

<i>Name</i>	http.request.protocol
<i>Description</i>	Contains the protocol used by the client to send the request message to the Enterprise Gateway. Typically, the protocol is either HTTP or HTTPS.
<i>Type</i>	java.lang.String
<i>Generated By</i>	
<i>Consumed By</i>	
<i>Required By</i>	

http.request.uri

<i>Name</i>	http.request.uri
<i>Description</i>	Contains the URI on which the HTTP request is received by the Enterprise Gateway.
<i>Type</i>	java.net.URI
<i>Generated By</i>	Web Service Filter

<i>Consumed By</i>	
<i>Required By</i>	Connection Dynamic Router Operation Name Rewrite URL Relative Path Return WSDL

http.request.verb

<i>Name</i>	http.request.verb
<i>Description</i>	Contains the HTTP verb used in the client HTTP request to the Enterprise Gateway.
<i>Type</i>	java.lang.String
<i>Generated By</i>	
<i>Consumed By</i>	
<i>Required By</i>	Connection HTTP Basic Authentication HTTP Digest Authentication Web Service Filter

http.request.version

<i>Name</i>	http.request.version
<i>Description</i>	Contains the HTTP version used in the request from the client to the Enterprise Gateway.
<i>Type</i>	java.lang.String
<i>Generated By</i>	
<i>Consumed By</i>	
<i>Required By</i>	

http.response.info

<i>Name</i>	http.response.info
<i>Description</i>	Contains the reason-phrase from the HTTP status-line (for example, Not found from the 404 Not found status-line).
<i>Type</i>	java.lang.String
<i>Generated By</i>	Connection Web Service Filter
<i>Consumed By</i>	
<i>Required By</i>	

http.response.status

<i>Name</i>	http.response.status
<i>Description</i>	Stores the HTTP status-code of the response from the Web Service, (for example, 404 from the 404 Not found status-line).
<i>Type</i>	java.lang.Integer
<i>Generated By</i>	Connection Web Service Filter
<i>Consumed By</i>	
<i>Required By</i>	

http.response.version

<i>Name</i>	http.response.version
<i>Description</i>	Stores the HTTP version used in the response from the Web Service.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Connection Web Service Filter
<i>Consumed By</i>	
<i>Required By</i>	

kerberos.client.context.established

<i>Name</i>	kerberos.client.context.established
<i>Description</i>	Indicates whether the client-side context has been established (true or false).
<i>Type</i>	java.lang.String
<i>Generated By</i>	Kerberos Client Authentication
<i>Consumed By</i>	
<i>Required By</i>	

kerberos.context

<i>Name</i>	kerberos.context
<i>Description</i>	Used on client-side to reload the context to consume the service-side token, if there is one.
<i>Type</i>	org.ietf.jgss.GSSContext
<i>Generated By</i>	Kerberos Client Authentication
<i>Consumed By</i>	
<i>Required By</i>	

kerberos.mechanism.oid

<i>Name</i>	kerberos.mechanism.oid
<i>Description</i>	Contains the mechanism OID (SPNEGO or Kerberos).
<i>Type</i>	java.lang.String
<i>Generated By</i>	Kerberos Client Authentication Kerberos Service Authentication
<i>Consumed By</i>	
<i>Required By</i>	

kerberos.profile.ap.req.bst.id

<i>Name</i>	kerberos.profile.ap.req.bst.id
<i>Description</i>	Contains the wsu:Id of the BinarySecurityToken containing the AP_REQ message generated by the GssInitiatorFilter .
<i>Type</i>	java.lang.String
<i>Generated By</i>	Kerberos Client Authentication
<i>Consumed By</i>	
<i>Required By</i>	

kerberos.profile.ap.req.sha1

<i>Name</i>	kerberos.profile.ap.req.sha1
<i>Description</i>	Contains the SHA1 hash of a Kerberos AP_REQ message. This is generated when a Kerberos service consumes it, or when a Kerberos client generates it.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Kerberos Client Authentication Kerberos Service Authentication
<i>Consumed By</i>	
<i>Required By</i>	

kerberos.service.authenticator.principal

<i>Name</i>	kerberos.service.authenticator.principal
<i>Description</i>	Contains the principal extracted from the AP_REQ message on the Kerberos acceptor side.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Kerberos Service Authentication
<i>Consumed By</i>	
<i>Required By</i>	

kerberos.service.authenticator.realm

<i>Name</i>	kerberos.service.authenticator.realm
<i>Description</i>	Contains the realm extracted from the AP_REQ message on the Kerberos acceptor side.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Kerberos Service Authentication
<i>Consumed By</i>	
<i>Required By</i>	

kerberos.service.authenticator.time

<i>Name</i>	kerberos.service.authenticator.time
<i>Description</i>	Contains the time extracted from the AP_REQ message on the Kerberos acceptor side.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Kerberos Service Authentication
<i>Consumed By</i>	
<i>Required By</i>	

kerberos.service.context.established

<i>Name</i>	kerberos.service.context.established
<i>Description</i>	Indicates whether the service-side context has been established (true or false).
<i>Type</i>	java.lang.String
<i>Generated By</i>	Kerberos Service Authentication
<i>Consumed By</i>	
<i>Required By</i>	

kerberos.service.subject.id

<i>Name</i>	kerberos.service.subject.id
<i>Description</i>	Contains the principal name of the Kerberos service.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Kerberos Service Authentication Kerberos Client Authentication
<i>Consumed By</i>	
<i>Required By</i>	

kerberos.service.ticket.principal

<i>Name</i>	kerberos.service.ticket.principal
<i>Description</i>	Contains the principal extracted from the AP_REQ message on the Kerberos acceptor side.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Kerberos Service Authentication
<i>Consumed By</i>	
<i>Required By</i>	

kerberos.service.ticket.realm

<i>Name</i>	kerberos.service.ticket.realm
<i>Description</i>	Contains the realm extracted from the AP_REQ message on the Kerberos acceptor side.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Kerberos Service Authentication
<i>Consumed By</i>	
<i>Required By</i>	

kerberos.session.key

<i>Name</i>	kerberos.session.key
<i>Description</i>	On the Kerberos server-side, contains the session key extracted from the service ticket and authenticator in the Kerberos AP_REQ message. On the Kerberos client-side, contains the session key extracted from the private credentials of the client subject in the KerebroTicket .
<i>Type</i>	javax.security.auth.kerberos.KerberosKey
<i>Generated By</i>	Kerberos Service Authentication Kerberos Client Authentication
<i>Consumed By</i>	
<i>Required By</i>	

mcafee.status

<i>Name</i>	mcafee.status
<i>Description</i>	Stores the overall message status of all message parts after a McAfee virus scan (for example, NOVIRUS , INFECTED , REPAIRED , and REMOVED).
<i>Type</i>	java.lang.String
<i>Generated By</i>	McAfee Anti-Virus
<i>Consumed By</i>	

<i>Required By</i>	
--------------------	--

message.key

<i>Name</i>	message.key
<i>Description</i>	Contains a hash of the request message. By default, used as the key to search for objects in the cache.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Create Key
<i>Consumed By</i>	
<i>Required By</i>	Cache Attribute Is Cached? Remove Cached Attribute

saml.assertion

<i>Name</i>	saml.assertion
<i>Description</i>	Contains the SAML assertion that was used for authentication, authorization, or attribute extraction.
<i>Type</i>	org.w3c.dom.Element
<i>Generated By</i>	Retrieve Attributes from SAML Attribute Assertion SAML Authentication SAML Authorization
<i>Consumed By</i>	
<i>Required By</i>	

saml.assertion.position

<i>Name</i>	saml.assertion.position
<i>Description</i>	Indicates the position of the SAML assertion within a given WS-Security block. There may be more than 1 assertion in a WS-Security block, and so this attribute can be used to select the appropriate one.
<i>Type</i>	java.lang.Integer
<i>Generated By</i>	Retrieve Attributes from SAML Attribute Assertion SAML Authentication SAML Authorization
<i>Consumed By</i>	
<i>Required By</i>	

saml.wsblockinfo

<i>Name</i>	saml.wsblockinfo
<i>Description</i>	Stores the WS-Security block that contains the relevant SAML assertion.
<i>Type</i>	com.vordel.common.util.WSBlockInfo
<i>Generated By</i>	Retrieve Attributes from SAML Attribute Assertion SAML Authentication SAML Authorization
<i>Consumed By</i>	
<i>Required By</i>	

samlpdp.response.assertion

<i>Name</i>	samlpdp.response.assertion
<i>Description</i>	Contains the SAML assertion from the SAML P response.
<i>Type</i>	org.w3c.dom.Element
<i>Generated By</i>	Retrieve Attributes from SAML Attribute Assertion SAML PDP Authorization
<i>Consumed By</i>	
<i>Required By</i>	

samlpdp.response.doc

<i>Name</i>	samlpdp.response.doc
<i>Description</i>	Contains the SAML P response from the SAML PDP as an XML Document.
<i>Type</i>	org.w3c.dom.Document
<i>Generated By</i>	Retrieve Attributes from SAML PDP SAML PDP Authorization SAML PDP Response XML-Signature Verification SAML PDP XML-Signature Response Verification
<i>Consumed By</i>	
<i>Required By</i>	

samlpdp.response.namespace.saml

<i>Name</i>	samlpdp.response.namespace.saml
<i>Description</i>	Stores the SAML namespace that was used in the SAML P response.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Retrieve Attributes from SAML PDP SAML PDP Authorization
<i>Consumed By</i>	

<i>Required By</i>	
--------------------	--

samlpdp.response.namespace.samlp

<i>Name</i>	samlpdp.response.namespace.samlp
<i>Description</i>	Stores the SAML P namespace that was used in the SAML P response.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Retrieve Attributes from SAML PDP SAML PDP Authorization
<i>Consumed By</i>	
<i>Required By</i>	

samlpdp.subject.format

<i>Name</i>	samlpdp.subject.format
<i>Description</i>	Contains the subject format used in the SAML P request to the SAML PDP.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Retrieve Attributes from SAML PDP SAML PDP Authorization
<i>Consumed By</i>	
<i>Required By</i>	

samlpdp.subject.id

<i>Name</i>	samlpdp.subject.id
<i>Description</i>	Identifies the subject used in the SAML P request to the SAML PDP.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Retrieve Attributes from SAML PDP SAML PDP Authorization
<i>Consumed By</i>	
<i>Required By</i>	

service.name

<i>Name</i>	service.name
<i>Description</i>	Stores the name of the backend Web Service. For example, this is used when the service is displayed in real-time monitoring tools.

<i>Type</i>	java.lang.String
<i>Generated By</i>	Set Service Name Set Web Service Context Web Service Filter
<i>Consumed By</i>	
<i>Required By</i>	

siteminder.agent

<i>Name</i>	siteminder.agent
<i>Description</i>	Indicates the name of the agent that Enterprise Gateway uses to connect to SiteMinder as.
<i>Type</i>	java.lang.String
<i>Generated By</i>	SiteMinder Certificate Authentication SiteMinder Session Validation
<i>Consumed By</i>	
<i>Required By</i>	SiteMinder Authorization SiteMinder Logout

siteminder.decision

<i>Name</i>	siteminder.decision
<i>Description</i>	Enterprise Gateway can ask SiteMinder to make an authorization decision based on whether or not SiteMinder authenticates the user. SiteMinder returns its decision to Enterprise Gateway where it is stored in this attribute.
<i>Type</i>	com.vordel.circuit.siteminder.SiteMinderDecision
<i>Generated By</i>	SiteMinder Certificate Authentication SiteMinder Session Validation
<i>Consumed By</i>	
<i>Required By</i>	SiteMinder Authorization SiteMinder Logout

soap.request.action

<i>Name</i>	soap.request.action
<i>Description</i>	Specifies the SOAP Action in the HTTP header.
<i>Type</i>	java.lang.String
<i>Generated By</i>	SOAP Action Resolver
<i>Consumed By</i>	
<i>Required By</i>	

soap.request.method

<i>Name</i>	soap.request.method
<i>Description</i>	Stores the SOAP operation name. This is the first element under the SOAP body for an RPC encoded SOAP message.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Operation Name
<i>Consumed By</i>	
<i>Required By</i>	

soap.request.method.namespace

<i>Name</i>	soap.request.method.namespace
<i>Description</i>	Contains the namespace of the element identified by the value of the soap.request.method attribute. In other words, this attribute indicates the namespace of the SOAP operation.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Operation Name
<i>Consumed By</i>	
<i>Required By</i>	

soasecuritymanager.action

<i>Name</i>	soasecuritymanager.action
<i>Description</i>	Contains the action to take.
<i>Type</i>	java.lang.String
<i>Generated By</i>	
<i>Consumed By</i>	
<i>Required By</i>	

soasecuritymanager.agent

<i>Name</i>	soasecuritymanager.agent
<i>Description</i>	Contains the name of the agent used by Enterprise Gateway to connect to the CA SOA Security Manager.
<i>Type</i>	java.lang.String
<i>Generated By</i>	CA SOA Security Manager Authentication
<i>Consumed By</i>	
<i>Required By</i>	CA SOA Security Manager Authorization

soasecuritymanager.decision

<i>Name</i>	soasecuritymanager.decision
<i>Description</i>	Contains the authentication/authorization decision made by CA SOA Security Manager.
<i>Type</i>	java.lang.String
<i>Generated By</i>	CA SOA Security Manager Authentication
<i>Consumed By</i>	
<i>Required By</i>	CA SOA Security Manager Authorization

soasecuritymanager.realmdef

<i>Name</i>	soasecuritymanager.realmdef
<i>Description</i>	Contains the authentication/authorization realm of the CA SOA Security Manager.
<i>Type</i>	java.lang.String
<i>Generated By</i>	CA SOA Security Manager Authentication
<i>Consumed By</i>	
<i>Required By</i>	CA SOA Security Manager Authorization

soasecuritymanager.resource

<i>Name</i>	soasecuritymanager.resource
<i>Description</i>	Contains the name of the resource that the client is attempting to access.
<i>Type</i>	java.lang.String
<i>Generated By</i>	
<i>Consumed By</i>	
<i>Required By</i>	

soasecuritymanager.resource.context

<i>Name</i>	soasecuritymanager.resource.context
<i>Description</i>	Describes the context of the resource that the user is attempting to access.
<i>Type</i>	java.lang.String
<i>Generated By</i>	CA SOA Security Manager Authentication
<i>Consumed By</i>	
<i>Required By</i>	CA SOA Security Manager Authorization

webservice.context

<i>Name</i>	webservice.context
<i>Description</i>	Stores the Web Service context of the backend Web Service. For example, this is used to identify the service in the Web Service Repository when a WSDL request is received.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Set Web Service Context Web Service Filter
<i>Consumed By</i>	
<i>Required By</i>	Return WSDL Schema Validation

ws.username.token.name

<i>Name</i>	ws.username.token.name
<i>Description</i>	Associates the generated UsernameToken with the authenticated user.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Insert WS-Security Username Token
<i>Consumed By</i>	
<i>Required By</i>	

wss.timestamp

<i>Name</i>	wss.timestamp
<i>Description</i>	The timestamp in the WSS header block that is obtained from the message for a particular SOAP actor/role. Indicates how long the security data remains valid for.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract WSS Timestamp
<i>Consumed By</i>	
<i>Required By</i>	

wss.usernameToken

<i>Name</i>	wss.usernameToken
<i>Description</i>	The usernameToken in the WSS header block that is obtained from the message for a particular SOAP actor/role.
<i>Type</i>	java.lang.String
<i>Generated By</i>	Extract WSS UsernameToken
<i>Consumed By</i>	

<i>Required By</i>	
--------------------	--

xacml.decision

<i>Name</i>	xacml.decision
<i>Description</i>	Contains the authorization decision sent by the XACML Policy Decision Point to the XACML Policy Enforcement Point (for example, Permit , Deny , Indeterminate , or NotApplicable).
<i>Type</i>	java.lang.String
<i>Generated By</i>	XACML PEP
<i>Consumed By</i>	
<i>Required By</i>	

xacml.result.xml

<i>Name</i>	xacml.result.xml
<i>Description</i>	Contains the XML response message that includes the authorization decision sent by the XACML Policy Decision Point to the XACML Policy Enforcement Point (for example, Permit , Deny , Indeterminate , or NotApplicable).
<i>Type</i>	java.lang.String
<i>Generated By</i>	XACML PEP
<i>Consumed By</i>	
<i>Required By</i>	

xacml.statuscode

<i>Name</i>	xacml.statuscode
<i>Description</i>	Contains the XACML status code message (for example, urn:oasis:names:tc:xacml:1.0:status:ok syntax-error , processing-error , or missing-attribute).
<i>Type</i>	java.lang.String
<i>Generated By</i>	XACML PEP
<i>Consumed By</i>	
<i>Required By</i>	

xsd.errors

<i>Name</i>	<code>xsd.errors</code>
<i>Description</i>	Stores validation errors generated when a schema validation check fails. For example, you can return an appropriate SOAP Fault to the client by writing out the contents of this attribute. You can configure a Set Message filter to write a custom response message back to the client, and place it on the failure path of the Schema Validation filter.
<i>Type</i>	<code>java.lang.String</code>
<i>Generated By</i>	Schema Validation
<i>Consumed By</i>	
<i>Required By</i>	

Message Filter Reference

Extract WSS Header

<i>Name</i>	Extract WSS Header
<i>Description</i>	Extracts a WS-Security <Header> block from a message, and stores it in the <code>authentication.ws.wsblockinfo</code> message attribute.
<i>Category</i>	Attributes
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	authentication.ws.wsblockinfo
<i>Tutorial</i>	Extract WSS Header

Extract WSS Timestamp

<i>Name</i>	Extract WSS Timestamp
<i>Description</i>	Extracts a WS-Utility Timestamp from a message. The timestamp is stored in a specified message attribute to be processed later in a circuit. Defaults to the <code>wss.timestamp</code> message attribute.
<i>Category</i>	Attributes
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	wss.timestamp
<i>Tutorial</i>	Extract WSU Timestamp

Extract WSS Username Token

<i>Name</i>	Extract WSS Username Token
<i>Description</i>	Extracts a WS-Security UsernameToken from a message if it exists. The extracted UsernameToken is stored in the <code>wss.usernameToken</code> message attribute.
<i>Category</i>	Attributes
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	wss.usernameToken
<i>Tutorial</i>	Extract WSS Username Token

Insert SAML Attribute Assertion

<i>Name</i>	Insert SAML Attribute Assertion
-------------	---------------------------------

<i>Description</i>	Inserts a SAML attribute assertion into the downstream message.
<i>Category</i>	Attributes
<i>Required Attributes</i>	attribute.lookup.list attribute.subject.format attribute.subject.id content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Insert SAML Attribute Assertion

Retrieve Attributes from Directory Server

<i>Name</i>	Retrieve Attribute from Directory Server
<i>Description</i>	Retrieves user attributes from an LDAP directory.
<i>Category</i>	Attributes
<i>Required Attributes</i>	authentication.subject.id authentication.subject.format
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	attribute.lookup.list attribute.subject.format attribute.subject.id
<i>Tutorial</i>	Retrieve Attribute from Directory Server

Retrieve Attribute from HTTP Header

<i>Name</i>	Retrieve Attribute from HTTP Header
<i>Description</i>	Retrieves the value of an HTTP header and sets it to a user-specified message attribute.
<i>Category</i>	Attributes
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Retrieve Attribute from HTTP Header

Retrieve Attributes from Database

<i>Name</i>	Retrieve Attributes from Database
<i>Description</i>	Retrieves user attributes from a specified database.
<i>Category</i>	Attributes
<i>Required Attributes</i>	authentication.subject.id

	authentication.subject.format
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	attribute.lookup.list authentication.subject.id authentication.subject.format
<i>Tutorial</i>	Retrieve Attributes from Database

Retrieve Attribute from Message

<i>Name</i>	Retrieve Attribute from Message
<i>Description</i>	Retrieves the value of an XML attribute or element from the message and sets it to a user-specified message attribute.
<i>Category</i>	Attributes
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Retrieve Attribute from Message

Retrieve Attributes from SAML Attribute Assertion

<i>Name</i>	Retrieve Attribute from SAML Attribute Assertion
<i>Description</i>	Retrieves user attributes from a SAML attribute assertion and stores them in the attribute.lookup.list message attribute.
<i>Category</i>	Attributes
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	attribute.lookup.list attribute.subject.format attribute.subject.id saml.assertion saml.assertion.position saml.wsblockinfo
<i>Tutorial</i>	Retrieve Attribute from SAML Attribute Assertion

Retrieve Attributes from SAML PDP

<i>Name</i>	Retrieve Attribute from SAML PDP
<i>Description</i>	When a user has been successfully authenticated, the Enterprise Gateway can send a SAML (SAML Protocol) request to the SAML PDP to obtain user attributes. The PDP packages the relevant attributes into a SAML attribute assertion and returns the assertion to Enterprise Gateway in

	a SAML response. Enterprise Gateway validates the response and can optionally insert the attribute assertion into the downstream message.
<i>Category</i>	Attributes
<i>Required Attributes</i>	authentication.subject.id authentication.subject.format authentication.method
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	attribute.lookup.list samlpdp.response.assertion samlpdp.response.doc samlpdp.response.namespace.saml samlpdp.response.namespace.samlp samlpdp.subject.format samlpdp.subject.id
<i>Tutorial</i>	Retrieve Attributes from SAML PDP

Retrieve Attributes from Tivoli

<i>Name</i>	Retrieve Attributes from Tivoli
<i>Description</i>	You can use this filter when you need to retrieve user attributes independently from authorizing the user against Tivoli Access Manager.
<i>Category</i>	Attributes
<i>Required Attributes</i>	authentication.subject.id
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	attribute.lookup.list attribute.subject.format attribute.subject.id
<i>Tutorial</i>	Retrieve Attributes from Tivoli

Retrieve Attributes from User Store

<i>Name</i>	Retrieve from User Store
<i>Description</i>	Retrieves user attributes from the User Store and stores them in the attribute.lookup.list message attribute.
<i>Category</i>	Attributes
<i>Required Attributes</i>	authentication.subject.id
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	attribute.lookup.list authentication.subject.id authentication.subject.format
<i>Tutorial</i>	Retrieve Attributes from User Store

SAML PDP XML-Signature Response Verification

<i>Name</i>	SAML PDP XML-Signature Response Verification
<i>Description</i>	Typically, a SAML PDP will sign SAML responses returned to Enterprise Gateway. In such cases, Enterprise Gateway can validate the signature on the response using this filter.
<i>Category</i>	Attributes
<i>Required Attributes</i>	samlpdp.response.doc
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	certificate certificates
<i>Tutorial</i>	SAML PDP Attributes

Attribute Authentication

<i>Name</i>	Attribute Authentication
<i>Description</i>	Authenticates user credentials specified in Enterprise Gateway message attributes against a configured user store.
<i>Category</i>	Authentication
<i>Required Attributes</i>	authentication.subject.id authentication.subject.password
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	authentication.method authentication.subject.format authentication.subject.id authentication.subject.orig.format authentication.subject.orig.id authentication.subject.password
<i>Tutorial</i>	Attribute Authentication

HTML Form-based Authentication

<i>Name</i>	HTML Form-based Authentication
<i>Description</i>	Authenticates Enterprise Gateway client user credentials specified in an HTML form against a configured user store.
<i>Category</i>	Authentication
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	authentication.method authentication.subject.format authentication.subject.id authentication.subject.orig.format authentication.subject.orig.id authentication.subject.password

<i>Tutorial</i>	HTML Form-based Authentication
-----------------	--

HTTP Basic Authentication

<i>Name</i>	HTTP Basic Authentication
<i>Description</i>	Authenticates a client against a configured user store using HTTP basic authentication.
<i>Category</i>	Authentication
<i>Required Attributes</i>	http.headers http.request.verb
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	authentication.method authentication.subject.format authentication.subject.id authentication.subject.orig.format authentication.subject.orig.id authentication.subject.password
<i>Tutorial</i>	HTTP Basic Authentication

HTTP Digest Authentication

<i>Name</i>	HTTP Digest Authentication
<i>Description</i>	Authenticates a client against a configured user store using HTTP digest authentication.
<i>Category</i>	Authentication
<i>Required Attributes</i>	http.headers http.request.verb
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	authentication.method authentication.subject.format authentication.subject.id authentication.subject.orig.format authentication.subject.orig.id authentication.subject.password
<i>Tutorial</i>	HTTP Digest Authentication

IP Address

<i>Name</i>	IP Address
<i>Description</i>	Allows or denies access to an IP address or range of IP addresses.
<i>Category</i>	Authentication
<i>Required Attributes</i>	http.request.clientaddr

<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	IP Address

SSL Authentication

<i>Name</i>	SSL Authentication
<i>Description</i>	Authenticates a user's SSL certificate.
<i>Category</i>	Authentication
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	authentication.cert authentication.issuer.format authentication.issuer.id authentication.method authentication.subject.format authentication.subject.id certificate certificates
<i>Tutorial</i>	SSL Authentication

CA SOA Security Manager Authentication

<i>Name</i>	CA SOA Security Manager Authentication
<i>Description</i>	Authenticates a user against CA SOA Security Manager.
<i>Category</i>	Authentication
<i>Required Attributes</i>	content.body http.request.clientaddr http.request.uri http.request.verb
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	authentication.method authentication.subject.format authentication.subject.id soasecuritymanager.agent soasecuritymanager.decision soasecuritymanager.realmdef soasecuritymanager.resource.context
<i>Tutorial</i>	CA SOA Security Manager Authentication

HTTP Header Authentication

<i>Name</i>	HTTP Header Authentication
<i>Description</i>	Extracts a user credential from an HTTP header and uses

	it to authenticate the user. Typically, a username, X.509 certificate, or Distinguished Name is extracted from the HTTP header.
<i>Category</i>	Authentication
<i>Required Attributes</i>	http.headers
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	authentication.cert authentication.issuer.format authentication.issuer.id authentication.issuer.orig.format authentication.issuer.orig.id authentication.method authentication.subject.format authentication.subject.id authentication.subject.orig.format authentication.subject.orig.id certificate certificates
<i>Tutorial</i>	HTTP Header Authentication

Insert SAML Authentication Assertion

<i>Name</i>	Insert SAML Authentication Assertion
<i>Description</i>	Inserts a SAML authentication assertion into the downstream message on behalf of an authenticated user.
<i>Category</i>	Authentication
<i>Required Attributes</i>	authentication.method authentication.subject.format authentication.subject.id content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Insert SAML Authentication Assertion

Insert WS-Security Username Token

<i>Name</i>	Insert WS-Security Username Token
<i>Description</i>	Inserts a WS-Security Token into the downstream message on behalf of an authenticated client.
<i>Category</i>	Authentication
<i>Required Attributes</i>	authentication.subject.id content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	ws.username.token.name
<i>Tutorial</i>	Insert WS-Security Username Token

Insert Timestamp

<i>Name</i>	Insert Timestamp
<i>Description</i>	Inserts a WS-Utility (WSU) Timestamp into a WS-Security Header to specify the lifetime of the message to which it is added.
<i>Category</i>	Authentication
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Insert Timestamp

Kerberos Client Authentication

<i>Name</i>	Kerberos Client Authentication
<i>Description</i>	Obtains a service ticket for a Kerberos Service, and uses it to authenticate to the service.
<i>Category</i>	Authentication
<i>Required Attributes</i>	http.destination.host
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	authentication.method authentication.subject.format authentication.subject.id kerberos.client.context.established kerberos.context kerberos.mechanism.oid kerberos.profile.ap.req.bst.id kerberos.profile.ap.req.sha1 kerberos.service.subject.id kerberos.session.key
<i>Tutorial</i>	Kerberos Client Authentication

Kerberos Service Authentication

<i>Name</i>	Kerberos Service Authentication
<i>Description</i>	Consumes a Kerberos token to authenticate a Kerberos Client.
<i>Category</i>	Authentication
<i>Required Attributes</i>	http.destination.host
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	authentication.method authentication.subject.format authentication.subject.id kerberos.mechanism.oid kerberos.profile.ap.req.sha1 kerberos.service.authenticator.principal

	kerberos.service.authenticator.realm kerberos.service.authenticator.time kerberos.service.context.established kerberos.service.subject.id kerberos.service.ticket.principal kerberos.service.ticket.realm kerberos.session.key
<i>Tutorial</i>	Kerberos Service Authentication

SAML Authentication

<i>Name</i>	SAML Authentication
<i>Description</i>	Validates a SAML authentication assertion to make sure it has not expired.
<i>Category</i>	Authentication
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	authentication.method authentication.subject.id authentication.subject.format saml.assertion saml.assertion.position saml.wsblockinfo
<i>Tutorial</i>	SAML Authentication

SAML Authentication XML-Signature Verification

<i>Name</i>	SAML Authentication XML-Signature Verification
<i>Description</i>	Validates the signature on a SAML authentication assertion.
<i>Category</i>	Authentication
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	authentication.cert authentication.issuer.format authentication.issuer.id certificate certificates
<i>Tutorial</i>	SAML Authentication XML-Signature Verification

SAML PDP Response XML-Signature Verification

<i>Name</i>	Authentication: SAML PDP Response XML-Signature Verification
-------------	--

<i>Description</i>	Typically a SAML PDP will sign SAML responses and/or the issued SAML assertion itself. This filter can be used to validate the signature on the SAML response.
<i>Category</i>	Authentication
<i>Required Attributes</i>	samlpdp.response.doc
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	certificate certificates
<i>Tutorial</i>	Authentication: SAML PDP Response XML-Signature Verification

XML-Signature Authentication

<i>Name</i>	XML-Signature Authentication
<i>Description</i>	Enterprise Gateway can authenticate a client by validating the XML-Signature on an incoming request. A successful signature validation proves that the client had access to the private key that was used to sign the request.
<i>Category</i>	Authentication
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	authentication.cert authentication.issuer.format authentication.issuer.id authentication.method authentication.subject.format authentication.subject.id certificate certificates
<i>Tutorial</i>	XML-Signature Authentication

Attribute Authorization

<i>Name</i>	Attribute Authorization
<i>Description</i>	This filter checks the values of user attributes that are stored in the <code>attribute.lookup.list</code> message attribute.
<i>Category</i>	Authorization
<i>Required Attributes</i>	attribute.lookup.list
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Attribute Authorization

CA SOA Security Manager Authorization

<i>Name</i>	CA SOA Security Manager Authorization
<i>Description</i>	Authorizes an authenticated user against CA SOA Security Manager.
<i>Category</i>	Authorization
<i>Required Attributes</i>	http.request.clientaddr soasecuritymanager.agent soasecuritymanager.decision soasecuritymanager.realmdef soasecuritymanager.resource.context
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	attribute.lookup.list attribute.subject.format attribute.subject.id
<i>Tutorial</i>	CA SOA Security Manager Authorization

Certificate Attributes Authorization

<i>Name</i>	Certificate Attributes Authorization
<i>Description</i>	Authorizes a user by examining the attributes in that user's X.509 certificate.
<i>Category</i>	Authorization
<i>Required Attributes</i>	authentication.subject.id authentication.subject.format
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Certificate Attributes Authorization

Check Group Membership

<i>Name</i>	Check Group Membership
<i>Description</i>	Checks whether the specified Enterprise Gateway User is a member of the specified Enterprise Gateway Group.
<i>Category</i>	Authorization
<i>Required Attributes</i>	authentication.subject.id
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Check Group Membership

Entrust GetAccess Authorization

<i>Name</i>	GetAccess Authorization
-------------	-------------------------

<i>Description</i>	Authorizes an authenticated user against Entrust's GetAccess.
<i>Category</i>	Authorization
<i>Required Attributes</i>	authentication.subject.id authentication.subject.format content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	GetAccess Authorization

Insert SAML Authorization Assertion

<i>Name</i>	Insert SAML Authorization Assertion
<i>Description</i>	When the user has been successfully authorized, Enterprise Gateway can insert a SAML authorization assertion into the downstream message.
<i>Category</i>	Authorization
<i>Required Attributes</i>	authentication.subject.id authentication.subject.format content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Insert SAML Authorization Assertion

RSA Access Manager Authorization

<i>Name</i>	Access Manager
<i>Description</i>	Authorizes an authenticated user against RSA's ClearTrust Authorization Server.
<i>Category</i>	Authorization
<i>Required Attributes</i>	authentication.subject.id authentication.subject.format
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	ClearTrust Authorization

SAML Authorization

<i>Name</i>	SAML Authorization
<i>Description</i>	Authorizes a user by validating the SAML authorization assertion in an incoming request.
<i>Category</i>	Authorization

<i>Required Attributes</i>	authentication.subject.id authentication.subject.format content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	saml.assertion saml.assertion.position saml.wsblockinfo
<i>Tutorial</i>	SAML Authorization

SAML Authorization XML-Signature Verification

<i>Name</i>	SAML Authorization XML-Signature Verification
<i>Description</i>	Validates the XML-Signature on a SAML authorization assertion.
<i>Category</i>	Authorization
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	certificate certificates
<i>Tutorial</i>	SAML Authorization XML-Signature Verification

SAML PDP Authorization

<i>Name</i>	SAML PDP Authorization
<i>Description</i>	Generates a SAML PDP authorization request to a SAML PDP on behalf of an authenticated user. The SAML PDP generates a SAML authorization assertion and returns it to Enterprise Gateway in a SAML PDP response. Enterprise Gateway validates the response and can optionally insert the assertion into the downstream message.
<i>Category</i>	Authorization
<i>Required Attributes</i>	authentication.method authentication.subject.id authentication.subject.format
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	samlpdp.response.assertion samlpdp.response.doc samlpdp.response.namespace.saml samlpdp.response.namespace.samlp samlpdp.subject.id samlpdp.subject.format
<i>Tutorial</i>	SAML PDP Authorization

SAML PDP Response XML-Signature Verification

<i>Name</i>	SAML PDP Response XML-Signature Verification
<i>Description</i>	Typically a SAML PDP will sign SAML responses and/or the issued SAML assertion itself. This filter can be used to validate the signature on the SAML response.
<i>Category</i>	Authorization
<i>Required Attributes</i>	samlpdp.response.doc
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	certificate certificates
<i>Tutorial</i>	SAML PDP Response XML-Signature Verification

Tivoli Authorization

<i>Name</i>	Tivoli Authorization
<i>Description</i>	Authorizes an authenticated user against IBM's Tivoli Access Manager.
<i>Category</i>	Authorization
<i>Required Attributes</i>	authentication.subject.id authentication.subject.format
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Tivoli Authorization

XACML PEP

<i>Name</i>	XACML PEP
<i>Description</i>	Configures an eXtensible Access Control Markup Language (XACML) Policy Enforcement Point (PEP)
<i>Category</i>	Authorization
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	xacml.decision xacml.result.xml xacml.statuscode
<i>Tutorial</i>	XACML PEP

SiteMinder Authorization

<i>Name</i>	SiteMinder Authorization
<i>Description</i>	Authorizes a user against CA's SiteMinder. The user must have been authenticated to SiteMinder before they can be

	authorized.
<i>Category</i>	CA SiteMinder
<i>Required Attributes</i>	siteminder.agent siteminder.decision
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	attribute.lookup.list attribute.subject.format attribute.subject.id
<i>Tutorial</i>	SiteMinder Authorization

SiteMinder Certificate Authentication

<i>Name</i>	SiteMinder Certificate Authentication
<i>Description</i>	Authenticates a user's certificate against SiteMinder.
<i>Category</i>	CA SiteMinder
<i>Required Attributes</i>	certificate certificates
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	authentication.method authentication.format authentication.subject.id siteminder.agent siteminder.decision
<i>Tutorial</i>	SiteMinder Certificate Authentication

SiteMinder Logout

<i>Name</i>	SiteMinder Logout
<i>Description</i>	Terminates a user's SiteMinder session by invalidating the user's single sign-on token.
<i>Category</i>	CA SiteMinder
<i>Required Attributes</i>	siteminder.agent siteminder.decision
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	SiteMinder Logout

SiteMinder Session Validation

<i>Name</i>	SiteMinder Session Validation
<i>Description</i>	Extracts a user's single sign-on token from the message and validates it against SiteMinder.

<i>Category</i>	CA SiteMinder
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	authentication.method authentication.subject.format authentication.subject.id siteminder.agent siteminder.decision
<i>Tutorial</i>	SiteMinder Session Validation

Cache Attribute

<i>Name</i>	Cache Attribute
<i>Description</i>	Specifies which part of the message is cached. Typically, response messages are cached, so this filter is usually configured after the routing filters in a circuit.
<i>Category</i>	Cache
<i>Required Attributes</i>	message.key content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Cache Attribute

Create Key

<i>Name</i>	Create Key
<i>Description</i>	Specifies which part of a message determines if the message is unique (for example, message body, HTTP header, client IP address, and so on).
<i>Category</i>	Cache
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	message.key
<i>Tutorial</i>	Create Key

Is Cached?

<i>Name</i>	Is Cached?
<i>Description</i>	Looks up a named cache to see if a specified message attribute is already cached. A message attribute is used as the key to search for in the cache (defaults to <code>message.key</code>). If the lookup succeeds, the retrieved value overrides a specified message attribute (defaults to <code>con-</code>

	<code>tent.body</code>).
<i>Category</i>	Cache
<i>Required Attributes</i>	message.key
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	content.body
<i>Tutorial</i>	Is Cached?

Remove Cached Attribute

<i>Name</i>	Remove Cached Attribute
<i>Description</i>	Deletes a message attribute value that has been stored in a cache.
<i>Category</i>	Cache
<i>Required Attributes</i>	message.key
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Remove Cached Attribute

Certificate Chain

<i>Name</i>	Certificate Chain
<i>Description</i>	Ensures that a trusted CA (Certificate Authority) issued the certificate. Trusted CA certificates are stored in the Oracle Trusted Certificate Store.
<i>Category</i>	Certificate
<i>Required Attributes</i>	certificates
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Certificate Chain

Certificate Revocation List (Dynamic)

<i>Name</i>	Certificate Revocation List (dynamic)
<i>Description</i>	Validates a certificate against a CRL and automatically retrieves the CRL periodically.
<i>Category</i>	Certificate
<i>Required Attributes</i>	certificates
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Certificate Revocation List (Dynamic)

Certificate Revocation List (LDAP)

<i>Name</i>	CRL (in LDAP)
<i>Description</i>	Looks up a user's certificate in an LDAP-based CRL to see if that user has been revoked.
<i>Category</i>	Certificate
<i>Required Attributes</i>	certificates
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Certificate Revocation List (LDAP)

Certificate Revocation List Responder

<i>Name</i>	CRL Responder
<i>Description</i>	Configures the Enterprise Gateway to act as CRL responder by returning CRL files to clients.
<i>Category</i>	Certificate
<i>Required Attributes</i>	certificates
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	CRL Responder

Certificate Revocation List (Static)

<i>Name</i>	CRL (static)
<i>Description</i>	Looks up a user's certificate in a file-based CRL to see if that user has been revoked.
<i>Category</i>	Certificate
<i>Required Attributes</i>	certificates
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Static CRL Certificate Validation

Create Thumbprint from Certificate

<i>Name</i>	Create Thumbprint
<i>Description</i>	Used to create a human-readable thumbprint (or fingerprint) from the X.509 certificate that is stored in the <code>certificate</code> message attribute. The generated thumbprint is stored in the <code>certificate.thumbprint</code> attribute.
<i>Category</i>	Certificate
<i>Required Attributes</i>	certificate

<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	certificate.thumbprint
<i>Tutorial</i>	Create Thumbprint

Extract Certificate Attributes

<i>Name</i>	Extract Certificate Attributes
<i>Description</i>	Extracts the X.509 attributes from a certificate stored in a specified Oracle message attribute. Typically, this filter is used in conjunction with a Find Certificate filter.
<i>Category</i>	Certificate
<i>Required Attributes</i>	certificate
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	attribute.lookup.list attribute.subject.format attribute.subject.id cert.basic.constraints cert.extended.key.usage cert.hash.md5 cert.hash.sha1 cert.issuer.alternative.name cert.issuer.id cert.issuer.id.c cert.issuer.id.cn cert.issuer.id.emailaddress cert.issuer.id.l cert.issuer.id.o cert.issuer.id.ou cert.issuer.id.st cert.key.usage.cRLSign cert.key.usage.dataEncipherment cert.key.usage.digitalSignature cert.key.usage.encipherOnly cert.key.usage.keyAgreement cert.key.usage.keyCertSign cert.key.usage.keyEncipherment cert.key.usage.nonRepudiation cert.not.after cert.not.before cert.serial.number cert.signature.algorithm cert.subject.alternative.name cert.subject.id cert.subject.id.c cert.subject.id.cn cert.subject.id.emailaddress cert.subject.id.o cert.subject.id.ou cert.subject.id.st cert.version
<i>Tutorial</i>	Extract Certificate Attributes

Find Certificate

<i>Name</i>	Find Certificate
<i>Description</i>	Locates a certificate from a message attribute, HTTP header, message attachment, or extracts a certificate from the User Store. The extracted certificate is stored in a user-specified message attribute. This new attribute will then appear as a Generated Attribute in the policy. The certificate is stored in the certificate attribute by default.
<i>Category</i>	Certificate
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	certificate certificates
<i>Tutorial</i>	Find Certificate

OCSP (Online Certificate Status Protocol)

<i>Name</i>	OCSP Certificate Validation
<i>Description</i>	Checks the status of a user's certificate against a group of OCSP responders.
<i>Category</i>	Certificate
<i>Required Attributes</i>	certificates
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	OCSP

XKMS (XML Key Management and Security)

<i>Name</i>	XKMS Certificate Validation
<i>Description</i>	Validates a user's certificate against a group of XKMS responders.
<i>Category</i>	Certificates
<i>Required Attributes</i>	certificates
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	XKMS

Content Type Filtering

<i>Name</i>	Content Type Filtering
<i>Description</i>	Filters MIME and DIME messages based on the types of their attachments.

<i>Category</i>	Content Filtering
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Content Type Filter

Content Validation

<i>Name</i>	Content Validation
<i>Description</i>	Runs a boolean XPath expression on the incoming request.
<i>Category</i>	Content Filtering
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Content Validation

Throttling

<i>Name</i>	Throttling
<i>Description</i>	Limits the number of messages a client can send in a specified interval through the policy in which this filter is configured. In other words, it provides filtering of messages on a per client, per service basis.
<i>Category</i>	Content Filtering
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Throttling

McAfee Anti-Virus

<i>Name</i>	McAfee Anti-Virus
<i>Description</i>	Scans incoming HTTP requests and their attachments for viruses and exploits. Supports cleaning of messages from infections, provides scan presets for detection levels, and reports overall message status after scanning.
<i>Category</i>	Content Filtering
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	mcafee.status

<i>Tutorial</i>	McAfee Virus Scanner
-----------------	--------------------------------------

Schema Validation

<i>Name</i>	Schema Validation
<i>Description</i>	Validates the contents of the message body against a selected XML Schema. This ensures that the message adheres to the correct message format, and can also ensure that the message contains appropriate data.
<i>Category</i>	Content Filtering
<i>Required Attributes</i>	content.body webservice.context
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	xsd.errors
<i>Tutorial</i>	Schema Validation

Validate HTTP Headers

<i>Name</i>	Validate HTTP Headers
<i>Description</i>	Filters MIME and DIME messages based on the types of their attachments.
<i>Category</i>	Content Filtering
<i>Required Attributes</i>	http.headers
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Validate HTTP Headers

Validate Message Attribute

<i>Name</i>	Validate Message Attribute
<i>Description</i>	Compares the value of a message attribute to a configured regular expression. It can also check for the presence of Threatening Content regular expressions such as SQL injection and buffer overflow attacks.
<i>Category</i>	Content Filtering
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Validate Message Attributes

XML Complexity

<i>Name</i>	XML Complexity
<i>Description</i>	Checks the depth and complexity of XML messages.
<i>Category</i>	Content Filtering
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	XML Complexity

Add HTTP Header

<i>Name</i>	Add HTTP Header
<i>Description</i>	Adds a user-specified HTTP header to the downstream message.
<i>Category</i>	Conversion
<i>Required Attributes</i>	http.headers
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Add HTTP Header

Set Message

<i>Name</i>	Set Message
<i>Description</i>	Sets the content of the message payload.
<i>Category</i>	Conversion
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	content.body
<i>Tutorial</i>	Set Message

Load File

<i>Name</i>	Load file
<i>Description</i>	Loads the contents of the specified file, and sets them as message content to be processed.
<i>Category</i>	Conversion
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	content.body
<i>Tutorial</i>	Load File

Remove HTTP Header

<i>Name</i>	Remove HTTP Header
<i>Description</i>	Removes a user-specified HTTP header from the downstream message.
<i>Category</i>	Conversion
<i>Required Attributes</i>	http.headers
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Remove HTTP Header

XSLT Transformation

<i>Name</i>	XSLT Transformation
<i>Description</i>	This filter uses an XSLT stylesheet to convert the body of the incoming request to an alternative XML grammar or format.
<i>Category</i>	Conversion
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	XSLT Transformation

XML-Decryption

<i>Name</i>	XML-Decryption
<i>Description</i>	Decrypts an XML-Encrypted message according to the settings configured in the XML-Decryption Settings filter. These settings are stored in the decryption.properties message attribute, which is a required attribute for this filter.
<i>Category</i>	Encryption
<i>Required Attributes</i>	content.body decryption.properties
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	XML-Decryption

XML-Decryption Settings

<i>Name</i>	XML-Decryption Settings
<i>Description</i>	This filter is used to specify the XML-Encrypted blocks to decrypt in the message. All of the encrypted blocks can be decrypted or a single encrypted block can be selected us-

	ing an XPath expression. The actual decryption is performed by the XML-Decryption filter.
<i>Category</i>	Encryption
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	decryption.properties
<i>Tutorial</i>	XML-Decryption Settings

XML-Encryption

<i>Name</i>	XML-Encryption
<i>Description</i>	Encrypts (part of) an XML message as specified in the XML Encryption Settings filter. The message will be encrypted such that only its intended recipients can decrypt it.
<i>Category</i>	Encryption
<i>Required Attributes</i>	content.body encryption.properties
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	XML-Encryption

XML-Encryption Settings

<i>Name</i>	XML-Encryption Settings
<i>Description</i>	This filter is used to specify the part(s) of the message to encrypt, and for whom the message is to be encrypted. Only the intended recipients will be able to decrypt the message.
<i>Category</i>	Encryption
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	encryption.properties
<i>Tutorial</i>	XML-Encryption Settings

SOAP Fault

<i>Name</i>	SOAP Fault
<i>Description</i>	If a SOAP Fault handler is configured for a policy, it handles any exceptions that occur in the policy. As such it dictates the format of the SOAP Fault that is returned to the client.
<i>Category</i>	Fault Handlers

<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	SOAP Fault

Sign Message

<i>Name</i>	XML Signature Generation
<i>Description</i>	Signs the selected part of the incoming request.
<i>Category</i>	Integrity
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Sign Message

XML-Signature Verification

<i>Name</i>	XML-Signature Verification
<i>Description</i>	Verifies the integrity of the incoming message by validating its XML-Signature. This ensures that the message was not tampered with after it was signed.
<i>Category</i>	Integrity
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	certificate certificates
<i>Tutorial</i>	Integrity XML-Signature Verification

Set Service Name

<i>Name</i>	Set Service Name
<i>Description</i>	Configures service-level monitoring details. For example, you can specify the service name displayed in real-time monitoring tools, and whether to store service usage metrics.
<i>Category</i>	Monitoring
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	service.name
<i>Tutorial</i>	Set Service Name

Alert

<i>Name</i>	Alert
<i>Description</i>	Sends an alert to a configured alerting destination.
<i>Category</i>	Monitoring
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Alert

Log Access

<i>Name</i>	Log Access
<i>Description</i>	Logs message details in Common Log Format to an Access Log. The log file is written to the /logs directory of your Enterprise Gateway installation.
<i>Category</i>	Monitoring
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Log Access

Log Message Payload

<i>Name</i>	Log Message Payload
<i>Description</i>	Logs the message payload, including HTTP headers and MIME/DIME attachments, at a particular point in the policy.
<i>Category</i>	Monitoring
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Log Message Payload

Operation Name

<i>Name</i>	Operation Name
<i>Description</i>	Compares the first element of the SOAP body (i.e. the SOAP operation) and its namespace to the values configured here.
<i>Category</i>	Resolvers
<i>Required Attributes</i>	content.body

<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	soap.request.method soap.request.method.namespace
<i>Tutorial</i>	Operation Name

Relative Path

<i>Name</i>	Relative Path
<i>Description</i>	Matches the relative path (i.e. uri) on which the request was received to the value configured here.
<i>Category</i>	Resolvers
<i>Required Attributes</i>	http.request.uri
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Relative Path

SOAPAction

<i>Name</i>	SOAPAction
<i>Description</i>	Matches the SOAPAction HTTP header on the incoming request to the value configured in this filter.
<i>Category</i>	Resolvers
<i>Required Attributes</i>	http.headers
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	soap.request.action
<i>Tutorial</i>	SOAP Action

Connection

<i>Name</i>	Connection
<i>Description</i>	This filter is responsible for connecting to the target Web Service or system. Enterprise Gateway can mutually authenticate to the endpoint using SSL certificates or HTTP basic/digest authentication.
<i>Category</i>	Routing
<i>Required Attributes</i>	content.body http.destination.host http.destination.port http.destination.protocol http.headers http.request.uri http.request.verb

<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	http.headers http.request.connection.error http.response.info http.response.status http.response.version
<i>Tutorial</i>	Connection

Dynamic Router

<i>Name</i>	Dynamic Router
<i>Description</i>	In cases where Enterprise Gateway is acting as a proxy, it can extract the URL from the request line of the HTTP request and route the message to this address.
<i>Category</i>	Routing
<i>Required Attributes</i>	http.request.uri
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	http.destination.host http.destination.port http.destination.protocol
<i>Tutorial</i>	Dynamic Router

Rewrite URL

<i>Name</i>	Rewrite URL
<i>Description</i>	In cases where Enterprise Gateway is acting as a proxy, it can forward the message on to the address specified in the request line of the HTTP request. This filter can be used to rewrite the URL of the original request to an alternative one, i.e. service virtualization.
<i>Category</i>	Routing
<i>Required Attributes</i>	http.request.uri
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Rewrite URL

Static Router

<i>Name</i>	Static Router
<i>Description</i>	The static router is used to configure connection details for a particular endpoint. Enterprise Gateway will route messages to the endpoint configured here.
<i>Category</i>	Routing

<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	http.destination.host http.destination.port http.destination.protocol
<i>Tutorial</i>	Static Router

Insert WS-Addressing

<i>Name</i>	Insert WS-Addressing
<i>Description</i>	Inserts WS-Addressing information into a SOAP message.
<i>Category</i>	Routing
<i>Required Attributes</i>	http.destination.host http.destination.port http.destination.protocol http.headers http.request.uri soap.request.action
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	SSL Authentication

Read WS-Addressing

<i>Name</i>	Read WS-Addressing
<i>Description</i>	Uses WS-Addressing information contained within a SOAP message to route the message.
<i>Category</i>	Routing
<i>Required Attributes</i>	content.body
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	http.destination.host http.destination.port http.destination.protocol http.request.uri
<i>Tutorial</i>	SSL Authentication

Save to File

<i>Name</i>	Save to file
<i>Description</i>	Writes the current message contents to a file.
<i>Category</i>	Routing
<i>Required Attributes</i>	content.body

<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Save to File

Abort

<i>Name</i>	Abort
<i>Description</i>	Forces a policy path to abort and throw an exception. This causes a SOAP Fault to be returned to the client.
<i>Category</i>	Utility
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Abort Filter

Copy/Modify Attributes

<i>Name</i>	Copy/Modify Attributes
<i>Description</i>	This filter can be used to copy the value of one message attribute to another.
<i>Category</i>	Utility
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Copy/Modify Attributes

False

<i>Name</i>	False
<i>Description</i>	Forces the policy path to return false.
<i>Category</i>	Utility
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	False Filter

Policy Shortcut

<i>Name</i>	Policy Shortcut
-------------	-----------------

<i>Description</i>	This filter can be used to pass control to another policy. It is very useful for creating a policy macro that contains small pieces of logic that you may wish to keep outside of a policy so that it can be re-used. This helps to keep the main logic of a policy uncluttered.
<i>Category</i>	Utility
<i>Required Attributes</i>	The required attributes for this filter are whatever attributed are required by the start node of the policy shortcut.
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	The generated attributes for this filter are the attributes that are returned from the end node of the policy shortcut.
<i>Tutorial</i>	Policy Shortcut

Reflect

<i>Name</i>	Reflect
<i>Description</i>	Echoes the request body back to the client.
<i>Category</i>	Utility
<i>Required Attributes</i>	content.body http.headers
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	http.response.status
<i>Tutorial</i>	Reflect Message

Reflect Message and Attributes

<i>Name</i>	Reflect Message and Attributes
<i>Description</i>	Echoes the request body and the current message attributes back to the client.
<i>Category</i>	Utility
<i>Required Attributes</i>	content.body http.headers
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Reflect Message and Attributes

True

<i>Name</i>	True
<i>Description</i>	Forces a true result from a policy path.
<i>Category</i>	Utility

<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	True Filter

Execute process

<i>Name</i>	Execute process
<i>Description</i>	Executes an external process from a policy circuit.
<i>Category</i>	Utility
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Execute External Process

Pause

<i>Name</i>	Pause
<i>Description</i>	This filter forces the policy to suspend processing for a specified time interval. When this interval has elapsed, the next filter in the policy path is executed immediately.
<i>Category</i>	Utility
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Pause Filter

Set Response Status

<i>Name</i>	Set Response Status
<i>Description</i>	Explicitly sets the response status of a given message.
<i>Category</i>	Utility
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Set Response Status

String Replace

<i>Name</i>	String Replace
<i>Description</i>	Replaces all or part of the value of the specified string in a message attribute.
<i>Category</i>	Utility
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	String Replace Filter

Trace

<i>Name</i>	Trace
<i>Description</i>	Forces the Enterprise Gateway to trace the current message attributes to the configured trace destination. By default, trace files are written to the /trace directory of your Enterprise Gateway installation.
<i>Category</i>	Utility
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Trace Message Attributes

Return WSDL

<i>Name</i>	Return WSDL
<i>Description</i>	Returns a WSDL file from the Web Services Repository. This filter is configured automatically when a WSDL file is imported into the repository.
<i>Category</i>	Web Service
<i>Required Attributes</i>	http.headers http.request.uri webservice.context
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	
<i>Tutorial</i>	Return WSDL

Set Web Service Context

<i>Name</i>	Set Web Service Context
<i>Description</i>	Specifies which service to take resources from in the Web Service Repository. For example, if you set this filter to a <code>getQuote</code> service in the repository, and configure the Re-

	turn WSDL filter , the WSDL definition for the <code>getQuote</code> service is returned when a WSDL request is received.
<i>Category</i>	Web Service
<i>Required Attributes</i>	
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	service.name webservice.context
<i>Tutorial</i>	Set Web Service Context

Web Service Filter

<i>Name</i>	Web Service Filter
<i>Description</i>	Controls and validates requests to the Web Service and responses from the Web Service. This is automatically generated as the Service Handler when a WSDL file is imported into the Web Services Repository.
<i>Category</i>	Web Service
<i>Required Attributes</i>	content.body http.headers http.request.verb
<i>Consumed Attributes</i>	
<i>Generated Attributes</i>	http.destination.host http.destination.port http.destination.protocol http.headers http.request.connection.error http.request.uri http.response.info http.response.status http.response.version service.name webservice.context
<i>Tutorial</i>	Web Service Filter

Glossary of Terms

ASN.1

ASN.1 (Abstract Syntax Notation One) is a standard format for transmitting messages over a network. The standard defines a syntax for describing the structure of a message, and also rules for encoding the various data types contained within the message. ASN.1 is defined in the following ISO standards:

- ISO 8824/ITU X.208 specifies the ASN.1 syntax.
- ISO 8825/ITU X.209 specifies the encoding rules for ASN.1

Base64

A method of encoding 8-bit characters as ASCII printable characters. It is typically used to encode binary data so that it may be sent over text-based protocols such as HTTP and SMTP. Base64 is a scheme where 3 bytes are concatenated, then split to form 4 groups of 6-bits each; and each 6-bits gets translated to an encoded printable ASCII character, via a table lookup. The specification is described in RFC 2045.

CA

A Certificate Authority (CA) issues digital certificates (especially X.509 certificates) and vouches for the binding between the data items in a certificate.

cacert

A file used to keep the root certificates of signing authorities. The default password is *changeit*. It is typically stored in `c:\jdk1.6\jre\lib\security\cacerts`. Each entry is identified by a unique alias, and is either a key entry or a certificate entry. Key entries consist of a key pair, whereas certificate entries consist of just a certificate.

Since you implicitly trust all the Certificate Authorities in the cacerts file for code signing and verification, you must manage the cacerts file carefully. The cacerts file should contain only certificates of the CAs you trust.

CRL

A Certificate Revocation List is a signed list indicating a set of certificates that are no longer considered valid by the certificate issuer. CRLs may be used to identify revoked public-key certificates or attribute certificates and may represent revocation of certificates issued to authorities or to users. The term CRL is also commonly used as a generic term applying to different types of revocation lists.

DER

Distinguished Encoding Rules is a type of ASN.1 encoding and is widely used to define the format of X.509 certificates.

DName

An identifier that uniquely represents an object in the X.500 Directory Information Tree (DIT). A DName is a set of attribute values that identify the path leading from the base of the DIT to the object that is named. An X.509 public-key certificate or CRL contains a DName that identifies its issuer, and an X.509 attribute certificate contains a DN or other form of name that identifies its subject.

DOM

Document Object Model (DOM) is a generic interface (platform- and language-neutral) that allows external programs to edit a document's contents, structure, and style.

DTD

A Document Type Definition defines a formal grammar for specifying the structure of an XML document. An XML document is said to be *valid* if it conforms to the syntactic rules specified in the DTD.

ISO

ISO is a worldwide consortium of national standards bodies from more than 140 countries. The goal of ISO is to promote standardization in the world with a view to facilitating the international exchange of goods and services, and to develop cooperation in scientific, technological and economic activity.

keystore

The keystore file of the JDK contains your public and private keys. It has a file name ".keystore", the peculiar leading dot makes the file read-only in Unix. It is stored in PKCS #12 format, contains both public and private keys, and is protected by a passphrase.

LDAP

LDAP is a "lightweight" version of Directory Access Protocol (DAP), which is part of X.500, a standard for directory services in a network. An LDAP directory stores information on resources in a hierarchical fashion. This makes data

retrieval very efficient.

OCSF

Online Certificate Status Protocol is an automated certificate checking network protocol. A client will query the OCSF responder for the status of a certificate. The responder returns whether the certificate is still trusted by the CA that issued it.

PEM

PEM (Privacy Enhanced Mail) was originally intended for securing Internet mail through authentication, message integrity and confidentiality using various encryption techniques. Its scope was widened in later years for use in a broader range of applications, such as Web Servers. Its format is essentially a base64-encoded certificate wrapped in *BEGIN CERTIFICATE* and *END CERTIFICATE* directives.

PKCS#12

PKCS#12 is a standard for storing private keys and certificates securely. It is used in (among other things) Netscape and Microsoft Internet Explorer with their import and export options.

Private-Key

The secret component of a pair of cryptographic keys used for asymmetric cryptography.

Public-Key

The publicly-disclosable component of a pair of cryptographic keys used for asymmetric cryptography.

SAML

Security Assertion Markup Language (SAML) is an XML standard for establishing trust between entities. SAML assertions can contain identity information about users (authentication assertions) and also information about the access permissions of users (authorization assertions). The basic idea is that when a user is authenticated at one site, that site issues a SAML authentication assertion and gives it to the user. The user can then use this assertion in requests at other affiliated sites. These sites need only check the details contained within the authentication assertion in order to authenticate the user. In this way, SAML allows authentication and authorization information to be shared between separate sites.

SAX

The Simple API for XML is an interface that allows applications to read in XML data. SAX is an *event-based* interface, which means that it responds to certain *events*. SAX is a read-only interface, which means that it cannot be used to generate XML elements in the same way that the DOM can. However, it is generally more efficient than DOM for reading XML documents since it does not keep the entire XML tree in memory like DOM parsing does.

Signature

A value computed with a cryptographic algorithm and appended to a data object in such a way that any recipient of the data can use the signature to verify the data's origin and integrity.

SOAP

SOAP, or Simple Object Access Protocol is an XML-based object invocation protocol. SOAP was originally developed for distributed applications to communicate over HTTP and through corporate firewalls. SOAP defines the use of XML and HTTP to access services, objects and servers in a platform-independent manner. SOAP provides a way to access services, objects, and servers in a completely platform-independent manner. SOAP is a wire protocol that can be used to facilitate highly ultra-distributed architecture.

SOAP is simple. It is nothing more and nothing less than a protocol that defines how to access services, objects, and servers in a platform-independent manner using HTTP (also SMTP) and XML. See the [Simple Object Access Protocol Specification](http://www.w3.org/TR/SOAP/) [http://www.w3.org/TR/SOAP/] for more details.

SSL

Secure Sockets Layer (SSL) is an encrypted communications protocol for sending information securely across the Internet. It sits just above the transport layer, and below the application layer and transparently handles the encryption and decryption of data when a client establishes a secure connection to the server. It optionally provides peer entity authentication between client and server.

TLS

Transport Layer Security is the successor to SSL 3.0. Like SSL, it allows applications to communicate over a secure channel.

UDDI

Universal Description, Discovery, and Integration (UDDI) is an XML-based lookup service for locating Web Services in an Internet scenario. See the [Universal Description Discovery Integration \(UDDI\) standard](http://www.uddi.org/) [http://www.uddi.org/] for more details.

URI

Uniform Resource Identifiers (URIs), are a platform-independent way to specify a file or resource somewhere on the web. Strictly speaking, every URL is also a URI, but not every URI is also a URL. Two RFCs specify the format of a URI:

- [RFC 2396: Uniform Resource Identifiers \(URI\): Generic Syntax](http://www.faqs.org/rfcs/rfc2396.html) [http://www.faqs.org/rfcs/rfc2396.html]
- [RFC 2732: Format for Literal IPv6 Addresses in URIs](http://www.faqs.org/rfcs/rfc2732.html) [http://www.faqs.org/rfcs/rfc2732.html]

WSDL

Web Services Description Language (WSDL) is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services).

WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate, however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET/POST, and MIME. See the [Web Services Description Language Specification](http://www.w3.org/TR/wsdl) [http://www.w3.org/TR/wsdl] for more details.

X509

X509 is the standard which defines the contents and data format of a public-key certificate.

XKMS

XML Key Management Specification (XKMS) uses the relative simplicity of XML to provide key management services so that a Web Service can query the trustworthiness of a user's certificate over the Internet. XKMS aims to simplify application building by separating digital-signature handling and encryption from the applications themselves. See the [XML Key Management Specification](http://www.w3.org/TR/xkms/) [http://www.w3.org/TR/xkms/] for more details.

XPath

XML Path Language (XPath), is a language that describes how to locate and process specific parts of an XML document. See the [XML Path Language Specification](http://www.w3.org/TR/xpath) [http://www.w3.org/TR/xpath] for more details.

XSL

XML Stylesheet Language is used to convert XML documents into different formats, the most common of which is HTML. In a typical scenario, an XML document will reference an XSL stylesheet, which will define how the XML elements of the document should be displayed as HTML. Therefore, a clear separation of content and presentation is achieved.

XSLT

Extensible Stylesheet Language Transformations are used to convert XML documents into other formats.